

NIST Special Publication 1019
Sixth Edition

Fire Dynamics Simulator

User's Guide

Kevin McGrattan
Simo Hostikka
Jason Floyd
Randall McDermott
Marcos Vanella
Eric Mueller
Chandan Paul

<http://dx.doi.org/10.6028/NIST.SP.1019>



Fire Safety
Research Institute

NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

NIST Special Publication 1019
Sixth Edition

Fire Dynamics Simulator

User's Guide

Kevin McGrattan
Randall McDermott
Marcos Vanella
Eric Mueller

Fire Research Division, Engineering Laboratory, Gaithersburg, Maryland

Simo Hostikka
Aalto University, Espoo, Finland

Jason Floyd
Fire Safety Research Institute, UL Research Institutes, Columbia, Maryland

Chandan Paul
The George Washington University, Washington, D.C.

<http://dx.doi.org/10.6028/NIST.SP.1019>

April 4, 2025
Revision: FDS-6.10.1-0-g12efa16



U.S. Department of Commerce
Howard Lutnick, Secretary

National Institute of Standards and Technology
Craig Burkhardt, Acting NIST Director and Acting Under Secretary of Commerce for Standards and Technology

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

National Institute of Standards and Technology Special Publication 1019
Natl. Inst. Stand. Technol. Spec. Publ. 1019, 504 pages (October 2013)
CODEN: NSPUE2

FDS Developers

The Fire Dynamics Simulator and Smokeview are the products of an international collaborative effort led by the National Institute of Standards and Technology (NIST) and Fire Safety Research Institute, UL Research Institutes. Its developers and contributors are listed below.

Principal Developers of FDS

Kevin McGrattan, NIST, Gaithersburg, Maryland
Simo Hostikka, Aalto University, Espoo, Finland
Jason Floyd, Fire Safety Research Institute, UL Research Institutes, Columbia, Maryland
Randall McDermott, NIST, Gaithersburg, Maryland
Marcos Vanella, NIST, Gaithersburg, Maryland
Eric Mueller, NIST, Gaithersburg, Maryland
Chandan Paul, The George Washington University, Washington, D.C.

Principal Developer of Smokeview

Glenn Forney, NIST, Gaithersburg, Maryland

Collaborators and Contributors

Anthony Hamins, NIST, Gaithersburg, Maryland
Jonathan Hodges, Jensen Hughes, Blacksburg, Virginia
Emanuele Gissi, Corpo Nazionale dei Vigili del Fuoco, Italy
William Mell, U.S. Forest Service, Seattle, Washington
Julio Cesar Silva, Rio on Fire, Orlando, Florida
Craig Weinschenk, Fire Safety Research Institute, UL Research Institutes, Columbia, Maryland

About the Developers

Kevin McGrattan is a mathematician in the Fire Research Division of NIST. He received a bachelor of science degree from the School of Engineering and Applied Science of Columbia University in 1987 and a doctorate at the Courant Institute of New York University in 1991. He joined the NIST staff in 1992 and has since worked on the development of fire models, most notably the Fire Dynamics Simulator.

Simo Hostikka is an associate professor of fire safety engineering at Aalto University School of Engineering, since January 2014. Before joining Aalto, he worked as a Principal Scientist and Team Leader at VTT Technical Research Centre of Finland. He received a master of science (technology) degree in 1997 and a doctorate in 2008 from the Department of Engineering Physics and Mathematics of the Helsinki University of Technology. He is the principal developer of the radiation and solid phase sub-models within FDS.

Jason Floyd is a Principle Research Engineer at the Fire Safety Research Institute, part of the UL Research Institutes, in Columbia, Maryland. He received a B.S. (1993), M.S (1995), and a Ph.D. (2000) from the Nuclear Engineering Program of the University of Maryland. After graduating, he was awarded a National Research Council Post-Doctoral Fellowship at the Building and Fire Research Laboratory of NIST. He is a principal developer of the combustion, control logic, aerosol, droplet evaporation, and HVAC sub-models within FDS.

Randall McDermott joined the Fire Research Division at NIST in 2008. He received a B.S. from the University of Tulsa in Chemical Engineering in 1994 and a Ph.D. from the University of Utah in 2005. His research interests include subgrid-scale models and numerical methods for large-eddy simulation, turbulent combustion, immersed boundary methods, and Lagrangian particle methods.

Marcos Vanella joined the Fire Research Division at NIST in 2019. He received diplomas in Mechanical and Aeronautical Engineering from the National University of Cordoba, Argentina, and M.S. and Ph.D. degrees in Mechanical Engineering from the University of Maryland, College Park. His research interests include computer simulation and scientific software development applied to engineering systems, mainly in the areas of fluid flow and multiphysics interaction problems.

Glenn Forney is a computer scientist in the Fire Research Division of NIST. He received a bachelor of science degree in mathematics from Salisbury State College and a master of science and a doctorate in mathematics from Clemson University. He joined NIST in 1986 (then the National Bureau of Standards) and has since worked on developing tools that provide a better understanding of fire phenomena, most notably Smokeview, an advanced scientific software tool for visualizing Fire Dynamics Simulation data.

Eric Mueller joined the Fire Research Division at NIST in 2021. He received a B.S. in Engineering Physics from Tufts University (2010), an M.S. from Worcester Polytechnic Institute (2012), and a Ph.D. from the University of Edinburgh (2017), both in fire safety engineering. His research interests include the

development of sub-models relevant to heat and mass transfer in wildland and wildland-urban interface fires.

Chandan Paul joined the Fire Research Division at NIST in 2023. He received a Ph.D. in Mechanical Engineering with a minor in Computational Science from The Pennsylvania State University in 2018. His research interests include the development of combustion and radiative heat transfer models for practical engineering applications.

Emanuele Gissi is a professional fire chief of the Italian Fire and Rescue Service (CNVVF) since 2002. He received a Ph.D. in Engineering Physics from the University of Ancona, Italy, in 2000. In addition to managing fire brigades, his main interest is bridging the gap between fire research and fire safety engineering practice. He develops open source tools for FDS.

Anthony Hamins joined the Fire Research Division at NIST in 1989. He received his B.S. from the University of California, Berkeley, in Physics and is Ph.D. from U.C. San Diego in Engineering Physics in 1985. His research interests include fire model validation, fire dynamics, heat and mass transfer processes in fires of multiple scales, fire suppression, flame structure, wildland-urban interface fires, and micro-gravity combustion.

Jonathan Hodges is a Lead Engineer in the Research, Development, Testing, and Evaluation Division at Jensen Hughes in Blacksburg, Virginia. He received a B.S. (2012) and M.S. (2014) from Clemson University and Ph.D. (2018) from Virginia Tech in Mechanical Engineering. His research interests include compartment fire dynamics, heat transfer from fires to surfaces, predicting fire growth, wildland-urban interface fires, and the intersection of physics modeling and artificial intelligence.

William (Ruddy) Mell is an applied mathematician currently at the U.S. Forest Service in Seattle, Washington. He holds a B.S. degree from the University of Minnesota (1981) and doctorate from the University of Washington (1994). His research interests include the development of large-eddy simulation methods and sub-models applicable to the physics of large fires in buildings, vegetation, and the wildland-urban interface.

Julio Cesar Silva is a Lead Researcher at Rio on Fire. He worked in the Fire Research Division of NIST as a Guest Researcher from National Council for Scientific and Technological Development, Brazil. He received a M.Sc. in 2010 and a doctorate in 2014 from Federal University of Rio de Janeiro in Civil Engineering. His research interests include fire-structure interaction and coupling strategies between FDS and finite-element codes.

Craig Weinschenk is a Director of Research at the Fire Safety Research Institute, part of the UL Research Institutes, in Columbia, Maryland. He worked in the Fire Research Division at NIST as a National Research Council Postdoctoral Research Associate in 2011. He received a B.S. from Rowan University in 2006 in Mechanical Engineering. He received an M.S. in 2007 and a doctorate in 2011 from The University of Texas at Austin in Mechanical Engineering. His research interests include numerical combustion, fire-structure interaction, and human factors research of fire-fighting tactics.

Preface

This Guide describes how to use the Fire Dynamics Simulator (FDS). Because new features are added periodically, check the current version number on the inside front jacket of this manual.

Note that this Guide does not provide the background theory for FDS. A four volume set of companion documents, referred to collectively as the FDS Technical Reference Guide [1], contains details about the governing equations and numerical methods, model verification, experimental validation, and configuration management. The FDS User's Guide contains limited information on how to operate Smokeview, the companion visualization program for FDS. Its full capability is described in the Smokeview User's Guide [2].

Disclaimer

The US Department of Commerce makes no warranty, expressed or implied, to users of the Fire Dynamics Simulator (FDS), and accepts no responsibility for its use. Users of FDS assume sole responsibility under Federal law for determining the appropriateness of its use in any particular application; for any conclusions drawn from the results of its use; and for any actions taken or not taken as a result of analysis performed using these tools.

Users are warned that FDS is intended for use only by those competent in the fields of fluid dynamics, thermodynamics, heat transfer, combustion, and fire science, and is intended only to supplement the informed judgment of the qualified user. The software package is a computer model that may or may not have predictive capability when applied to a specific set of factual circumstances. Lack of accurate predictions by the model could lead to erroneous conclusions with regard to fire safety. All results should be evaluated by an informed user.

Throughout this document, the mention of computer hardware or commercial software does not constitute endorsement by NIST, nor does it indicate that the products are necessarily those best suited for the intended purpose.

Acknowledgments

The Fire Dynamics Simulator, in various forms, has been under development for almost 25 years. It was first released to the public in 2000. Since then, continued improvements have been made to the software based largely on feedback from its users. Included below are some who made important contributions related to the application of FDS.

- Ulf Wickström of Luleå University of Technology, Sweden, provided guidance and articles on the adiabatic surface temperature concept.
- At NIST, Dan Madrzykowski, Doug Walton, Bob Vettori, Dave Stroup, Steve Kerber, Nelson Bryner, and Adam Barowy have used FDS and Smokeview as part of several investigations of fire fighter line of duty deaths. They have provided valuable information on the model's usability and accuracy when compared to large scale measurements made during fire reconstructions.
- Bryan Klein of Thunderhead Engineering assisted in adding cross-referencing functionality to this document, making it easier to view electronically. He also designed the on-line services for revision control, bug reporting, and general discussion of topics related to FDS.
- At VTT, Joonas Ryynänen implemented and documented the FED/FIC routine.
- The US Nuclear Regulatory Commission has provided financial support for the verification and validation of FDS, along with valuable insights into how fire models are used as part of probabilistic risk assessments of nuclear facilities. Special thanks to Mark Salley and Dave Stroup.
- The Society of Fire Protection Engineers (SFPE) sponsors a training course on the use of FDS and Smokeview. Chris Wood of ArupFire, Dave Sheppard of the US Bureau of Alcohol, Tobacco and Firearms (ATF), and Doug Carpenter of Combustion Science and Engineering developed the materials for the course, along with Morgan Hurley of the SFPE.
- David McGill of Seneca College, Ontario, Canada has conducted a remote-learning course on the use of FDS, and he has also maintained a web site that has provided valuable suggestions from users.
- Paul Hart of Swiss Re, GAP Services, and Pravinray Gandhi of Underwriters Laboratories provided useful suggestions about water droplet transport on solid objects.
- François Demouge of the Centre Scientifique et Technique du Bâtiment (CSTB) in France assisted with implementation of synthetic turbulence inflow boundary conditions.
- Max Gould, Summer Undergraduate Research Fellow, assisted in the testing and verification of non-standard boundary treatment methods.
- Salah Benkorichi at Buro Happold, UK, helped develop the verification cases for 3D heat transfer.

- Daniel Haarhoff of Jülich Supercomputing Centre, Germany, helped to implement OpenMP directives and profile the code for FDS v6.
- Timo Korhonen of VTT, Finland, was the principle developer of FDS+Evac (FDS versions prior to 6.7.8).
- Vivien Lecoustre, while at the University of Maryland, College Park, improved RadCal which included adding 8 additional new fuel species to its database.
- Anna Matala, VTT, Finland, helped develop the 1D pyrolysis model and pyrolysis kinetics parameter estimation methods.
- Kristopher Overholt, while a post-doc at NIST, was the principal developer of “firebot”, the FDS continuous integration framework.
- Benjamin Ralph, University of Edinburgh, UK, helped to develop mass and energy transport in the HVAC model and the system curve approach for defining the operation point of a fan.
- Topi Sikanen, VTT, Finland, worked on the droplet and evaporation algorithms.
- Ben Trettel, The University of Texas at Austin, developed high-order droplet tracking methods and primary droplet breakup models.

Finally, on the following pages is a list of individuals and organizations who have volunteered their time and effort to “beta test” FDS and Smokeview prior to its official release. Their contribution is invaluable because there is simply no other way to test all of the various features of the model.

FDS 6 Beta Testers	
Mohammed Assal	CFD Algeria
Choon-Bum Choi	KyungWon Tech Co., Ltd. (KW Tech), Korea
William J. Ferrante	Roosevelt Fire District, Hyde Park, New York, USA
Emanuele Gissi	Corpo Nazionale dei Vigili del Fuoco, Italy
Timothy M. Groch	Engineering Planning and Management, Inc., Framingham, Massachusetts, USA
Georges Guigay	Mannvit Engineering, Iceland
Simon J. Ham	Fire Safety Engineering Consultants Limited, UK
Chris Lautenberger	Reax Engineering, Berkeley, California, USA
Tim McDonald	Endress Ingenieurgesellschaft mbH, Germany
Dave McGill	Seneca College, Ontario, Canada
Adrian Milford	Sereca Fire Consulting Ltd., British Columbia, Canada
Luca Nassi	National Fire Department, Italy
Stephen Olenick	Combustion Science and Engineering, Inc., Columbia, Maryland, USA
Natalie Ong	Arup Fire Singapore
Chris Salter	Hoare Lea and Partners, UK
Joakim Sandström	LTU/Brandskyddslaget, Sweden
Julio Cesar Silva	Federal University of Rio de Janeiro, Brazil
Boris Stock	BFT Cognos GmbH, Aachen, Germany
Csaba Szilagyi	OPTOMM Ltd., Budapest, Hungary
Giacomo Villi	Università di Padova, Italy
Andreas Vischer	Wijnveld//Ingenieure, Osnabrück, Germany
Christopher Wood	FireLink, LLC, Tewksbury, Massachusetts, USA

Contents

FDS Developers	i
About the Developers	iii
Preface	v
Disclaimer	vii
Acknowledgments	ix
Contents	xiii
List of Figures	xxv
List of Tables	xxix
I The Basics of FDS	1
1 Introduction	3
1.1 Features of FDS	3
1.2 What's New in FDS 6?	4
2 Getting Started	7
2.1 How to Acquire FDS and Smokeview	7
2.2 Computer Hardware Requirements	7
2.3 Computer Operating System (OS) and Software Requirements	8
2.4 Installation Testing	8
3 Running FDS	9
3.1 Computer Basics	9
3.1.1 A Brief Primer on Computer Hardware	9
3.1.2 Two Ways to Use Multiple Processors	10
3.2 Launching an FDS Job	12
3.2.1 Single Computer Running MS Windows	12
3.2.2 Multiple Computers Running MS Windows	12
3.2.3 macOS	13
3.2.4 Linux	14
3.2.5 Using MPI and OpenMP Together	14
3.2.6 Running Very Large Jobs	15

3.3	Efficiency of Multi-Process Simulations	17
3.3.1	MPI Efficiency	17
3.3.2	OpenMP Efficiency	18
3.4	Monitoring Progress	19
4	User Support	21
4.1	The Version Number	21
4.2	Common Error Statements	22
4.3	Support Requests and Bug Tracking	24
II	Writing an FDS Input File	27
5	The Basic Structure of an Input File	29
5.1	Naming the Input File	29
5.2	Namelist Formatting	29
5.3	Input File Structure	30
5.4	Concatenating Input Files	33
5.5	Protecting Old Cases	33
5.6	Stopping and Restarting Calculations (.restart)	33
6	Setting the Bounds of Time and Space	35
6.1	Naming the Job	35
6.2	Simulation Time	36
6.2.1	Basics	36
6.2.2	Controlling the Time Step	36
6.2.3	Steady-State Applications	38
6.3	Computational Meshes	39
6.3.1	Basics	39
6.3.2	Two-Dimensional and Axially-Symmetric Calculations	39
6.3.3	Multiple Meshes	40
6.3.4	Mesh Alignment	41
6.3.5	Mesh Stretching	43
6.3.6	Mesh Resolution	45
7	Model Geometry	47
7.1	Specifying Boundary Conditions	47
7.2	Creating Rectilinear Obstructions	48
7.2.1	Basics	48
7.2.2	Thin Obstructions	48
7.2.3	Specified Versus Actual Areas	49
7.2.4	Overlapping Obstructions	49
7.2.5	Preventing Obstruction Removal	49
7.2.6	Transparent or Outlined Obstructions	50
7.2.7	Method for avoiding mesh interfaces in blockage outline view	50
7.2.8	Creating Holes in Obstructions	51
7.3	Creating Unstructured Geometry (Beta)	52
7.3.1	My First Geometry	52

7.3.2	Unstructured Geometry Basics	52
7.3.3	Triangulated Surfaces Quality	52
7.3.4	Intersections	54
7.3.5	Coloring and Texture Maps	54
7.3.6	Self-Generated Geometries	54
7.3.7	Generating Complex Geometries	59
7.3.8	Transforming Objects	59
7.3.9	Reading Geometry Node Locations And Connectivity Data From Binary	59
7.3.10	Handling Split Cells and Thin Geometries	60
7.3.11	GEOM Limitations	60
7.3.12	The Big Picture, Run Time, and Accuracy	61
7.4	Applying Surface Properties	62
7.4.1	Basics	62
7.4.2	Special Vents	63
7.4.3	Controlling Vents	65
7.4.4	Trouble-Shooting Vents	66
7.5	Coloring Obstructions, Geometry, Vents, Surfaces, and Meshes	67
7.5.1	Colors	67
7.5.2	Texture Maps	67
7.6	Repeated Objects: The MULT Namelist Group (Table 23.20)	70
7.6.1	Using MULT for Mesh Refinement	71
7.6.2	Using MULT to make shapes out of obstructions	71
8	Thermal Boundary Conditions	77
8.1	Basics	77
8.2	Surface Temperature and Heat Flux	77
8.2.1	Specified Solid Surface Temperature	77
8.2.2	Convective Heat Transfer Options	78
8.2.3	Adiabatic Surfaces	81
8.3	One-Dimensional Heat Conduction in Solids	82
8.3.1	Structure of Solid Boundaries	82
8.3.2	Thermal Properties	83
8.3.3	Back Side Boundary Conditions	84
8.3.4	Initial Solid Temperature	84
8.3.5	Walls with Different Materials Front and Back	84
8.3.6	Specified Internal Heat Source	85
8.3.7	Non-Planar Walls and Targets	86
8.3.8	Solid Phase Numerical Gridding Issues	86
8.3.9	Specified Front and Back Surface Temperature	87
8.4	3-D Heat Conduction (Beta)	89
8.4.1	Basics	89
8.4.2	Limitations	90
8.4.3	Surface Linings	92
8.4.4	Working with Thin Plates	93
8.4.5	1-D Heat Conduction in Solids of Varying Thickness	94
9	Fire and Pyrolysis	95
9.1	Specified Heat Release Rate	95

9.1.1	A Radially-Spreading Fire	96
9.1.2	Compensating for the unresolved surface area	97
9.1.3	Thermally-Thick Solids that Burn at a Specified Rate	97
9.1.4	Scaling Pyrolysis (SPyro) Model: Scaled Burning Rate from Cone Data	98
9.2	Complex Pyrolysis Models	101
9.2.1	Reaction Mechanism	101
9.2.2	Solid Phase Gas Transport	102
9.2.3	Reaction Rates	103
9.2.4	Shrinking and Swelling materials	107
9.2.5	Multiple Solid Phase Reactions	108
9.2.6	The Heat of Reaction	109
9.2.7	Liquid Fuels	109
9.2.8	Fuel Burnout	111
9.2.9	Solid Fuels that can Burn Away	112
9.3	Testing a Pyrolysis Model	117
9.3.1	Simulating the Cone Calorimeter	117
9.3.2	Simulating Bench-scale Measurements like the TGA, DSC, and MCC	119
9.4	Pyrolysis and Energy Conservation	122
10	Ventilation	125
10.1	Simple Vents, Fans and Heaters	125
10.1.1	Simple Supply and Exhaust Vents	125
10.1.2	Total Mass Flux	126
10.1.3	Heaters	126
10.1.4	Louvered Vents	127
10.1.5	Specified Normal Velocity Gradient	127
10.1.6	Species and Species Mass Flux Boundary Conditions	128
10.1.7	Tangential Velocity Boundary Conditions at Solid Surfaces	128
10.1.8	Synthetic Turbulence Inflow Boundary Conditions	129
10.1.9	Random Mass Flux Variation	131
10.2	HVAC Systems	132
10.2.1	HVAC Duct Parameters	133
10.2.2	HVAC Dampers	135
10.2.3	HVAC Node Parameters	136
10.2.4	HVAC Fan Parameters	137
10.2.5	HVAC Filter Parameters	140
10.2.6	HVAC Aircoil Parameters	141
10.2.7	Louvered HVAC Vents	142
10.2.8	HVAC Mass Transport	142
10.2.9	Specified Flow vs. Unspecified Flow	143
10.3	Sealed Compartments, Leakage, and Void Spaces	144
10.3.1	Specifying Pressure Zones	144
10.3.2	Leaks	147
10.3.3	Breaking Pressure Zones	149
10.3.4	Filling Pressure Zones	150
10.4	Pressure Boundary Conditions	151
10.5	Special Flow Profiles	152

11	User-Specified Functions	155
11.1	Time-Dependent Functions	155
11.2	Temperature-Dependent Functions	157
11.3	Spatially-Dependent Velocity Profiles	158
11.4	Scaling, Rotation and Translation	158
12	Chemical Species	161
12.1	Specifying Primitive Species	161
12.1.1	Basics	162
12.1.2	Pre-Defined Gas and Liquid Properties	163
12.1.3	User-Defined Gas and Liquid Properties	163
12.1.4	Air	167
12.1.5	Two Gas Species with the Same Properties	168
12.1.6	Soot	169
12.2	Specifying Lumped Species (Mixtures of Primitive Species)	170
12.2.1	Combining Lumped and Primitive Species	171
13	Combustion	173
13.1	Single-Step, Mixing-Controlled Combustion	173
13.1.1	Simple Chemistry Parameters	173
13.1.2	Heat of Combustion	175
13.1.3	Two-Step Simple Chemistry	176
13.1.4	Complete Heat of Combustion	179
13.1.5	Turbulent Combustion	179
13.1.6	Flame Extinction	179
13.1.7	Piloted Ignition	182
13.1.8	Local Quenching (AIT per ZONE)	183
13.2	Complex Stoichiometry	184
13.2.1	Balancing the Atoms	184
13.2.2	Complex Fuel Molecules	185
13.2.3	Multiple Fast Reactions	187
13.2.4	Multiple Fuels	188
13.2.5	Using the EQUATION input parameter	190
13.3	Finite Rate Combustion	192
13.3.1	Multiple Step Reaction	193
13.3.2	Finite Rate Chemistry using a Detailed Chemical Mechanism	194
13.3.3	Reaction Rates from Equilibrium Constants	195
13.3.4	Third Body Reactions	195
13.3.5	Fall-off Reactions	195
13.3.6	Catalysts	196
13.3.7	Chemical Time Integration	196
13.4	Aerosol Deposition	199
13.4.1	Example Case: Soot Deposition from a Propane Flame	199
13.4.2	Soot Surface Oxidation	200
13.5	Aerosol Agglomeration	200
13.6	Aerosol Scrubbing	201
13.7	Vapor Condensation	202

14	Radiation	203
14.1	Basic Radiation Parameters: The <code>RADI</code> Namelist Group	203
14.1.1	Radiation Option 1. No Radiation Transport	203
14.1.2	Radiation Option 2. Optically-Thin Limit; Specified Radiative Fraction	204
14.1.3	Radiation Option 3. Optically-Thick; Specified Radiative Fraction	205
14.1.4	Radiation Option 4. Optically-Thick; Unspecified Radiative Fraction	205
14.2	Spatial and Temporal Resolution of the Radiation Transport Solver	205
14.3	Absorption Coefficient of Gases and Soot	206
14.3.1	Gray Gas Model (default)	207
14.3.2	Wide Band Model (Box Model)	207
14.3.3	Weighted Sum of Gray Gases (WSGG) Model (Experimental)	207
14.4	Radiative Absorption and Scattering by Particles	208
14.5	Other Considerations	208
15	Particles and Droplets	211
15.1	Basics	211
15.2	Massless Particles	212
15.3	Liquid Droplets	213
15.3.1	Thermal Properties	213
15.3.2	Radiative Properties	215
15.3.3	Size Distribution	216
15.3.4	Dense Clouds of Droplets	218
15.3.5	Primary Breakup	218
15.3.6	Secondary Breakup	219
15.3.7	Warning Messages Related to Droplets	219
15.4	Solid Particles	220
15.4.1	Basic Geometry and Boundary Conditions	220
15.4.2	Drag	221
15.4.3	Radiation Absorption and Emission	221
15.4.4	Size Distribution	222
15.4.5	Solid Particle Movement on Solid Surfaces	223
15.4.6	Particle Orientation	223
15.4.7	Gas Generating Particles	223
15.4.8	Porous Media	224
15.4.9	Screens	225
15.4.10	Electrical Cables	225
15.4.11	Target Particles	227
15.5	Particle Insertion	228
15.5.1	Particles Introduced at a Solid Surface	228
15.5.2	Particles or Droplets Introduced at a Sprinkler or Nozzle	230
15.5.3	Particles or Droplets Introduced within a Volume	230
15.5.4	Controlled Particle Orientation	234
15.6	Particle Removal	235
15.7	Suppression by Water	235
15.7.1	Droplet Movement on Solid Surfaces	235
15.7.2	Reduction of the Burning Rate	236
16	Wind and Atmospheric Stratification	239

16.1	Wind Method 1: Specified Wind Speed and Direction	239
16.2	Wind Method 2: Monin-Obukhov Similarity	241
16.2.1	Basic Equations	241
16.2.2	Applying Monin-Obukhov Profiles to FDS	243
16.3	Wind Method 3: Advanced Meteorological Concepts	245
16.3.1	Pressure Gradient Force	245
16.3.2	Coriolis Force	246
16.3.3	Geostrophic Wind	246
16.3.4	Surface Roughness	247
16.3.5	Thermal Boundary Conditions at the Ground	248
16.3.6	Example	248
16.4	Wind Method 4: The “Wall of Wind”	250
16.5	Temperature Stratification	251
16.5.1	Stack Effect	251
16.6	External Boundary Conditions	253
17	Wildland Fire Spread	255
17.1	Thermal Degradation Model for Vegetation	255
17.1.1	Solid Phase	255
17.1.2	Gas Phase	260
17.1.3	Examples	260
17.2	Lagrangian Particle Model	262
17.2.1	Trees	264
17.2.2	Bulk Density Input Files	265
17.2.3	Firebrands	266
17.2.4	Ember Generation From Particles	267
17.3	Boundary Fuel Model	268
17.3.1	Burnout Time	268
17.4	Comparing the Particle and Boundary Fuel Models of Vegetation	270
17.4.1	Combustible Load	270
17.4.2	Vegetation Drag	270
17.4.3	Vegetation Radiation Absorption	270
17.4.4	Vegetation Convective Heating	271
17.5	Level Set Model for Wildland Fire Spread	273
17.5.1	Level Set Vegetation Drag	274
17.5.2	Simple Test Cases	274
17.5.3	Ember Generation From Surfaces	278
17.5.4	Ember Ignition	279
17.5.5	Wildfire Spread over Realistic Terrain using QGIS	279
18	Devices and Control Logic	283
18.1	Device Location and Orientation	283
18.2	Device Output	284
18.3	Special Device Properties	285
18.3.1	Sprinklers	285
18.3.2	Nozzles	290
18.3.3	Specified Entrainment (Velocity Patch)	290
18.3.4	Heat Detectors	291

18.3.5	Smoke Detectors	292
18.3.6	Beam Detection Systems	293
18.3.7	Aspiration Detection Systems	296
18.3.8	Fire Depth	297
18.4	Basic Control Logic	297
18.4.1	Creating and Removing Obstructions	298
18.4.2	Activating and Deactivating Vents	299
18.5	Advanced Control Functions: The CTRL Namelist Group	300
18.5.1	Control Functions: ANY, ALL, ONLY, and AT_LEAST	302
18.5.2	Control Function: TIME_DELAY	302
18.5.3	Control Function: DEADBAND	303
18.5.4	Control Function: RESTART and KILL	303
18.5.5	Control Function: CUSTOM	304
18.5.6	Control Function: Math Operations	304
18.5.7	Control Function: PID Control Function	305
18.5.8	Control Function: PERCENTILE	305
18.5.9	Combining Control Functions: A Deluge System	306
18.5.10	Combining Control Functions: A Dry Pipe Sprinkler System	307
18.5.11	Example Case: activate_vents	308
18.6	Controlling a RAMP	308
18.6.1	Changing the Independent Variable	308
18.6.2	Changing the Dependent Variable	309
18.6.3	Freezing the Output Value, Example Case: hrr_freeze	309
18.6.4	Example Case: Heat Release Rate of a Spreading Fire	310
18.7	Visualizing FDS Devices in Smokeview	312
18.7.1	Devices that Indicate Activation	312
18.7.2	Devices with Variable Properties	314
18.7.3	Objects that Represent Lagrangian Particles	316
18.8	External Control of FDS (Beta)	318
19	Numerical Considerations	321
19.1	Simulation Mode	321
19.2	Large Eddy Simulation Parameters	322
19.3	Numerical Stability Parameters	323
19.3.1	The Courant-Friedrichs-Lewy (CFL) Constraint	323
19.3.2	The Von Neumann Constraint	324
19.3.3	Stability of particle transport	325
19.3.4	Heat Transfer Constraint	325
19.4	Flux Limiters	325
19.5	Limiting the Bounds of Key Variables	326
19.5.1	Temperature	326
19.5.2	Density	326
20	Changing the Initial Conditions	329
20.1	Gas Species	329
20.2	Temperature and Pressure	330
20.3	Heat Release Rate Per Unit Volume	330
20.4	Velocity Field	331

20.4.1	Default Initial Velocity Field	331
20.4.2	Turning off the Flow Field	331
20.4.3	One-Dimensional Velocity Component Fields	331
20.5	Nonuniform Initial Fields: Velocity, Temperature, Species	331
20.6	Unfreezing the Initial Flow Field	333
20.7	Gravity	333
21	Pressure	335
21.1	Accuracy of the Pressure Solver	336
21.1.1	Optional Pressure Solvers	336
21.1.2	Example Case: <code>Pressure_Solver/duct_flow</code>	339
21.1.3	Example Case: <code>Pressure_Solver/dancing_eddies</code>	339
21.1.4	Example Case: <code>Random Obstructions</code>	341
21.2	Baroclinic Vorticity	342
21.3	Pressure Considerations in Long Tunnels	343
21.4	Pressure Considerations in Stairwells	347
22	Output	349
22.1	Controlling the Frequency of Output	350
22.2	Device Output: The <code>DEVC</code> Namelist Group	351
22.2.1	Gas Phase Quantity at a Single Point	351
22.2.2	Solid Phase Quantity at a Single Point	352
22.2.3	Spatially-Integrated Outputs	353
22.2.4	Temporally-Integrated Outputs	357
22.2.5	Linear Array of Point Devices	359
22.3	In-Depth Profiles within Solids: The <code>PROF</code> Namelist Group	362
22.4	Animated Planar Slices: The <code>SLCF</code> Namelist Group	363
22.5	Animated Boundary Quantities: The <code>BNDF</code> Namelist Group	364
22.6	Animated Isosurfaces: The <code>ISO</code> Namelist Group	365
22.7	Plot3D Static Data Dumps	366
22.8	SMOKE3D: Realistic Smoke and Fire	366
22.9	Particle Output Quantities	367
22.9.1	Liquid Droplets that are Attached to Solid Surfaces	367
22.9.2	Solid Particles on Solid Surfaces	367
22.9.3	Droplet and Particle Densities and Fluxes in the Gas Phase	368
22.9.4	Coloring Particles and Droplets in Smokeview	369
22.9.5	Detailed Properties of Solid Particles	369
22.10	Special Output Features	371
22.10.1	Heat Release Rate and Energy Conservation	371
22.10.2	Gas Species Mass	373
22.10.3	Mass Loss Rates	373
22.10.4	Zone Pressures	373
22.10.5	Visibility and Obscuration	373
22.10.6	Flame Height and Flame Tilt	374
22.10.7	Layer Height and the Average Upper and Lower Layer Temperatures	375
22.10.8	Thermocouples	376
22.10.9	Volume Flow	377
22.10.10	Mass Flow	378

22.10.11	Enthalpy Flow	380
22.10.12	Heat Flux	380
22.10.13	Adiabatic Surface Temperature	382
22.10.14	Extracting Detailed Radiation Data	383
22.10.15	Flame Temperature	384
22.10.16	Detailed Spray Properties	384
22.10.17	Output Associated with Thermogravimetric Analysis (TGA)	388
22.10.18	Fractional Effective Dose (FED) and Fractional Irritant Concentration (FIC) . . .	389
22.10.19	Histograms	390
22.10.20	Complex Terrain and Related Quantities	391
22.10.21	Wind and the Pressure Coefficient	391
22.10.22	Dry Volume and Mass Fractions	392
22.10.23	Aerosol and Soot Concentration	392
22.10.24	Gas Velocity	393
22.10.25	Enthalpy	393
22.10.26	Computer Performance	393
22.10.27	Output File Precision	394
22.10.28	<i>A Posteriori</i> Mesh Quality Metrics	394
22.10.29	Extinction	399
22.10.30	Fire spread over a surface	399
22.11	Extracting Numbers from the Output Data Files	400
22.12	Gas Phase Output Quantities	402
22.13	Solid Phase Output Quantities	406
22.14	Device, Control, and Other Miscellaneous Output Quantities	408
22.15	Droplet and Particle Output Quantities	409
22.16	Summary of HVAC Output Quantities	410

23 Alphabetical List of Input Parameters 413

23.1	BACK (Background species)	413
23.2	BNDF (Boundary File Parameters)	414
23.3	CATF (Concatenate Input Files Parameters)	414
23.4	CLIP (Clipping Parameters)	414
23.5	COMB (General Combustion Parameters)	415
23.6	CSVF (Comma Separated Velocity Files)	415
23.7	CTRL (Control Function Parameters)	416
23.8	DEVC (Device Parameters)	416
23.9	DUMP (Output Parameters)	418
23.10	GEOM (Unstructured Geometry Parameters)	420
23.11	HEAD (Header Parameters)	421
23.12	HOLE (Obstruction Cutout Parameters)	421
23.13	HVAC (HVAC System Definition)	422
23.14	INIT (Initial Conditions)	423
23.15	ISOF (Isosurface Parameters)	425
23.16	MATL (Material Properties)	425
23.17	MESH (Mesh Parameters)	426
23.18	MISC (Miscellaneous Parameters)	427
23.19	MOVE (Coordinate Transformation Parameters)	429
23.20	MULT (Multiplier Function Parameters)	430

23.21	OBST (Obstruction Parameters)	430
23.22	PART (Lagrangian Particles/Droplets)	432
23.23	PRES (Pressure Solver Parameters)	433
23.24	PROF (Wall Profile Parameters)	434
23.25	PROP (Device Properties)	434
23.26	RADF (Radiation Output File Parameters)	436
23.27	RADI (Radiation Parameters)	436
23.28	RAMP (Ramp Function Parameters)	437
23.29	REAC (Reaction Parameters)	438
23.30	SLCF (Slice File Parameters)	439
23.31	SM3D (Smoke3D Parameters)	440
23.32	SPEC (Species Parameters)	440
23.33	SURF (Surface Properties)	442
23.34	TABL (Table Parameters)	446
23.35	TIME (Time Parameters)	446
23.36	TRNX, TRNY, TRNZ (MESH Transformations)	447
23.37	VENT (Vent Parameters)	448
23.38	WIND (Wind and Atmospheric Parameters)	449
23.39	ZONE (Pressure Zone Parameters)	450
24	Error Codes	451
III	FDS and Smokeview Development Tools	465
25	The FDS and Smokeview Repositories	467
26	Compiling FDS	469
26.1	FDS Source Code	469
27	Output File Formats	471
27.1	Diagnostic Output (.out)	471
27.2	Heat Release Rate and Related Quantities (_hrr.csv)	472
27.3	Device Output Data (_devc.csv)	472
27.4	Control Output Data	472
27.5	Device and Control Log File	472
27.6	CPU Usage Data	473
27.7	Time Step Data	473
27.8	Gas Mass Data	473
27.9	Slice Files (.sf)	474
27.10	Plot3D Data (.xyz, .q)	474
27.11	Boundary Files (.bf)	475
27.12	Particle Data (.prt5)	475
27.13	Profile Files	476
27.14	3-D Smoke Files (.s3d)	476
27.15	Iso-surface Triangulation Files (.iso)	477
27.16	Geometry Boundary, Unstructured Slice Files (.gcf, .gsf)	478
27.17	Isosurface, Geometry and Unstructured Slice Data Files (.viso, .be)	479

27.18 Terrain Data Files (.ter)	479
27.19 FDS GEOM I/O binary format (.bingeom)	480
27.20 Unstructured Geometry (.ge, .ge2)	480
27.21 FDS HVAC I/O binary format	481
27.22 File Extension Glossary	482
Bibliography	485
A Predefined Species	493

List of Figures

3.1	MPI scaling study	18
3.2	OpenMP timing study	18
6.1	Ramping simulation time values	37
6.2	Multiple-mesh geometry	40
6.3	Rules governing the alignment of meshes	42
6.4	Piecewise-linear mesh transformation	44
6.5	Polynomial mesh transformation	44
7.1	Blockage outlines at mesh interfaces and as specified in the input file	50
7.2	Good GEOM: consistent normals on a manifold orientable surface	54
7.3	Good GEOM: a well formed triangulated surface	54
7.4	Bad GEOM: vertex shared by the two cubes	55
7.5	Bad GEOM: the edge shared by the two cubes	55
7.6	Bad GEOM: FDS cannot differentiate the internal volume	55
7.7	Good GEOM: adding a thickness	55
7.8	Bad GEOM: volume is self-intersecting	56
7.9	Linearization at intersections: sphere example	56
7.10	Handling thin geometry with cell blocking	60
7.11	Results of the <code>circular_burner</code> test case	65
7.12	Texture mapped sphere	69
7.13	An example of the multiplier function	71
7.14	Using <code>MULT</code> for mesh refinement	72
7.15	Creating an <code>OBST</code> sphere using <code>MULT</code> and <code>SHAPE</code>	73
7.16	Creating an <code>OBST</code> cylinder using <code>MULT</code> and <code>SHAPE</code>	74
7.17	Creating an <code>OBST</code> cone using <code>MULT</code> and <code>SHAPE</code>	74
7.18	Creating an <code>OBST</code> rotated box using <code>MULT</code> and <code>SHAPE</code>	75
8.1	The <code>ht3d_demo</code> test case	91
8.2	Schematic diagram of an I-beam	92
8.3	Results of the <code>checkerboard</code> test case	94
9.1	Specifying <code>HRRPUA</code> with multiple reactions	96
9.2	Extrapolating cone test data to other heat fluxes	100
9.3	Simple demonstration of the pyrolysis model	105
9.4	Results of the <code>couch</code> test case	107
9.5	A more complicated demonstration of the pyrolysis model	108
9.6	Results of the <code>methanol_evaporation</code> test case	110
9.7	Results of the <code>liquid_mixture</code> test case	111
9.8	Results of the <code>box_burn_away</code> test cases 1-4	114

9.9	Results of the <code>box_burn_away</code> test cases 5-8	115
9.10	Results of the <code>box_burn_away9</code> and <code>box_burn_away10</code> test cases	116
9.11	Results of the <code>box_burn_away11</code> test case	116
9.12	Results of the <code>box_burn_away_2D</code> test cases	117
9.13	Sample results of a <code>tga_analysis</code>	121
10.1	Results of the <code>volume_flow</code> test cases	126
10.2	The <code>tangential_velocity</code> test case	127
10.3	Synthetic Eddy Method vent profiles	130
10.4	Synthetic Eddy Method at OPEN boundary	131
10.5	An example of simplifying a complex duct	135
10.6	Example of fan curves	139
10.7	Example of a jet fan	139
10.8	Results of the <code>HVAC_aircoil</code> case	142
10.9	Results of the <code>pressure_rise</code> test case	145
10.10	Results of the <code>zone_break</code> test cases	146
10.11	Results of the <code>zone_shape</code> test case	146
10.12	Results of the <code>door_crack</code> test case	150
10.13	Snapshots of the <code>pressure_boundary</code> test case	152
10.14	Results of the <code>parabolic_profile</code> test case	153
10.15	Boundary layer profile	153
12.1	Results of the <code>gas_filling</code> test case	162
13.1	Results of the <code>propane_flame_2reac</code> test cases	178
13.2	Results of the <code>multiple_reac_n_simple</code> test case.	178
13.3	The <code>extinction</code> test cases	182
13.4	Results of the <code>pvc_combustion</code> test case	187
13.5	HRR for <code>energy_budget_adiabatic_two_fuels</code> test case	190
13.6	Wall soot deposition for the <code>propane_flame_deposition</code> test case	200
14.1	Results of the <code>ramp_chi_r</code> test case	209
15.1	Results of the <code>spray_burner</code> test case	216
15.2	Droplet size distributions	217
15.3	Results of the <code>hot_rods</code> test case	222
15.4	Example of specified gas mass production from particles	224
15.5	Results of the <code>particle_flux</code> test case	229
15.6	Results of the <code>bucket_test_3</code> case	232
15.7	Results of the <code>part_path_ramp_jog</code> case	233
15.8	Results of the <code>part_orientation_ramp</code> case	234
15.9	Results of the <code>cascade</code> test case	236
15.10	Results of the <code>e_coefficient</code> test case	237
16.1	Results of the <code>wind_example_5</code> and <code>wind_example_10</code> test cases	241
16.2	Sample vertical wind and temperature profiles	244
16.3	Results of the <code>wind_example_32</code> test case	245
16.4	Coriolis effect	247
16.5	The <code>atmospheric_boundary_layer</code> test cases	249

16.6	Results of the <code>stack_effect</code> test case	252
17.1	Material parameters describing vegetation	259
17.2	The <code>char_oxidation_1</code> test case	261
17.3	The <code>char_oxidation_2</code> test case	262
17.4	Example of burning vegetation	264
17.5	Mass generation of firebrands in the <code>dragon_5a</code> test case	267
17.6	Results of the <code>ground_vegetation_load</code> test case	270
17.7	Results of the <code>ground_vegetation_drag</code> test case	271
17.8	Results of the <code>ground_vegetation_radi</code> test case	271
17.9	Results of the <code>ground_vegetation_conv</code> test case	272
17.10	Bova et al. level set test cases	277
17.11	No wind level set test case	278
17.12	Satellite and land use images for sample wildland fire spread case	280
17.13	Results of sample wildland fire spread simulations	281
18.1	Sketch of sprinkler spray	286
18.2	Spray pattern parameters	288
18.3	Results of the <code>bucket_test_2</code> case	289
18.4	Results of the <code>flow_rate</code> test case	291
18.5	Results of the <code>beam_detector</code> test case	295
18.6	Results of the <code>aspiration_detector</code> test case	297
18.7	Results of the <code>control_test_2</code> case	307
18.8	Snapshots of the <code>activate_vents</code> test case	308
18.9	Sample sine function time ramp	309
18.10	Example of freezing the output of a <code>RAMP</code>	310
18.11	HRR curve for a spreading fire	311
18.12	Results of the <code>external_test</code> test case	319
21.1	Results of the <code>duct_flow</code> test case	339
21.2	Results of the <code>dancing_eddies</code> test cases	340
21.3	Pressure iterations in the <code>dancing_eddies</code> test cases	341
21.4	Pressure iterations in the <code>random_obstructions</code>	341
21.5	Snapshot of the <code>helium_2d_isothermal</code> test case	343
21.6	Convergence test for the <code>tunnel_demo</code> test cases	345
21.7	The <code>tunnel_pressure_drop</code> cases	346
21.8	Results of the <code>stairwell</code> test case	347
21.9	Smokeview rendering of a stairwell	348
22.1	Results of the <code>bucket_test_1</code> case	368
22.2	Results of the <code>bucket_test_4</code> case	368
22.3	Results of the <code>test_hrr_2d_cyl</code> test case	372
22.4	Results of the <code>hallways</code> test case	373
22.5	Results of the <code>thermocouple_time_constant</code> test case	377
22.6	Results of the <code>mass_flux_comparison</code> test case	379
22.7	Results of the <code>adiabatic_surface_temperature</code> test case	384
22.8	Format of <code>RADF</code> output file	385
22.9	Examples of the measure of turbulence resolution	395
22.10	Haar mother wavelet	396

22.11 Haar wavelet transforms on four typical signals 397

List of Tables

5.1	Namelist Group Reference Table	32
7.1	A sample of color definitions	68
7.2	OBST SHAPE parameters	72
8.1	Coefficients used for forced convection heat transfer correlations	79
11.1	Parameters used to control time-dependence	157
13.1	Default Critical Flame Temperatures for common fuels	181
14.1	Default radiative fraction for some common fuels	204
15.1	Drag laws available in FDS	221
16.1	Suggested values of Obukhov length	242
16.2	Davenport-Wieringa roughness length classification	242
17.1	Default vegetation kinetic constants	258
17.2	Rothermel wildland fuel models	275
18.1	Suggested values for smoke detector model	292
18.2	Control function types	301
18.3	Single frame static objects	312
18.4	Dual frame static objects	313
18.4	Dual frame static objects (continued)	314
18.5	Dynamic Smokeview objects	314
18.5	Dynamic Smokeview objects (continued)	315
18.5	Dynamic Smokeview objects (continued)	316
18.6	Dynamic Smokeview objects for Lagrangian particles	316
18.6	Dynamic Smokeview objects for Lagrangian particles (continued)	317
19.1	Parameters effected by SIMULATION_MODE	321
19.2	Turbulence model options	322
19.3	Flux limiter options	326
21.1	Summary of available pressure solvers	338
21.2	Friction factors in tunnels	346
22.1	Parameters that control the frequency of output	351
22.2	Output quantities available for PDPA	386
22.3	Coefficients used for the computation of irritant effects of gases	390

22.4	Gas phase output quantities	403
22.5	Solid phase output quantities	406
22.6	Output quantities for devices, controls, and miscellaneous functions	408
22.7	Particle and droplet output quantities	409
22.8	HVAC output quantities	411
23.1	Background species (BACK namelist group)	413
23.2	Boundary file parameters (BNDF namelist group)	414
23.3	Concatenate Input Files parameters (CATF namelist group)	414
23.4	Clipping parameters (CLIP namelist group)	414
23.5	General combustion parameters (COMB namelist group)	415
23.6	Comma separated velocity files (CSVF namelist group)	415
23.7	Control function parameters (CTRL namelist group)	416
23.8	Device parameters (DEVC namelist group)	416
23.9	Output control parameters (DUMP namelist group)	418
23.10	Unstructured geometry parameters (GEOM namelist group)	420
23.11	Header parameters (HEAD namelist group)	421
23.12	Obstruction cutout parameters (HOLE namelist group)	421
23.13	HVAC parameters (HVAC namelist group)	422
23.14	Initial conditions (INIT namelist group)	423
23.15	Isosurface parameters (ISOF namelist group)	425
23.16	Material properties (MATL namelist group)	425
23.17	Mesh parameters (MESH namelist group)	426
23.18	Miscellaneous parameters (MISC namelist group)	427
23.19	Coordinate transformation parameters (MULT namelist group)	429
23.20	Multiplier function parameters (MULT namelist group)	430
23.21	Obstruction parameters (OBST namelist group)	430
23.22	Lagrangian particles (PART namelist group)	432
23.23	Pressure solver parameters (PRES namelist group)	433
23.24	Wall profile parameters (PROF namelist group)	434
23.25	Device properties (PROP namelist group)	434
23.26	Radiation output file parameters (RADF namelist group)	436
23.27	Radiation parameters (RADI namelist group)	436
23.28	Ramp function parameters (RAMP namelist group)	437
23.29	Reaction parameters (REAC namelist group)	438
23.30	Slice file parameters (SLCF namelist group)	439
23.31	Smoke3D parameters (SM3D namelist group)	440
23.32	Species parameters (SPEC namelist group)	440
23.33	Surface properties (SURF namelist group)	442
23.34	Table parameters (TABL namelist group)	446
23.35	Time parameters (TIME namelist group)	446
23.36	MESH transformation parameters (TRN* namelist groups)	447
23.37	Vent parameters (VENT namelist group)	448
23.38	Wind and atmospheric parameters (WIND namelist group)	449
23.39	Pressure zone parameters (ZONE namelist group)	450
26.1	FDS source code files	470

27.1	File Extension Reference Table	483
A.1	Common Pre-defined gas and liquid species	494
A.2	All Pre-defined gas and liquid species	495

Part I

The Basics of FDS

Chapter 1

Introduction

The software described in this document, Fire Dynamics Simulator (FDS), is a computational fluid dynamics (CFD) model of fire-driven fluid flow. FDS solves numerically a form of the Navier-Stokes equations appropriate for low-speed ($Ma^1 < 0.3$), thermally-driven flow with an emphasis on smoke and heat transport from fires. The formulation of the equations and the numerical algorithm are contained in the FDS Technical Reference Guide [3]. Verification and Validation of the model are discussed in the FDS Verification [4] and Validation [5] Guides.

Smokeyview is a separate visualization program that is used to display the results of an FDS simulation. A detailed description of Smokeyview is found in a separate user's guide [2].

1.1 Features of FDS

The first version of FDS was publicly released in February 2000. To date, about half of the applications of the model have been for design of smoke handling systems and sprinkler/detector activation studies. The other half consist of residential and industrial fire reconstructions. Throughout its development, FDS has been aimed at solving practical fire problems in fire protection engineering, while at the same time providing a tool to study fundamental fire dynamics and combustion.

Hydrodynamic Model FDS solves numerically a form of the Navier-Stokes equations appropriate for low-speed, thermally-driven flow with an emphasis on smoke and heat transport from fires. The core algorithm is an explicit predictor-corrector scheme, second order accurate in space and time. Turbulence is treated by means of Large Eddy Simulation (LES). It is possible to perform a Direct Numerical Simulation (DNS) if the underlying numerical mesh is fine enough. See Sec. 19.1 for further details.

Combustion Model For most applications, FDS uses a single step, mixing-controlled chemical reaction which uses three lumped species (a species representing a group of species). These lumped species are air, fuel, and products. By default the last two lumped species are explicitly computed. Options are available to include multiple reactions and reactions that are not necessarily mixing-controlled.

Radiation Transport Radiative heat transfer is included in the model via the solution of the radiation transport equation for a gray gas, and in some limited cases using a wide band model. The equation is solved using a technique similar to finite volume methods for convective transport, thus the name given to it is the Finite Volume Method (FVM). Using approximately 100 discrete angles, the finite volume solver requires about 20 % of the total CPU time of a calculation, a modest cost given the complexity of radiation heat transfer. The absorption coefficients of the gas-soot mixtures are computed using the RadCal

¹The Mach Number, Ma , is the ratio of the flow speed over the speed of sound.

narrow-band model [6]. Liquid droplets can absorb and scatter thermal radiation. This is important in cases involving mist sprinklers, but also plays a role in all sprinkler cases. The absorption and scattering coefficients are based on Mie theory.

Geometry FDS approximates the governing equations on a rectilinear mesh. Rectangular obstructions are forced to conform with the underlying mesh.

Multiple Meshes This is a term used to describe the use of more than one rectangular mesh in a calculation. It is possible to prescribe more than one rectangular mesh to handle cases where the computational domain is not easily embedded within a single mesh.

Parallel Processing FDS employs OpenMP [7], a programming interface that exploits multiple processing units on a single computer. For clusters of computers, FDS employs Message Passing Interface (MPI) [8]. Details can be found in Sec. 3.1.2.

Boundary Conditions All solid surfaces are assigned thermal boundary conditions, plus information about the burning behavior of the material. Heat and mass transfer to and from solid surfaces is usually handled with empirical correlations, although it is possible to compute directly the heat and mass transfer when performing a Direct Numerical Simulation (DNS).

1.2 What's New in FDS 6?

Many of the changes in FDS 6 are improvements to the various sub-models that do not affect the basic structure or parameters of the input file. Most of the changes listed below do not require additional input parameters beyond those used in FDS 5.

Hydrodynamics and Turbulence

- Conservative, total variation diminishing (TVD) scalar transport is implemented: Superbee (VLES default) and CHARM (LES and DNS default). These schemes prevent over-shoots and under-shoots in species concentrations and temperature.
- Improved models for the turbulent viscosity are implemented: Deardorff (default), Dynamic Smagorinsky, and Vreman. These models provide more dynamic range to the flow field for coarse resolution and converge to the correct solution at fine resolution.
- The conservative form of the sensible enthalpy equation is satisfied by construction in the FDS 6 formulation, eliminating temperature anomalies and energy conservation errors due to numerical mixing.
- The baroclinic torque is included by default.
- Improvements are made to the wall functions for momentum and heat flux. An optional wall heat flux model accounts for variable Prandtl number fluids.
- Jarrin's Synthetic Eddy Method (SEM) is implemented for turbulent boundary conditions at vents.

Species and Combustion

- Custom species mixtures ("lumped species") can be defined with the input group SPEC.

- Turbulent combustion is handled with a new partially-stirred batch reactor model. At the subgrid level, species exist in one of two states: unmixed or mixed. The degree of mixing evolves over the FDS time step by the interaction by exchange with the mean (IEM) mixing model. Chemical kinetics may be considered infinitely fast or obey an Arrhenius rate law.
- It is now possible to transport, produce, and consume product species such as CO and soot. Chemical mechanisms must be provided by the user and may include reversible reactions.
- It is now possible to deposit aerosol species onto surfaces.
- There are an increased number of predefined species that now include liquid properties.

Lagrangian Particles

- The functionality of Lagrangian particles has expanded to include the same heat transfer and pyrolysis models that apply to solid walls. In other words, you can now assign a set of surface properties to planar, cylindrical, or spherical particles much like you would for a solid surface.
- More alternatives and user-defined option are available for the liquid droplet size distribution.
- You can specify the radiative properties of the liquid droplets.
- Drag effects of thin porous media (i.e., window screens) can be simulated using planes of particles.

Solid Phase Heat Transfer and Pyrolysis

- The basic 1-D heat transfer and pyrolysis model for solid surfaces remains the same, but there has been a change in several of the input parameters to expand functionality and readability of the input file.
- The pyrolysis model allows for the surface to shrink or swell, based on the specified material densities.

HVAC

- Filters, louvered vents, and heating/cooling capability has been added for HVAC systems.
- HVAC is now functional with MPI.

Radiation

- RadCal database has been extended to include additional fuel species.
- In cells with heat release, the emission term is based on a corrected σT^4 such that when this term is integrated over the flame volume the specified radiative fraction (default 0.35) is recovered. This differs from FDS 5 and earlier where the radiative fraction times the heat release rate was applied locally as the emission term.

Multi-Mesh Computations

- By default, FDS now iterates pressure and velocity at mesh and solid boundaries. You can control the error tolerance and maximum number of iterations via parameters on the `PRES` line.

Control Functions

- CTRL functions have been extended to include math operations.
- The evaluation of RAMPS and DEVCS can be stopped, freezing their value, based upon the activation of a device or control function.

Devices and Output

- Multiple pipe networks can be specified for sprinklers for reduction of flow rate based on the number of operating heads.
- The numerical value of a control function can be output with a DEVC.
- A line of devices can be specified using a number of POINTS on one DEVC line.
- Statistical outputs for RMS, covariance, and correlation coefficient are available.

Chapter 2

Getting Started

FDS is a computer program that solves equations that describe the evolution of fire. It is a Fortran program that reads input parameters from a text file, computes a numerical solution to the governing equations, and writes user-specified output data to files. Smokeview is a companion program that reads FDS output files and produces animations on the computer screen. Smokeview has a simple menu-driven interface. FDS does not. However, there are various third-party programs that have been developed to generate the text file containing the input parameters needed by FDS.

This guide describes how to obtain FDS and Smokeview and how to use FDS. A separate document [2] describes how to use Smokeview.

2.1 How to Acquire FDS and Smokeview

The project homepage provides detailed instructions on how to download executables, manuals, source-code and related utilities.

<https://pages.nist.gov/fds-smv/>

The typical FDS/Smokeview distribution consists of an installation package or compressed archive, which is available for MS Windows, macOS, and Linux.

Old versions can be kept by copying the old version's installation directly to another location so that it is not overwritten when installing a new version.

2.2 Computer Hardware Requirements

The only hard requirement to run the compiled versions of FDS and Smokeview is a 64 bit Windows, Linux, or macOS operating system. The single computer or compute cluster ought to have fast processors (CPUs), and at least 2 to 4 GB RAM per core. The CPU speed will determine how long the computation will take to finish, while the amount of RAM will determine how many mesh cells can be held in memory. A large hard drive is required to store the output of the calculations since the FDS output for a single calculation may consume more than 10 GB of storage space.

Most computers purchased within the past few years are adequate for running Smokeview with the caveat that additional memory (RAM) should be purchased to bring the memory size up to at least 2 GB. This is so the computer can display results without “swapping” to disk. For Smokeview it is also important to obtain a fast graphics card for the PC used to display the results of the FDS computations.

Running FDS using MPI requires shared disk access to each computer on which cases will be run. On Windows systems this involves a domain network with the ability to share folders. On a Linux or macOS system this involves NFS cross mounted files systems with ssh keys setup for password-less login. For Multi-Mesh calculations, the FDS can operate over standard 100 Mb/s networks. A gigabit (1000 Mb/s) network will further reduce network communication times improving data transfer rates between instances of FDS running the parallel cases.

2.3 Computer Operating System (OS) and Software Requirements

The goal of making FDS and Smokeview publicly available has been to enable practicing engineers to perform fairly sophisticated simulations at a reasonable cost. Thus, FDS and Smokeview have been designed for computers running Microsoft Windows, macOS, and Linux.

MS Windows An installation package is available for the 64 bit Windows operating system. Running FDS/Smokeview any version of MS Windows released prior to Windows 7 is not recommended.

macOS Pre-compiled executables are installed into a user selected directory using an installation script. macOS 10.4.x or later is recommended. You can always download the latest version of FDS source and compile FDS for other versions of macOS (see Chapter 26 for details). Note that Rosetta allows running FDS compiled on x86-64 architecture on Apple silicon systems; therefore the FDS executable distributed with the macOS bundle will run on the newer Macs. However, if compiling the source natively on Apple silicon with GNU and Open MPI, it is not currently possible to use the ULMAT pressure solver (which requires Intel MKL libraries), instead ULMAT HYPRE may be used (see Sec. 21.1.1).

Linux Pre-compiled executables are installed into a user selected directory using an installation script. If the pre-compiled FDS executable does not work (usually because of library incompatibilities), the FDS Fortran source code can be downloaded and compiled (See Chapter 26 for details). If Smokeview does not work on the Linux workstation, you can use the Windows version to view FDS output.

2.4 Installation Testing

If you are running FDS under a quality assurance plan that requires installation testing, a test procedure is provided in Appendix B of the FDS Verification Guide [4]. This guide can be obtained from the FDS-SMV website.

Chapter 3

Running FDS

Each FDS simulation is controlled by a single text-based input file, typically given a name that helps identify the particular case, and ending with the file extension `.fds`. This input file can be written directly with a text editor or with the help of a third-party graphical user interface (GUI). The simulation is started directly via the command prompt or through the GUI. The creation of an input file is covered in detail in Part II. This chapter describes how the simulation is run once the input file is written.

If you are new to FDS and Smokeview, it is strongly suggested that you start with an existing input file, run it as is, and then make the appropriate changes to the file for your desired scenario. By running a sample case, you become familiar with the procedure, learn how to use Smokeview, and ensure that your computer is up to the task before embarking on learning how to create new input files.

Sample input files are included as part of the standard installation. A good case for a first time user is located in the sub-folder called `Fires` within the folder called `Examples`. Find the file called `simple_test.fds` and copy it to a folder on your computer that is not within the installation folder. The reason for doing this is to avoid cluttering up the installation folder with a lot of output files. Follow the instructions in Sec. 3.1.2 to run this simple single mesh case. The simulation should only take a few minutes. Once the simulation is completed, use Smokeview to examine the output. In this way, you will quickly learn the basics of running and analyzing simulations.

3.1 Computer Basics

3.1.1 A Brief Primer on Computer Hardware

FDS simulations can exploit multiple processing units on a single computer or multiple computers on a network. Before running an FDS simulation, you should familiarize yourself with your computer hardware.

When using a computer running Microsoft Windows, open up the Task Manager, Performance tab, and look for the number of *sockets*, *cores*, and *logical processors*¹. The socket refers to the physical connector on the motherboard that has a power supply and a connection to random access memory (RAM). This is usually referred to as the central processing unit or CPU. Some motherboards have multiple sockets that can in turn support multiple CPUs, but for typical Windows desktops or laptops, there is one socket/CPU. Each CPU, however, typically has multiple cores, and each core is essentially an independent processing unit that shares access to power and memory. Sometimes cores are referred to as *physical cores* to distinguish them from *logical cores* or *logical processors*. A logical processor is one of multiple *threads* that can be supported

¹The terms sockets, cores, and logical processors are used by the Windows 10 Task Manager on a computer using an Intel processor. These terms might vary with different versions of Windows and different processors.

by a core. For the purpose of running FDS on a Windows computer, the number of logical processors is the most important consideration.

If you are running FDS under any variety of Linux or macOS, you can determine the number of logical processors using the command “lscpu” for Linux or “sysctl hw” for OS X. These operating systems might use slightly different terms, but the processors are similar if not the same as those on a Windows computer.

3.1.2 Two Ways to Use Multiple Processors

FDS can be run on a single computer, using one or more cores, or it can be run on multiple computers. Starting with FDS version 6.2.0, for each supported operating system (Windows, Linux, macOS) there is a single² executable file called `fds` (with an `.exe` file extension on Windows).

There are two ways that FDS can be run in parallel; that is, exploit multiple cores on a single computer or multiple processors/cores distributed over multiple computers on a network or compute cluster. The first way is OpenMP (Open Multi-Processing) [7] which allows a single computer to run a single or multiple mesh FDS simulation on multiple cores. The use of OpenMP does not require the computational domain to be broken up into multiple meshes, and it will still work with cases that have multiple meshes defined. The second way to run FDS in parallel is by way of MPI (Message Passing Interface). Here, the computational domain must be divided into multiple meshes and typically each mesh is assigned its own *process*. These processes can be limited to a single computer, or they can be distributed over a network.

What is OpenMP?

If your simulation involves only a single mesh, you can only run it on one computer, but you can exploit its multiple processors or cores using OpenMP. When you install FDS, it will query your computer to determine the number of available cores. By default, FDS will use approximately half of the available cores³ on a single computer. This is done for two reasons: (1) to prevent taking over your entire machine when running a simulation, and (2) because using all cores for a single simulation may not minimize the run time. OpenMP works best when exploiting multiple (logical) cores associated with a single (physical) processor or “socket”. For example, if your computer has two processors, each with 4 cores, using all 8 cores in an OpenMP simulation may not be worthwhile. You need to experiment with your own machine to determine the strategy that is best for you. To change the number of cores that are available for a given FDS simulation, you can set an environment variable called `OMP_NUM_THREADS`. The way to do this depends on the operating system and will be explained below.

When the job is started, FDS will print the number of cores being used for that job. Note that this setting only applies until you log out of or restart your machine. To set the default value of available cores upon startup, the `OMP_NUM_THREADS` environment variable can also be set in the startup configuration scripts on the machine. Refer to the documentation for the machine’s operating system for more information on how to configure environment variables upon startup.

What is MPI?

MPI (Message-Passing Interface) [8] enables multiple computers, or multiple cores on one computer, to run a multi-mesh FDS job. The main idea is to break up the FDS domain into multiple meshes, and then the flow field in each mesh is computed as an MPI *process*. The *process* can be thought of as a “task” that you would see in the Windows Task Manager or by executing the “top” command on a Linux/Unix machine.

²Previous releases of FDS contained two executables, one that ran on a single processor and one that ran on multiple processors. Starting with FDS 6.2.0, these two executables have been combined into one, and it can run either in serial or parallel mode.

³To determine the number of cores used by OpenMP, just type `fds` at the command prompt.

MPI handles the transfer of information between the meshes, i.e. MPI processes. Usually, each mesh is assigned its own *process* in an MPI calculation, although it is also possible to assign multiple meshes to a single MPI *process*. In this way, large meshes can be computed on dedicated cores, while smaller meshes can be clustered together in a single *process* running on a single core, without the need for MPI message passing.

Also note that FDS refers to its meshes by the numbers 1, 2, 3, and so on, whereas MPI refers to its processes by the numbers 0, 1, 2, and so on. Thus, Mesh 1 is assigned to Process 0; Mesh 2 to Process 1, and so on. You do not explicitly number the meshes or the processes yourself, but error statements from FDS or from MPI might refer to the meshes or processes by number. As an example, if a FDS case with five meshes, the first printout (usually to the screen unless otherwise directed) is:

```
Mesh 1 is assigned to MPI Process 0
Mesh 2 is assigned to MPI Process 1
Mesh 3 is assigned to MPI Process 2
Mesh 4 is assigned to MPI Process 3
Mesh 5 is assigned to MPI Process 4
```

This means that 5 MPI processes (numbered 0 to 4) have started and that each mesh is being handled by its own process. The processes may be on the same or different computers. Each computer has its own memory (RAM), but each individual MPI process has its own independent memory, even if the processes are on the same computer.

There are different implementations of MPI, much like there are different Fortran and C compilers. Each implementation is essentially a library of subroutines called from FDS that transfer data from one process to another across a fast network. The format of the subroutine calls has been widely accepted in the community, allowing different vendors and organizations the freedom to develop better software while working within an open framework. For macOS, we use Open MPI, an open source implementation that is developed and maintained by a consortium of academic, research, and industry partners (www.open-mpi.org). For Windows and Linux, we use Intel MPI.

MPI and OpenMP can be used together. For example, 4 MPI processes can be assigned to 4 different computers, and each MPI process can be supported by, say, 8 OpenMP threads, assuming each computer has 8 cores. Most of the speed up is achieved by the MPI. For a reasonably fast network, you can expect 4 MPI processes to speed up the computation time by a factor of about 0.9 times 4. The OpenMP can provide an extra factor up to about 2, regardless of the number of cores used beyond about 4.

3.2 Launching an FDS Job

To start an FDS simulation, you can either use a third-party graphical user interface (GUI) or you can invoke the computer's command prompt and type a one-line command, as described in the following sections.

3.2.1 Single Computer Running MS Windows

Open up the special FDS command prompt, CMDfds, which should appear on your desktop when you install FDS. By opening this special command prompt, a script is run automatically ensuring that the FDS commands and libraries are all consistent. Change directories ("cd") to where the input file for the case is located. Decide how many logical processes you want to devote to the simulation. Suppose you have 8 logical processors (cores) available, and you have an FDS job that uses 4 meshes, type the following at the command prompt:

```
fds_local -p 4 -o 2 job_name.fds
```

This job will exploit all $4 \times 2 = 8$ logical processors, which you can confirm by opening the Task Manager. The `-p` parameter indicates the number of MPI processes, and the `-o` indicates the number of OpenMP threads.

The progress of the simulation is indicated by diagnostic output that is written onto the screen. Detailed diagnostic information is automatically written to a file `job_name.out`. Screen output can be redirected to a file via the alternative command:

```
fds_local ... job_name.fds > job_name.err
```

It is recommend that the simulation be run in a folder associated with a drive that is physically attached to the computer. Saving files to a network share or a folder controlled by a cloud service (e.g. OneDrive) can result in simulation failures if a network issue or a file lock during automated backup results in Windows preventing FDS from writing data to a file.

3.2.2 Multiple Computers Running MS Windows

The following procedure is intended for a Windows domain network; that is, a network where user accounts are centrally managed such that any user can log in to any machine using the same credentials.

1. The first time you run a job, you must provide your domain name and password by issuing this command:

```
mpiexec -register
```

2. Create a text file, say `hosts.txt`, and in it list, line by line, the names of the computers:

```
fred:3  
wilma:2  
dino:3
```

where the number following the name is the number of MPI processes that you want to invoke on that computer. The sum of all the numbers should be the number of MPI processes for the job, usually equal to the number of meshes.

3. Test your network by running the following test program:

```
mpiexec -machine hosts.txt test_mpi
```

If this command returns a “Hello World” message from each of the computers you listed in the `hosts.txt` file, proceed to the next step. If this command fails, check the network connection between the computers by using each computer to “ping” the other computers. Also, make sure that the same version of FDS is installed on all computers.

4. Share (with both read and write privilege) a working directory on your machine. Do not put this directory within the “Program Files” folder because it is write-protected. Share the working directory with everybody so that all other computers can see it. Note how this directory is defined on the other computers. Sometimes it is `\\<my_computer>\<my_shared_directory>` and sometimes it is defined via the numerical IP address, like `\\129.6.129.87\<my_shared_directory>`. The definition depends on the way your domain name server (DNS) works. Blank spaces in directory names or filenames should be avoided as they can cause problems unless you are a DOS/Windows expert.
5. Within the command prompt, `cd` to the working directory. At the command prompt, type:

```
mpiexec -wdir <working directory> -machinefile hosts.txt fds job_name.fds
```

where `<working directory>` is the full path name of the directory from which the command was invoked.

If you want to use more than one OpenMP thread per MPI process, add the argument

```
mpiexec ... -env OMP_NUM_THREADS <threads> fds_openmp job_name.fds
```

If you accidentally set too many OpenMP threads, you can overload your logical processors and reduce the job efficiency.

6. If successful, you should see the usual FDS printout indicating which MPI processes are assigned to which computer. If not successful, check with your network administrator or monitor the FDS help forums for advice.

3.2.3 macOS

To run a single core FDS job on macOS, `cd` to the working directory and simply type:

```
fds job_name.fds
```

Mac systems tend to be single machines, but may have several cores to utilize. It may be helpful to use the OpenMP version of the code that is provided in the distribution. To run FDS on macOS with OpenMP do the following:

```
export OMP_NUM_THREADS=4 # may be set in ~/.bash_profile as well
fds_openmp job_name.fds
```

Even if you have only one machine, you may use the cores for MPI (distributed) parallel computing. You can map one or more meshes to a given MPI process or rank. On macOS, without OpenMP (one core per MPI process), run a four process job like this:

```
mpiexec -n 4 fds job_name.fds
```

To utilize MPI and OpenMP together (4 MPI ranks with 4 OpenMP threads each, requiring 16 total cores) on macOS do:

```
export OMP_NUM_THREADS=4
mpiexec -n 4 fds_openmp job_name.fds
```

or

```
mpiexec -n 4 -x OMP_NUM_THREADS=4 fds_openmp job_name.fds
```

3.2.4 Linux

A compute cluster that consists of a rack of dedicated compute nodes usually runs one of several variants of the Linux operating system. In such an environment, it is suggested, or required, that you use a job scheduler like PBS/Torque or Slurm to submit jobs by writing a short script that includes the command that launches the job, the amount of resources you require, and so on. Tips for running FDS under Linux or macOS can be found [here](#).

If you opt to run the job without using a job scheduler, you can issue the commands directly at the command prompt. It is best to do this only when running short, small jobs, for example when testing a new computer or a new installation. Do not run large, time-consuming jobs this way because your jobs can potentially interfere with other scheduled jobs. Here is an example of how to run a job that uses four meshes where two MPI processes are assigned to node001 and two are assigned to node002:

```
mpiexec -n 4 -host node001,node002 /home/username/.../fds job_name.fds
```

When the job starts, you should see print out to the screen that looks like this:

```
Starting FDS ...

MPI Process      0 started on node001
MPI Process      1 started on node001
MPI Process      2 started on node002
MPI Process      3 started on node002
...
Number of MPI Processes:  4
```

Note that the pre-compiled packages for either macOS and Linux contain the program `mpiexec`⁴, but they are not exactly the same on each operating system. The Linux installation of FDS makes use of the Intel MPI libraries, whereas macOS uses Open MPI. The command shown above works under Linux. There are many options for `mpiexec` and it is best to experiment with them using a small, multi-mesh job. Check the screen printout, and also, if possible, login to the nodes that you have specified and run the `top` command to see if your processes are running properly.

3.2.5 Using MPI and OpenMP Together

MPI is the better choice when using multiple meshes because it more efficiently divides the computational work than OpenMP. However, combining MPI and OpenMP in the same simulation is possible. If you have multiple computers at your disposal, and each computer has multiple cores, you can assign one MPI process to each computer, and use multiple cores on each computer to speed up the processing of a given mesh using OpenMP. Typically, the use of OpenMP speeds the calculation by at most factor of 2, regardless of how many OpenMP threads you assign to each MPI process. It is usually better to divide the computational domain into more meshes and set the number of OpenMP threads to 1. This all depends on your particular OS, hardware, network traffic, and so on. You should choose a good test case and try different meshing and

⁴There are two very similar programs used to launch MPI jobs—`mpiexec` and `mpirun`. The former is typically used at the command line and the latter is typically used within a job scheduling script.

parallel processing strategies to see what is best for you. The following command runs a 4 mesh FDS job using 4 MPI processes split over two nodes with 4 OpenMP threads attached to each process (Linux):

```
mpiexec -n 4 -host node001,node002 -genv OMP_NUM_THREADS 4 /home/username/.../fds_
openmp job_name.fds
```

When the job starts, you should see print out to the screen that looks like this:

```
Starting FDS ...

MPI Process      0 started on node001
MPI Process      1 started on node001
MPI Process      2 started on node002
MPI Process      3 started on node002
...
Number of MPI Processes:  4
Number of OpenMP Threads: 4
```

Note that the name of the FDS executable file is `fds_openmp` rather than `fds` because there is a separate FDS executable that recognizes OpenMP commands. The reason for this is that the compiler's optimization strategy changes with and without the presence of OpenMP directives. If OpenMP is not considered in the compilation, the optimization is faster. Thus, it can sometimes be of no advantage to add extra OpenMP threads to the MPI processes. Of course, results may be different on different computers with different hardware. It is best to experiment to see what is best for your situation.

3.2.6 Running Very Large Jobs

Most FDS simulations reported in the literature use one to several dozen meshes, and MPI is the method of choice to parallelize these jobs. Usually the meshes are mapped to MPI processes in a one-to-one manner and the meshes contain a comparable number of grid cells. However, it is possible to run FDS jobs that involve thousands of meshes. In 2016, the FDS developers at NIST were given access to the Oak Ridge Leadership Computing Facility at Oak Ridge National Laboratory in Tennessee. The facility provides users access to compute clusters with very large numbers of processors connected via a high speed network. FDS simulations were performed using up to approximately 10,000 MPI processes. If you have access to facilities such as this one, here are a few pointers:

1. Use MPI only. OpenMP will probably not speed up the run time appreciably, and it will consume cores that could be put to better use running more MPI processes.
2. Set `DT_CPU` to some convenient time interval on the `DUMP` line. This parameter directs FDS to periodically write out a file (`CHID_cpu.csv`) that records the wall clock time that each MPI process consumes in the major subroutines. This can help you determine if any of the MPI processes spend an inordinate amount of time idling. Note that the CPU file is written out automatically at the end of the simulation.
3. Run your job for a short amount of time to estimate the time required for the full job. Most large compute clusters will limit you to a certain amount of wall clock time, after which your job is simply stopped. If you have to use the restart feature in FDS, practice first with a short job to make sure that the job can be continued properly.
4. Do a strong scaling study for your particular case. That is, run the job a fixed number of time steps with the least number of meshes that can fit within the machine's memory. Then divide the mesh by factors

of 2, 4, or 8 until reaching a point where the increased number of meshes/processes does not provide a significant speed up.

3.3 Efficiency of Multi-Process Simulations

At the end of a calculation, FDS prints out a file called `CHID_cpu.csv` that records the amount of CPU time that each MPI process spends in the major routines. For example, the column header `VELO` stands for all the subroutines related to computing the flow velocity; `MASS` stands for all the subroutines related to computing the species mass fractions and density. The column header `MAIN` represents all of the CPU time that is not explicitly accounted for; that is, time spend in the main control loop. Ideally, this ought to be a few percent of the overall CPU time usage.

3.3.1 MPI Efficiency

There are two basic approaches to assessing the efficiency or *scalability* of MPI. The first is known as “weak scaling,” in which the amount of work done by each MPI process stays the same and additional processes are added to solve a larger problem. For example, if you are simulating the wind over a patch of terrain, and you keep adding more and more meshes of the same physical and numerical dimension, assigning each new mesh to its own MPI process, so as to simulate a larger and larger patch of terrain, then you would expect that the overall time of the simulation would not increase significantly with each additional mesh. The efficiency of such a calculation is given by the following expression:

$$E_w = \frac{t_1}{t_N} \quad (3.1)$$

where t_1 is the CPU time for the case with 1 mesh (MPI process), and t_N is the CPU time for the case with N meshes (MPI processes). The left hand plot of Fig. 3.1 shows the results of a weak scaling study of FDS. Meshes with dimension 50 by 50 by 50 are lined up side by side, ranging from 1 to 432 meshes. Ideally, the CPU time ought to be about the same for all cases, because each MPI process is doing the same amount of work. Only mesh to mesh communication should lead to inefficiencies. However, notice in the figure that the efficiency of the 1, 2, 4, and 8 mesh cases is greater than those with more MPI processes. The reason for this is that on most compute clusters, each node has multiple cores, and typically jobs run faster when a node is less than completely full. These test cases were run at NIST, where there is a compute cluster with 8 cores per node, and one with 12 cores per node.

The second way to assess MPI efficiency is known as “strong scaling.” Here, you simulate a given scenario on a single mesh, and then you divide the mesh so that the cell size and the overall number of cells does not change. Ideally, if you divide a given mesh into two and run the case with two MPI processes instead of one, you would expect computational time to decrease by a factor of two. However, more MPI processes requires more communication among the processes. Additionally, more meshes results in more mesh boundary cells to compute, even though the overall number of gas phase cells remains the same. The efficiency of such a set of calculations is given by:

$$E_s = \frac{t_1}{N t_N} \quad (3.2)$$

In the strong study demonstrated here, a single mesh of dimension 180 by 160 by 120 is divided into a range of smaller meshes, with the smallest partitioning being 432 meshes of dimension 20 by 20 by 20. The resulting decrease in the CPU time of the entire calculation and the major subroutines is shown in the right hand plot of Fig. 3.1. Ideally, the CPU time should be inversely proportional to the number of meshes (MPI processes); that is, the relative CPU times ought to follow the black dotted lines. The one notable exception to this rule is for “COMM” or COMMunications. This curve represents the time spent in communicating information across the network.

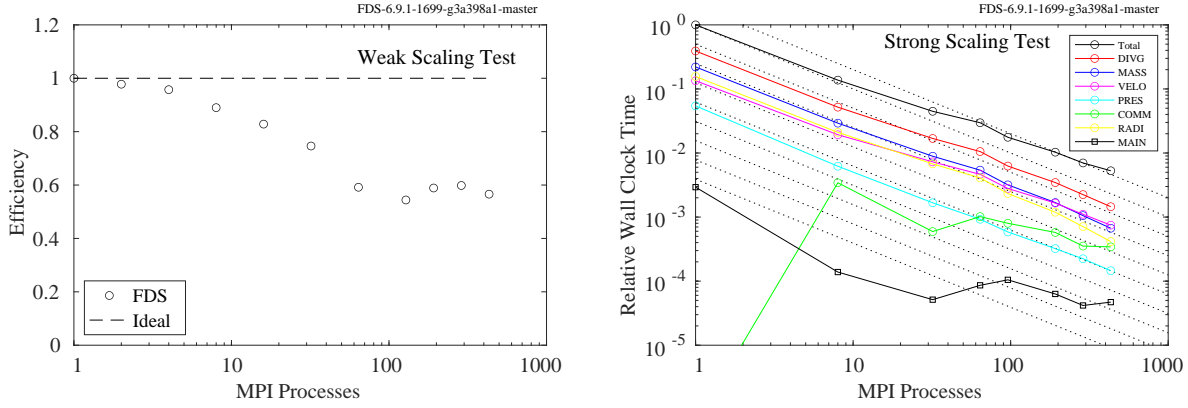


Figure 3.1: Example of a weak (left) and strong (right) scaling study.

3.3.2 OpenMP Efficiency

To confirm the speedup provided by OpenMP, a series of test cases⁵ are run for two mesh sizes (64^3 and 128^3), varying the number of OpenMP threads. The setup is a simple channel flow carrying two extra species to mimic the scalar transport performed in typical fire problems. The results are shown in Fig. 3.2. Generally, users can expect a factor of 2 speedup using 4 cores (default setting).

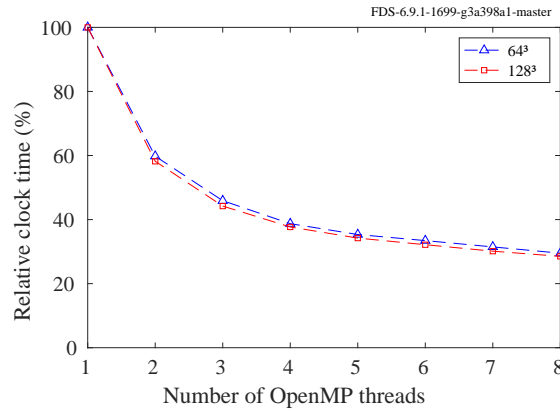


Figure 3.2: Benchmark timing comparison for the OpenMP test cases. The computer that ran these jobs has 2 (physical) sockets, and each socket has 4 (logical) cores. This explains the decrease in efficiency beyond 4 OpenMP threads.

⁵The input files are available in the [FDS GitHub repository](#).

3.4 Monitoring Progress

Diagnostics for a given calculation are written into a file called `CHID.out`. The current simulation time and time step is written here, so you can see how far along the program has progressed. By default, the diagnostics are written out every 100 time-steps after the first 100 time-steps. A different time interval can be specified by the user:

```
&DUMP DIAGNOSTICS_INTERVAL=100 /
```

At any time during a calculation, Smokeview can be run and the progress can be checked visually.

By default, the diagnostics in the `CHID.out` file are verbose. By default, its value is `F` for jobs involving 32 Meshes or less, and `T` for bigger numbers. When running large MPI jobs, quieting this output, which only written by MPI process 0, may be advantageous. To do this, add

```
&DUMP SUPPRESS_DIAGNOSTICS=T /
```

Be aware the output file will not monitor mesh boundary velocity errors in this case; it will echo only the simulation time and time step. You could still output a `BNDF` of `QUANTITY='VELOCITY ERROR'`, if necessary.

To stop a calculation before its scheduled time, create a file in the same directory as the output files called `CHID.stop`. The existence of this file stops the program gracefully, causing it to dump out the latest flow variables for viewing in Smokeview.

Since calculations can extend to hours or days, there is a restart feature in FDS. Details of how to use this feature are given in Sec. 5.6. Briefly, specify at the beginning of calculation how often a “restart” file should be saved. Should something happen to disrupt the calculation, like a power outage, the calculation can be restarted from the time the last restart file was saved.

It is also possible to control the stop time and the time restart files are dumped by using control functions as described in Sec. 18.5.

Chapter 4

User Support

Various problem, some related to FDS and some computer-related, may arise over the course of a project. FDS is a CPU and memory intensive simulation software that can push the computer's processor and memory to its limits. In fact, there are no hardwired bounds within FDS that prevent you from starting a calculation that is too large for your hardware. Even if your machine has adequate memory (RAM), you can still easily set up calculations that can require weeks or months to complete. Predicting at the start of a simulation just how long and how much memory will be required is difficult. Learn how to monitor the resource usage of your computer. Start with small calculations and build your way up.

Although many features in FDS are fairly mature, there are many that are not. FDS is used for practical engineering applications, but also for research in fire and combustion. As you become more familiar with the software, you will inevitably run into areas that are of current research interest. Indeed, burning a roomful of ordinary furniture is one of the most challenging applications of the model. So be patient, and learn to dissect a given scenario into its constitutive parts. For example, do not attempt to simulate a fire spreading through an entire floor of a building unless you have simulated the burning of the various combustibles with relatively small calculations.

Along with the FDS User's Guide, there are resources available on the Internet. These resources include an "Issue Tracker" for reporting bugs and requesting new features, a "Discussion Group" for clarifying questions and discussing more general topics rather than just specific problems, and "Wiki Pages" that provide supplementary information about FDS-SMV development, third-party tools, and other resources. Before using these on-line resources, it is important to first try to solve your own problems by performing simple test calculations or debugging your input file. The next few sections provide a list of error statements and suggestions on how to solve problems.

4.1 The Version Number

When requesting assistance with FDS problems, it is crucial to submit, along with a description of the problem, the FDS version number. Each release of FDS comes with a version number, for example 6.7.2, where the first number is the *major* release, the second is the *minor* release, and the third is the *maintenance* release. Major releases occur every few years, and as the name implies significantly change the functionality of the model. Minor releases occur every few months, and may cause minor changes in functionality. Release notes can help you decide whether the changes should affect the type of applications that you typically do. Maintenance releases are just bug fixes, and should not affect code functionality. To get the version number, just type the executable at the command prompt without an input file, and the relevant information will appear, along with a date of compilation (useful to you) and a so-called Git hash tag (useful to us). The Git hash tag refers to the GitHub repository number of the source code. It allows anyone to

obtain the exact source code files that were used to build that version's executable.

Get in the habit of checking the version number of your executable, periodically checking for new releases which might already have addressed your problem, and telling us what version you are using if you report a problem.

4.2 Common Error Statements

An FDS calculation may end before the specified time limit. Following is a list of common error statements and how to diagnose the problems:

Input File Errors: The most common errors in FDS are due to mistyped input statements. These errors result in the immediate halting of the program and a statement like, "ERROR: Problem with the HEAD line." For these errors, check the line in the input file named in the error statement. Make sure the parameter names are spelled correctly. Make sure that a / (forward slash) is put at the end of each namelist entry. Make sure that the correct type of input data is provided for each input parameter, such as one real number, several integers, or something else. Make sure there are no non-ASCII characters being used, as can sometimes happen when text is cut and pasted from other applications or word-processing software. Make sure that the number zero (0) is not typed as the capital letter O and *vice versa*. Make sure that the number one (1) is not an exclamation point (!) or lower case letter l. Make sure apostrophes are used to designate character strings. Make sure the text file on a Unix/Linux machine was not created on a Windows machine, and *vice versa*. Make sure that all the parameters listed are still being used – new versions of FDS often drop or change parameters forcing you to re-examine old input files.

Numerical Instability Errors: Input errors are numerical errors during a simulation may result in high velocities in the domain that cause the time step size to decrease to a point¹ where logic in the code decides that the results are unphysical and stops the calculation with an error message in the file `CHID.out`. In these cases, FDS ends by dumping out one final Plot3D file giving you a hint as to where the error is occurring within the computational domain. Usually, a numerical instability can be identified by fictitiously large velocity vectors emanating from a small region within the domain. Common causes of such instabilities are:

- mesh cells that have an aspect ratio larger than 2 to 1
- a change in grid resolution of more than a factor of 2 at a mesh interface
- mismatched obstructions or hollow ducts at a mesh interface where the grid resolution changes
- high speed flow through a small opening, in particular openings intended to model leakage. There are better ways to model leakage, as described in Sec. 10.3.2.
- a sudden change in the heat release rate
- the use of `ADIABATIC` as a solid phase boundary condition. This parameter should only be used for testing because no solid material has a thermal conductivity of zero. Using this parameter can cause unphysical fluctuations at the solid-gas interface.
- the removal or creation of an obstruction, like the opening or closing of a door
- a high (>100 g/mol) molecular weight fuel molecule that is not in the FDS database, Appendix A. In such cases, reduce the molecular weight but maintain the atom ratios or define the fuel specific heat on `SPEC`. See Sec. 12.1.3 for details.

¹By default, the calculation is stopped when the time step drops below 0.0001 of the initial time step. This factor can be changed via the `TIME` line by specifying the `LIMITING_DT_RATIO`.

- long, sealed tunnels, in which pressure fluctuations can cause spurious numerical artifacts. See Sec. 21 for details.
- large number of automatically generated pressure zones. This can happen when a preprocessor simply converts CAD geometry stored as faces to thin obstructions. For example, on a 10 cm grid, a typical interior gypsum board wall will appear in an FDS input file as two thin obstructions representing the gypsum board enclosing a single grid cell wide volume of air. FDS will automatically identify these regions as pressure zones; however, they can sometimes result in instabilities developing. The pressure zones identified by FDS can be determined by reviewing the `casename.out` file. Telling FDS to automatically fill these zones (see Section 10.3.4) or manually fixing the geometry to fill the regions can reduce these issues.

There are various ways to solve the problem, depending on the situation. Try to diagnose and fix the problem before reporting it. The originator of the input file is by far the best person to diagnose the problem.

Inadequate Computer Resources: The calculation might be using more RAM than the machine has (you will see an error message like “ERROR: Memory allocation failed for ZZ in the routine INIT”) , or the output files could have used up all the available disk space. In these situations, the computer may or may not produce an intelligible error message. Sometimes the computer is just unresponsive. It is your responsibility to ensure that the computer has adequate resources to do the calculation. Remember, there is no limit to how big or how long FDS calculations can be – it depends on the resources of the computer. For any new simulation, try running the case with a modest-sized mesh, and gradually make refinements until the computer can no longer handle it. Then back off somewhat on the size of the calculation so that the computer can comfortably run the case. Trying to run with 90 % to 100 % of computer resources is risky; using MPI and multiple machines would be better. If you are using a Linux/Unix machine, make sure that the stacksize is unlimited, which will allow FDS to access as much of the RAM as possible. Changing the stacksize limit differs with each shell type, so it is best to do an on-line search to find out how to ensure that the stacksize is unlimited.

Run-Time Errors: An error occurs either within the computer operating system or the FDS program. An error message is printed out by the operating system of the computer onto the screen or into the diagnostic output file. This message is most often unintelligible to most people, including the programmers, although occasionally one might get a small clue if there is mention of a specific problem, like “stack overflow,” “divide by zero,” or “file write error, unit=...” Sometimes the error message simply refers to a “Segmentation Fault.” These errors may be caused by a bug in FDS, for example if a number is divided by zero, or an array is used before it is allocated, or any number of other problems. Before reporting the error to the Issue Tracker, try to systematically simplify the input file until the error goes away. This process usually brings to light some feature of the calculation responsible for the problem and helps in the debugging.

MPI Timeout Errors: If there is an error or delay in communication from one MPI (Message Passing Interface) process to another, you might see an error statement like “ERROR: MPI exchange ... timed out for MPI process ...” This indicates that, for whatever reason, an MPI message never reached its destination and the program shut down. The input parameter `MPI_TIMEOUT`, specified on the `MISC` line, sets the time that FDS waits for the communication to complete. By default, it is 600 s. This is an extremely long period of time to wait for an MPI communication to complete and it typically means that something has gone very wrong. If you believe that there is a reason for the delay, for example one mesh is much bigger in size or has much more work to do than other meshes, then set `MPI_TIMEOUT` to a larger value and hope for the best. However, the timeout might indicate that there is something wrong

with the way the calculation has been set up, or something wrong with your network and the particular *communication fabric* that it uses. If you cannot determine the problem, submit a simple version of the case that fails to the [Issue Tracker](#).

Mesh Connection Error: If the arrangement of meshes is complicated, FDS may not be able to sort out which mesh is connected to which, and typically the error message “Mesh connection test timed out for MPI process ...” appears early in the simulation, before any time-stepping occurs. One possible remedy to this problem, besides simplifying the mesh arrangement, is to ensure that the fine meshes or any embedded meshes are listed first in the input file. That way, FDS has a better chance of registering the proper connections between meshes.

File Writing Errors: Occasionally, especially on Windows machines, FDS fails because it is not permitted to write to a file. A typical error statement reads:

```
forrtl: severe (47): write to READONLY file, unit 8598, file C:\Users\...\
```

The unit, in this case 8598, is just a number that FDS has associated with one of the output files. If this error occurs just after the start of the calculation, you can try adding the phrase

```
FLUSH_FILE_BUFFERS=F
```

on the `DUMP` line of the input file (see Sec. 22.1). This will prevent FDS from attempting to flush the contents of the internal buffers, something it does to make it possible to view the FDS output in Smokeview during the FDS simulation. On some Windows machines, you might encounter security settings that prevent command line programs such as FDS from writing to system folders that contain program files. In this case, try to rerun the case in a non-system folder (i.e., a location within your home directory).

Poisson Initialization: Sometimes at the very start of a calculation, an error appears stating that there is a problem with the “Poisson initialization.” The equation for pressure in FDS is known as the Poisson equation. The Poisson solver consists of large system of linear equations that must be initialized at the start of the calculation. Most often, an error in the initialization step is due to a mesh `IJK` dimension being less than 4 (except in the case of a two-dimensional calculation). It is also possible that something is fundamentally wrong with the coordinates of the computational domain. Diagnose the problem by checking the `MESH` lines in the input file.

4.3 Support Requests and Bug Tracking

Because FDS development is on-going, problems will inevitably occur with various routines and features. The developers need to know if a certain feature is not working, and reporting problems is encouraged. However, the problem must be clearly identified. The best way to do this is to simplify the input file as much as possible so that the bug can be diagnosed (i.e., create and submit a minimal working example). Also, limit the bug reports to those features that clearly do not work. Physical problems such as fires that do not ignite, flames that do not spread, etc., may be related to the mesh resolution or scenario formulation, and you need to investigate the problem first before reporting it. If an error message originates from the operating system as opposed to FDS, first investigate some of the more obvious possibilities, such as memory size, disk space, etc.

If that does not solve the problem, report the problem with as much information about the error message and circumstances related to the problem. The input file should be simplified as much as possible so that the bug occurs early in the calculation. Attach the simplified input file if necessary, following the instructions

provided at the web site. In this way, the developers can quickly run the problematic input file and hopefully diagnose the problem.

Note: Reports of specific bugs, problems, feature requests, and enhancements should be posted to the Issue Tracker and not the Discussion Group.

Part II

Writing an FDS Input File

Chapter 5

The Basic Structure of an Input File

5.1 Naming the Input File

The operation of FDS is based on a single ASCII¹ text file containing parameters organized into *namelist*² groups. The input file provides FDS with all of the necessary information to describe the scenario. The input file is saved with a name such as `job_name.fds`, where `job_name` is any character string that helps to identify the simulation. If this same string is repeated under the `HEAD` namelist group within the input file, then all of the output files associated with the calculation will then have this common prefix name.

There should be no blank spaces in the job name. Instead use the underscore character to represent a space. Using an underscore characters instead of a space also applies to the general practice of naming directories on your system.

Be aware that FDS will simply over-write the output files of a given case if its assigned name is the same. This is convenient when developing an input file because you save on disk space. Just be careful not to overwrite a calculation that you want to keep.

5.2 Namelist Formatting

Parameters are specified within the input file by using *namelist* formatted records. Each namelist record begins with the ampersand character, `&`, followed immediately by the name of the namelist group, then a comma-delimited list of the input parameters, and finally a forward slash, `/`. For example, the line

```
&DUMP NFRAMES=1800, DT_HRR=10., DT_DEVC=10., DT_PROF=30. /
```

sets various values of parameters contained in the `DUMP` namelist group. The meanings of these various parameters will be explained in subsequent chapters. The namelist records can span multiple lines in the input file, but just be sure to end the record with a slash or else the data will not be understood. Do not add anything to a namelist line other than the parameters and values appropriate for that group. Otherwise, FDS will stop immediately upon execution.

Parameters within a namelist record can be separated by either commas, spaces, or line breaks. It is recommended that you use commas or line breaks, and never use tab stops because they are not explicitly defined in the namelist data structure. Comments and notes can be written into the file so long as nothing comes before the ampersand except a space and nothing comes between the ampersand and the slash except appropriate parameters corresponding to that particular namelist group.

¹ ASCII – American Standard Code for Information Interchange. There are 128 characters that make up the standard ASCII text.

² A *namelist* is a Fortran input record.

The parameters in the input file can be integers, reals, character strings, or logical parameters. A logical parameter (outside of Fortran known as boolean parameter) is either `T` (for True) or `F` (for False). The periods following numbers (e.g. `10.`) are a convention for specifying a real as opposed to an integer (`10.` is equivalent to `10.0`). Character strings that are listed in this User's Guide must be copied exactly as written – the code is case sensitive and underscores *do* matter. The maximum length of most character input parameters is 60.

Most of the input parameters are simply real or integer scalars, like `DT=0.02`, but sometimes the inputs are multidimensional arrays. For example, when describing a particular solid surface, you need to express the mass fractions of multiple materials that are to be found in multiple layers. The input array `MATL_MASS_FRACTION(IL, IC)` is intended to convey to FDS the mass fraction of component `IC` of layer `IL`. For example, if the mass fraction of the second material of the third layer is 0.5, then write

```
MATL_MASS_FRACTION(3,2)=0.5
```

To enter more than one mass fraction, use this notation:

```
MATL_MASS_FRACTION(1,1:3)=0.5,0.4,0.1
```

which means that the first three materials of layer 1 have mass fractions of 0.5, 0.4, and 0.1, respectively. The notation `1:3` means array elements 1 through 3, inclusive.

Note that character strings can be enclosed either by single or double quotation marks. Be careful not to create the input file by pasting text from something other than a simple text editor, in which case the punctuation marks may not transfer properly into the text file.

Some text file encodings may not work on all systems. If file reading errors occur and no typographical errors can be found in the input file, try saving the input file using a different encoding. For example, the text file editor Notepad works fine on a Windows PC, but a file edited in Notepad may not work on Linux or macOS because of the difference in line endings between Windows and Unix/Linux operating systems. Other text editors, such as Notepad++, typically work better, but try a simple case first.

5.3 Input File Structure

In general, the namelist records can be entered in any order in the input file, but it is a good idea to organize them in some systematic way. Typically, general information is listed near the top of the input file, and detailed information, like obstructions, devices, and so on, are listed below. FDS scans the entire input file each time it processes a particular namelist group. With some text editors, it has been noticed that the last line of the file is often not read by FDS because of the presence of an “end of file” character. To ensure that FDS reads the entire input file, add

```
&TAIL /
```

as the last line at the end of the input file. This completes the file from `&HEAD` to `&TAIL`. FDS does not even look for this last line. It just forces the “end of file” character past relevant input.

Another general rule of thumb when writing input files is to only add parameters that make a change from the default value. That way, you can more easily distinguish between what you want and what FDS wants. Add comments liberally to the file, so long as these comments do not fall within the namelist records.

The general structure of an input file is shown below, with many lines of the original validation input file³ removed for clarity.

³The actual input file, `WTC_05.fds`, is part of the FDS Validation Suite

```

&HEAD CHID='WTC_05', TITLE='WTC Phase 1, Test 5' /
&MESH IJK=90,36,38, XB=-1.0,8.0,-1.8,1.8,0.0,3.82 /
&TIME T_END=5400. /
&MISC TMPA=20. /
&DUMP NFRAMES=1800, DT_HRR=10., DT_DEVC=10., DT_PROF=30. /

&REAC FUEL      = 'N-HEPTANE'
      FYI      = 'Heptane, C_7 H_16'
      C        = 7.
      H        = 16.
      CO_YIELD = 0.008
      SOOT_YIELD = 0.015 /

&OBST XB= 3.5, 4.5,-1.0, 1.0, 0.0, 0.0, SURF_ID='STEEL FLANGE' /  Fire Pan
...
&SURF ID        = 'STEEL FLANGE'
      COLOR     = 'BLACK'
      MATL_ID   = 'STEEL'
      BACKING   = 'EXPOSED'
      THICKNESS = 0.0063 /
...
&VENT MB='XMIN', SURF_ID='OPEN' /
...
&SLCF PBY=0.0, QUANTITY='TEMPERATURE', VECTOR=T /
...
&BNDF QUANTITY='GAUGE HEAT FLUX' /
...
&DEVC XYZ=6.04,0.28,3.65, QUANTITY='VOLUME FRACTION', SPEC_ID='OXYGEN', ID='EO2_FDS' /
...
&TAIL / End of file.

```

It is recommended that when looking at a new scenario, first select a pre-written input file that resembles the case, make the necessary changes, then run the case at fairly low mesh resolution to determine if the file is set up correctly. It is best to start off with a relatively simple file that captures the main features of the problem without getting tied down with too much detail that might mask a fundamental flaw in the calculation. Initial calculations ought to be meshed coarsely so that the run times are less than an hour and corrections can easily be made without wasting too much time. As you learn how to write input files, you will continually run and re-run your case as you add in complexity.

Table 5.1 provides a quick reference to all the namelist parameters and where you can find the reference to where it is introduced in the document and the table containing all of the keywords for each group.

Table 5.1: Namelist Group Reference Table

Group Name	Namelist Group Description	Reference Section	Parameter Table
BNDF	Boundary File Output	22.5	23.2
CATF	Concatenate Input Files	5.4	23.3
CLIP	Clipping Parameters	19.5	23.4
COMB	Combustion Parameters	13	23.5
CSVF	Initialization File	20.5	23.6
CTRL	Control Function Parameters	18.5	23.7
DEVC	Device Parameters	18.1	23.8
DUMP	Output Parameters	22	23.9
GEOM	Unstructured Geometry	7.3	23.10
HEAD	Input File Header	6.1	23.11
HOLE	Obstruction Cutout	7.2.8	23.12
HVAC	Heating, Vent., Air Cond.	10.2	23.13
INIT	Initial Condition	20	23.14
ISOF	Isosurface File Output	22.6	23.15
MATL	Material Property	8.3	23.16
MESH	Mesh Parameters	6.3	23.17
MISC	Miscellaneous	19	23.18
MOVE	Transformation Parameters	11.4	23.19
MULT	Multiplier Parameters	7.6	23.20
OBST	Obstruction	7.2	23.21
PART	Lagrangian Particle	15	23.22
PRES	Pressure Solver Parameters	21	23.23
PROF	Profile Output	22.3	23.24
PROP	Device Property	18.3	23.25
RADF	Radiation Output File	22.10.14	23.26
RADI	Radiation	14.1	23.27
RAMP	Ramp Profile	11	23.28
REAC	Reaction Parameters	13	23.29
SLCF	Slice File Output	22.4	23.30
SPEC	Species Parameters	12	23.32
SURF	Surface Properties	7.1	23.33
TABL	Tabulated Particle Data	18.3.1	23.34
TIME	Simulation Time	6.2	23.35
TRNX	Mesh Stretching	6.3.5	23.36
VENT	Vent Parameters	7.4	23.37
WIND	Wind Parameters	16.2	23.38
ZONE	Pressure Zone Parameters	10.3	23.39

5.4 Concatenating Input Files

The namelist group `CATF` allows for the inclusion of input information from different files into a simulation. The input line:

```
&CATF OTHER_FILES='file_1.txt','file_2.txt' /
```

adds the contents of the two listed files into the current input file. The contents of the other files will be inserted at the location of each respective `CATF` line in the input file. Up to 20 files can be listed on one `CATF` line, and multiple `CATF` lines can be included in the input file. After reading the input file, FDS creates a new input file, `CHID_cat.fds`, and then runs the case.

When using this feature, limit the number of characters per line in both the original and added input files to 400.

5.5 Protecting Old Cases

When you run a job that has been run previously, FDS will automatically overwrite the old output with new. If you do not want this to happen, set `OVERWRITE=F` on the `MISC` line, in which case FDS will check for the existence of previously created output files and stop execution if they exist.

5.6 Stopping and Restarting Calculations (`.restart`)

An important `MISC` parameter is called `RESTART`. Normally, a simulation consists of a sequence of events starting from ambient conditions. However, there are occasions when you might want to stop a calculation, make a few limited adjustments, and then restart the calculation from that point in time. To do this, first bring the calculation to a halt gracefully by creating a file called `CHID.stop` in the directory where the output files are located. Remember that FDS is case-sensitive. The file name must be exactly the same as the `CHID` and ‘stop’ should be lower case. If you have a `CATF` line in your input, you will need to use `CHID_cat.stop`. FDS checks for the existence of this file at each time step, and if it finds it, gracefully shuts down the calculation after first creating a file (or files in the case of a multiple mesh job) called `CHID.restart` (or `CHID_nn.restart`). To restart a job, the file(s) `CHID.restart` should exist in the working directory, and the phrase `RESTART=T` needs to be added to the `MISC` line of the input data file. For example, suppose that the job whose `CHID` is “plume” is halted by creating a dummy file called `plume.stop` in the directory where all the output files are being created. To restart this job from where it left off you will need to delete the file `CHID.stop` so that FDS does not immediately stop the calculation and add `RESTART=T` to the `MISC` line of the input file `plume.fds`, or whatever you have chosen to name the input file. The existence of a restart file with the same `CHID` as the original job tells FDS to continue saving the new data in the same files as the old⁴. If `RESTART_CHID` is also specified on the `MISC` line, then FDS will look for old output files tagged with this string instead of using the specified `CHID` on the `HEAD` line. In this case, the new output files will be tagged with `CHID`, and the old output files will not be altered. When running the restarted job, the diagnostic output of the restarted job is appended to output files from the original job.

There may be times when you want to save restart files periodically during a run as insurance against power outages or system crashes. If this is the case, at the start of the original run set `DT_RESTART=50.` on the `DUMP` line to save restart files every 50 s, for example. The default for `DT_RESTART` is 1000000, meaning

⁴By default, when a job is restarted, the spreadsheet output files will be appended at the time the job was restarted, not the time the job was stopped. If you want the output files to be appended without clipping off any existing data, even though some duplicate output will be left over, then set `CLIP_RESTART_FILES` to `F` on the `DUMP` line.

no restart files are created unless you gracefully stop a job by creating a dummy file called `CHID.stop`. The control function feature (see Sec. 18.5) can be used to stop a calculation or dump a restart file when the computation reaches some measurable condition such as the first sprinkler activation.

Between job stops and restarts, major changes cannot be made in the calculation like adding or removing vents and obstructions. The changes are limited to those parameters that do not instantly alter the existing flow field. For example, changing the output frequency of data sent to the heat release rate (`*_hrr.csv`) or device (`*_devc.csv`) files, or even slice file output. The new frequency is set on the `DUMP` line (see Table 23.9 for a complete list of output frequencies). If no frequency is set on the new run, the frequency from the previous run is maintained. However, this is not true of `DT_RESTART`; if no `DT_RESTART` is present for the new run, then no restart files will be written (this prevents overwrites of old restart files).

Since the restart capability has been used infrequently by the developers, it should be considered a fragile construct. Examine the output to ensure that no sudden or unexpected events occur during the stop and restart.

Chapter 6

Setting the Bounds of Time and Space

This chapter describes global input parameters that affect the general scope of the simulation, like the simulation time and the size and extent of the computational domain. Essentially, these parameters establish the spatial and temporal coordinate systems that are used by all other components of the simulation, which is why these parameters are usually listed at the top of the input file and why they are described here first.

6.1 Naming the Job

The first thing to do when setting up an input file is to give the job a name. The name of the job is important because often a project involves numerous simulations in which case the names of the individual simulations should be meaningful and help to organize the project. The namelist group `HEAD` contains two parameters, as in this example:

```
&HEAD CHID='WTC_05', TITLE='WTC Phase 1, Test 5' /
```

`CHID` stands for *Character ID*, it is a string of 50 characters or less used to tag the output files. If, for example, `CHID='WTC_05'`, it is convenient to name the input data file `WTC_05.fds` so that the input file can be associated with the output files. No periods or spaces are allowed in `CHID` because the output files are tagged with suffixes that are meaningful to certain computer operating systems. If `CHID` is not specified, then it will be set to the name of the input file minus everything at and beyond the first period.

`TITLE` is a string of 256 characters or less that describes the simulation. It is simply a descriptive text that is passed to various output files.

6.2 Simulation Time

`TIME` is the name of a group of parameters that define the time duration of the simulation and the initial time step used to advance the solution of the discretized equations.

6.2.1 Basics

Usually, only the duration of the simulation is required on this line, via the parameter `T_END`. The default is 1 s. For example, the following line will instruct FDS to run the simulation for 5400 s.

```
&TIME T_END=5400. /
```

If `T_END` is set to zero, only the set-up work is performed, allowing you to quickly check the geometry in Smokeview.

To start the time line a number other than zero, use the parameter `T_BEGIN` to specify the time written to file for the first time step. This would be useful for matching time lines of experimental data or video recordings.

6.2.2 Controlling the Time Step

The initial time step size can be specified with `DT`. This parameter is normally set automatically by dividing the size of a mesh cell by the characteristic velocity of the flow. During the calculation, the time step is adjusted so that the CFL (Courant, Friedrichs, Lewy) condition is satisfied. The default value of `DT` is $(\delta x \delta y \delta z)^{\frac{1}{3}} / V_{\text{char}}$ s. In this formula, the characteristic velocity scale is estimated to be $V_{\text{char}} = 0.2\sqrt{gH}$ m/s, where δx , δy , and δz are the dimensions of the smallest mesh cell, H is the height of the computational domain, and g is the acceleration of gravity. Physically the characteristic velocity is an estimate of the average velocity of a buoyant plume over the height of the computational domain which provides a reasonable initial guess for the time step in a typical fire simulation. By default, the time step is not allowed to increase above its initial value. To allow this to happen, set `RESTRICT_TIME_STEP=F`.

If something sudden is to happen right at the start of a simulation, like a sprinkler activation, setting the initial time step to avoid a numerical instability caused by too large a time step is beneficial. Experiment with different values of `DT` by monitoring the initial time step sizes recorded in the output file `job_name.out`.

At the end of the first part of the explicit predictor-corrector time update, the time step is checked to ensure that it is within the appropriate stability bounds. If it is not, it is adjusted up or down by 10 % (or until it is within limits) and the predictor part of the time step is re-run. If you want to prevent FDS from automatically changing the time step, set `LOCK_TIME_STEP` equal to `T` on the `TIME` line, in which case the specified time step, `DT`, will not be adjusted. This parameter is intended for diagnostic purposes only, for example, timing program execution. It can lead to numerical instabilities if the initial time step is set too high.

The diagnostic output file called `CHID.out` contains information about the time step and CFL number in each mesh. However, for simulations that involve dozens or hundreds of meshes, assessing the global time step information can be difficult. For detailed information about the time step, there is an optional output file called `CHID_cfl.csv` that contains, for every time step, the time (s), time step size (s), maximum CFL number, the mesh and mesh indices where the maximum CFL value occurs, the six velocity components on the six faces of the grid cell where the maximum CFL number occurs, and the divergence (1/s), viscosity (kg/m/s), `HRRPUV` (kW/m³), and combustion mixing time (s) where the maximum CFL occurs. Also listed is the maximum VN (Von Neumann) number and the mesh and cell indices where it occurs. To output this file, set `CFL_FILE` to `T` on the `DUMP` line. It is normally set to `F`.

The Final Time Step

In some special circumstances controlling the final time step of the calculation may be necessary. By default, FDS sets the last time step to be $T_{END} - T$ where T is the current time. Because of floating point arithmetic, this final time step may be very small and cause unintended consequences. You can control this final time step in two ways, using the parameters `DT_END_MINIMUM` or `DT_END_FILL` (you should not need both). Using `DT_END_MINIMUM` has the effect of setting a minimum for the final time step and so T_{END} is not strictly observed (hence our default `DT_END_MINIMUM` is roughly the machine epsilon). The final simulation time will be $T + DT_END_MINIMUM$. Alternatively, you could set `DT_END_FILL` to some value (should be less than the last time step). Then if $T + DT + DT_END_FILL > T_{END}$, where DT is the computed stable time step (or specified time step), FDS will recompute the time step to be $DT = T_{END} - T$. If you set one of these parameters, double check the `CHID.out` file to make sure the expected behavior has been observed.

Simulation Time Ramp

WARNING: This feature should only be used for verification purposes.

There may be situations where you would like to specify the time step sequence in FDS, most likely to make a comparison with another application. You can add a `TIME_RAMP` to the `TIME` line where the independent variable, T , specifies the *end-of-time-step* value of the *time* (s). The simulation time steps will then take on the value of the increments between the ramp values. The first time step will be the first value of the ramp minus T_BEGIN . For example, the following code snippet specifies the time values for the plot in Fig. 6.1, which simply outputs ambient temperature values at the specified time snapshots.

```
&TIME T_BEGIN=0., T_END=10., RAMP_TIME='time-ramp' /
&RAMP ID='time-ramp', T=1 /
&RAMP ID='time-ramp', T=2 /
&RAMP ID='time-ramp', T=5 /
&RAMP ID='time-ramp', T=7 /
&RAMP ID='time-ramp', T=9 /
```

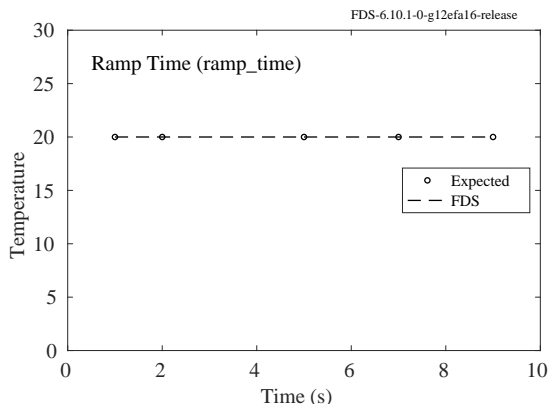


Figure 6.1: An example of ramping the simulation time values.

6.2.3 Steady-State Applications

Occasionally, there are applications in which only the steady-state solution (in a time-averaged sense) is desired. In a time-dependent code such as FDS, reaching steady-state requires that the simulation be run long enough that a steady-state flow field and gas temperature field develops and that steady-state temperature profiles develop in solid surfaces. Developing steady-state temperature profiles can take a long time as the surface must be heated up in-depth. This can make the cost of the calculation prohibitive. Given steady-state boundary conditions, the time it takes a surface to heat up is proportional to its specific heat. Therefore, under steady conditions, this time can be accelerated by reducing the specific heat. So that the `SPECIFIC_HEAT` inputs on `MATL` can remain the same but still provide a means to reduce the heating time of a surface, the `MISC` input of `TIME_SHRINK_FACTOR` can be used. The `SPECIFIC_HEAT` inputs on `MATL` are reduced by this factor. For a `TIME_SHRINK_FACTOR` of 10, the specific heats of the various materials is reduced by a factor of 10. This speeds up the heating of solid materials roughly by a factor of 10. An example of an application where this parameter is handy is a validation experiment where a steady heat source warms up a compartment to a nearly equilibrium state at which point time-averaged flow quantities are measured.

`TIME_SHRINK_FACTOR` does not reduce the time for the flow and gas temperature fields to develop; however, these often develop much faster than the wall temperature profile.

Note that when `TIME_SHRINK_FACTOR` is used a device with `QUANTITY='TIME'` or a device or control function with a `DELAY` will have those values adjusted by the value of `TIME_SHRINK_FACTOR`. For example if a 10 s `DELAY` is specified for a `CTRL` input with a `TIME_SHRINK_FACTOR` of 10, then FDS will adjust the `DELAY` to 1 s. However, use of time based device or control functions that change the simulation implies a non-steady scenario where the use of `TIME_SHRINK_FACTOR` may no longer be appropriate.

6.3 Computational Meshes

All FDS calculations must be performed within a domain that is made up of rectilinear volumes called *meshes*. Each mesh is divided into rectangular *cells*, the number of which depends on the desired resolution of the flow dynamics. `MESH` is the namelist group that defines the computational domain.

6.3.1 Basics

A mesh is a single right parallelepiped, i.e., a box. The coordinate system within a mesh conforms to the right hand rule. The origin point of a mesh is defined by the first, third and fifth values of the real number sextuplet, `XB`, and the opposite corner is defined by the second, fourth and sixth values. For example,

```
&MESH IJK=10,20,30, XB=0.0,1.0,0.0,2.0,0.0,3.0 /
```

defines a mesh that spans the volume starting at the origin and extending 1 m in the positive x direction, 2 m in the positive y direction, and 3 m in the positive z direction. The mesh is subdivided into uniform cells via the parameter `IJK`. In this example, the mesh is divided into 10 cm cubes. It is best if the mesh cells resemble cubes; that is, the length, width and height of the cells ought to be roughly the same. Non-uniform mesh cells in particular direction may be specified using the namelist groups `TRNX`, `TRNY` and/or `TRNZ` (See Sec. 6.3.5).

Any obstructions or vents that extend beyond the boundary of the mesh are cut off at the boundary. There is no penalty for defining objects outside of the mesh, and these objects will not appear in Smokeview.

The pressure solver in FDS employs Fast Fourier Transforms (FFTs) in the y and z directions, and this algorithm works most efficiently if the number of cells in these directions (the J and K of `IJK`) can be factored into low prime numbers, like 2, 3, and 5. The number of cells in the x direction (the I in `IJK`) is not affected by this restriction because the pressure solver does not use an FFT in the x direction. However, since the pressure solver uses less than 10 % of the total CPU time, the gains in using low prime dimensions are usually negligible. Experiment with different mesh dimensions to ensure that those that are ultimately used do not unduly slow down the calculation.

6.3.2 Two-Dimensional and Axially-Symmetric Calculations

The governing equations solved in FDS are written in terms of a three dimensional Cartesian coordinate system. However, a two dimensional Cartesian or two dimensional cylindrical (axially-symmetric) calculation can be performed by setting the J in the `IJK` triplet to 1 on the `MESH` line. For axial symmetry, add `CYLINDRICAL=T` to the `MESH` line, and the coordinate x is then interpreted as the radial coordinate r . If more than one mesh is used, all the meshes must be specified as 2-D or `CYLINDRICAL`—you cannot mix 2-D, 3-D and cylindrical geometries. No boundary conditions should be set at the planes $y = YMIN = XB(3)$ or $y = YMAX = XB(4)$, nor at $r = XMIN = XB(1)$ in an axially-symmetric calculation if $r = XB(1) = 0$ (Note that `XB(1)` does not have to be 0). For better visualizations, the difference between `XB(4)` and `XB(3)` should be small so that the Smokeview rendering appears to be in 2-D. An example of an axially-symmetric helium plume is given in Sec. 21.2.

When processing results for a `CYLINDRICAL` simulation, note that integrated output quantities with the `SPATIAL_STATISTIC` attribute apply only to the specified 2-D or cylindrical coordinates. Thus, the cylindrical coordinates define a cylindrical sector, like a slice of cake, even though Smokeview will not render it this way. The fully integrated quantity can be calculated by multiplying the reported value by $2\pi\delta\theta$, where $\delta\theta$ is the difference between `YMAX` and `YMIN` in radians. The values chosen for `YMAX` and `YMIN` do not matter as long as the rendering in Smokeview is to your liking.

When performing solid phase heat transfer while using a 2-D `CYLINDRICAL` coordinate system, you must designate `GEOMETRY='CYLINDRICAL'` on a surface (`SURF` line) that is facing radially outward (positive r direction) or `GEOMETRY='INNER CYLINDRICAL'` on a surface that is facing radially inward (negative r direction). In the latter instance, you must also specify the `INNER_RADIUS` (m) of the cylinder. For the outer cylindrical boundary, specify an `INNER_RADIUS` if appropriate. Its default value is 0 m. Because your inward and outward facing boundaries might occur at various radii, you must create separate `SURF` lines for each with the appropriate values of `GEOMETRY` and `INNER_RADIUS`. For an obstruction (`OBST`), use `SURF_ID6` to assign individual `SURF` IDs to each of the six faces. Because this is a 2-D simulation, the third and fourth entries representing the `SURF` IDs in the y or angular direction can just be designated `'INERT'`.

6.3.3 Multiple Meshes

The term “multiple meshes” means that the computational domain consists of more than one computational mesh, usually connected although this is not required. If more than one mesh is used, there should be a `MESH` line for each. The order in which these lines are entered in the input file matters. In general, the meshes should be entered from finest to coarsest. FDS assumes that a mesh listed first in the input file has precedence over a mesh listed second if the two meshes overlap. Meshes can overlap, abut, or not touch at all. In the last case, essentially two separate calculations are performed with no communication at all between them. Obstructions and vents are entered in terms of the overall coordinate system and need not apply to any one particular mesh. Each mesh checks the coordinates of all the geometric entities and decides whether or not they are to be included.

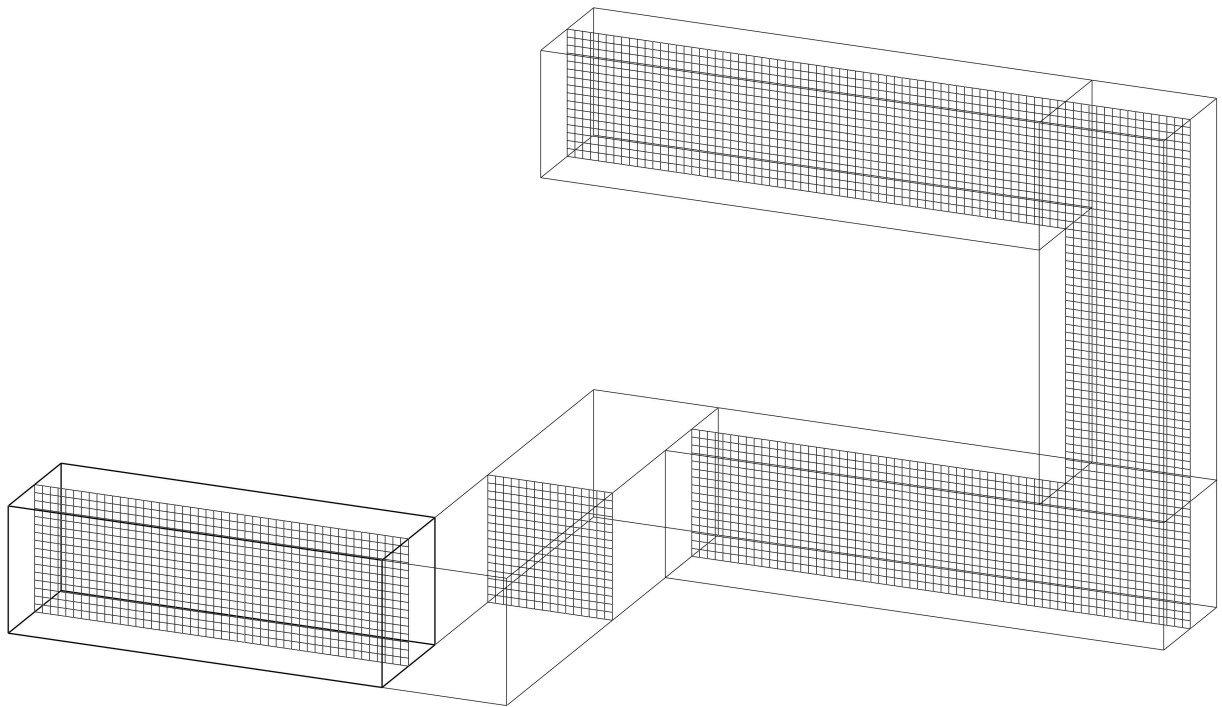


Figure 6.2: An example of a multiple-mesh geometry.

To run FDS in parallel using MPI (Message Passing Interface), you **must** break up the computational domain into multiple meshes so that the workload can be divided among the processes. In general, running multiple mesh cases using MPI is better if you have the resources available, but be aware that two processes

will not necessarily finish the job in half the time as one. For MPI to work well, there has to be a comparable number of cells assigned to each MPI process, or otherwise most of the processes will sit idle waiting for the one with the largest number of cells to finish processing each time step. You can use multiple meshes on a single processor without using MPI, in which case one CPU will serially process each mesh, one by one.

If your meshes are uniformly spaced, you can use the `MULT` feature (Section 7.6) to arrange them. For example, the lines

```
&MESH IJK=24,24,24, XB=-0.12,-0.06,-0.12,-0.06,-0.12,-0.06, MULT_ID='mesh' /
&MULT ID='mesh', DX=0.06, DY=0.06, DZ=0.06, I_UPPER=3, J_UPPER=3, K_UPPER=3 /
```

create a 4 by 4 by 4 array of meshes, each of which is 0.06 m on a side.

Usually in an MPI calculation, each mesh is assigned its own process, and each process its own processor. However, both assigning more than one mesh to a single process and assigning more than one process to a single processor is possible. Consider a case that involves six meshes:

```
&MESH ID='mesh1', IJK=..., XB=..., MPI_PROCESS=0 /
&MESH ID='mesh2', IJK=..., XB=..., MPI_PROCESS=1 /
&MESH ID='mesh3', IJK=..., XB=..., MPI_PROCESS=1 /
&MESH ID='mesh4', IJK=..., XB=..., MPI_PROCESS=2 /
&MESH ID='mesh5', IJK=..., XB=..., MPI_PROCESS=3 /
&MESH ID='mesh6', IJK=..., XB=..., MPI_PROCESS=3 /
```

The parameter `MPI_PROCESS` instructs FDS to assign that particular mesh to the given process. In this case, only four processes are to be started, numbered 0 through 3. Note that the processes need to be invoked in ascending order, starting with 0. Why would you do this? Suppose you only have four processors available for this job. By starting only four processes instead of six, you can save time because ‘mesh2’ and ‘mesh3’ can communicate directly with each other without having to transmit data using MPI calls over the network. Same goes for ‘mesh5’ and ‘mesh6’. In essence, it is as if these mesh pairs are neighbors and need not send mail to each other via the postal system. The letters can just be walked next door.

For cases involving many meshes, you might want to assign them colors using either the character string `COLOR` or the integer triplet `RGB`. You may also want to consider using the multiplying feature to easily create a 3-D array of meshes. See Sec. 7.6 for details.

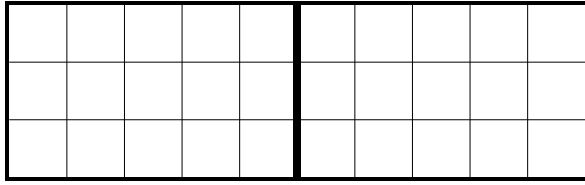
One other useful parameter for larger MPI jobs is called `VERBOSE` on the `MISC` line. This logical parameter suppresses information related to MPI process and OpenMP thread assignments that is printed to the diagnostic output files. By default, its value is `T` for MPI jobs involving 50 or less processes, and `F` for larger jobs.

6.3.4 Mesh Alignment

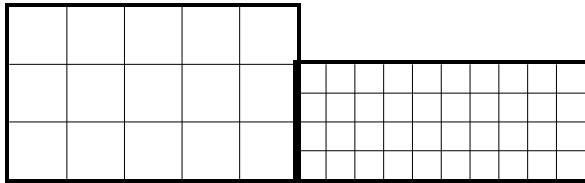
Whether the calculation is to be run using MPI or not, the rules of prescribing multiple meshes are similar, with some issues to keep in mind. The most important rule of mesh alignment is that abutting cells ought to have the same cross sectional area, or integral ratios, as shown in Fig. 6.3.

The following rules of thumb should also be followed when setting up a multiple mesh calculation:

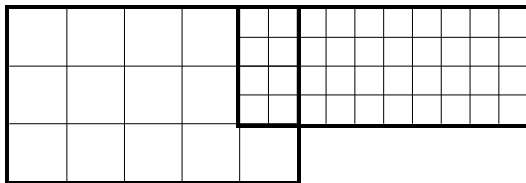
- When running multi-mesh calculations you need to closely monitor `VELOCITY_ERROR` at mesh interfaces, which is reported in the `CHID.out` file. Check that you are not consistently hitting the `MAX_PRESSURE_ITERATIONS`, which defaults to 10. If you are, then this indicates a potential problem and you may consider increasing this value to 20 or even 100 in some cases. Velocity errors are proportional to mass loss errors. If your errors are too large for your application, you may need to tighten the `VELOCITY_TOLERANCE` or else consider using a different pressure solver, see Sec. 21.1.



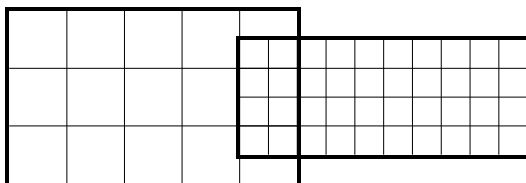
This is the ideal kind of mesh to mesh alignment.



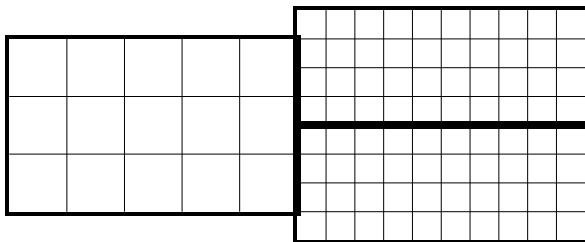
This is allowed so long as there are an integral number of fine cells abutting each coarse cell and each coarse cell only sees fine cells from one mesh.



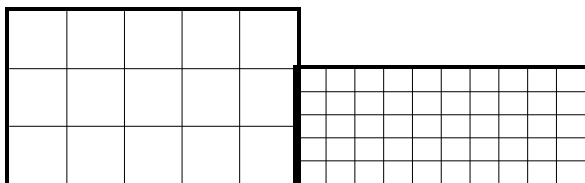
This is allowed, but of questionable value.



This is not allowed because each large cell must be completely covered by small ones.



This is not allowed because there is a coarse cell which sees fine cells from two different meshes.



This is not allowed because there is a non-integral number of fine cells abutting each coarse cell.

Figure 6.3: Rules governing the alignment of meshes.

- In general, there is little advantage to overlapping meshes because information is only exchanged at exterior boundaries. This means that a mesh that is completely embedded within another receives information at its exterior boundary, but the larger mesh receives no information from the mesh embedded within. Essentially, the larger, usually coarser, mesh is doing its own simulation of the scenario and is not affected by the smaller, usually finer, mesh embedded within it. Details within the fine mesh, especially related to fire growth and spread, may not be picked up by the coarse mesh. In such cases, it is preferable to isolate the detailed fire behavior within one mesh, and position coarser meshes at the exterior boundary of the fine mesh. Then the fine and coarse meshes mutually exchange information.
- Be careful when using the shortcut convention of declaring an entire face of the domain to be an `OPEN` vent. Every mesh takes on this attribute. See Sec. 7.4 for more details.
- If a planar obstruction is close to where two meshes abut, make sure that each mesh “sees” the obstruction. If the obstruction is even a millimeter outside of one of the meshes, that mesh does not account for it, in which case information is not transferred properly between meshes.

If you would like to check the mesh alignment without running the case, set `CHECK_MESH_ALIGNMENT` to `T` on any `MESH` line along with setting `T_END` to zero. If the job stops with no errors, the meshes obey the alignment rules. This check can sometimes take a few tens of seconds, but you can open Smokeview after launching the job to check the alignment by eye.

The criterion for rejecting the alignment of two meshes is as follows. Suppose the size of the abutting coarse grid cell is δx_c and the fine grid cell is δx_f . The meshes are out of alignment if

$$\left| \frac{\delta x_c - n \delta x_f}{\delta x_f} \right| < 0.001 \quad (6.1)$$

where n is the ratio of fine to coarse cells. The value of 0.001 can be changed via the `MISC` line parameter `ALIGNMENT_TOLERANCE`.

Accuracy of the Multiple Mesh Calculation

Experiment with different mesh configurations using relatively coarse mesh cells to ensure that information is being transferred properly from mesh to mesh. There are two issues of concern. First, does it appear that the flow is being badly affected by the mesh boundary? If so, check your velocity errors and consider tightening your velocity tolerance or changing pressure solvers, see Sec. 21.1. Second, is there too much of a jump in cell size from one mesh to another? If so, consider whether the loss of information moving from a fine to a coarse mesh is tolerable. Mesh refinement ratios of more than 4:1 to should be avoided if possible.

6.3.5 Mesh Stretching

By default the mesh cells that fill the computational domain are uniform in size. However, it is possible to specify that the cells be non-uniform in one or two¹ of the three coordinate directions. For a given coordinate direction, x , y or z , a function can be prescribed that transforms the uniformly-spaced mesh to a non-uniformly spaced mesh. **Be careful with mesh transformations!** If you shrink cells in one region you must stretch cells somewhere else. When one or two coordinate directions are transformed, the aspect ratio of the mesh cells in the 3D mesh will vary. To be on the safe side, transformations that alter the aspect ratio of cells beyond 2 or 3 should be avoided. Keep in mind that the large eddy simulation technique is based

¹If you are stretching the mesh in two coordinate directions, limit the number of cells to approximately 1 million. The Poisson solver that is used for two-coordinate stretching exhibits floating point overflow errors during the initialization phase when very large meshes are used. If you require more than a million cells, consider using multiple meshes instead of one large mesh.

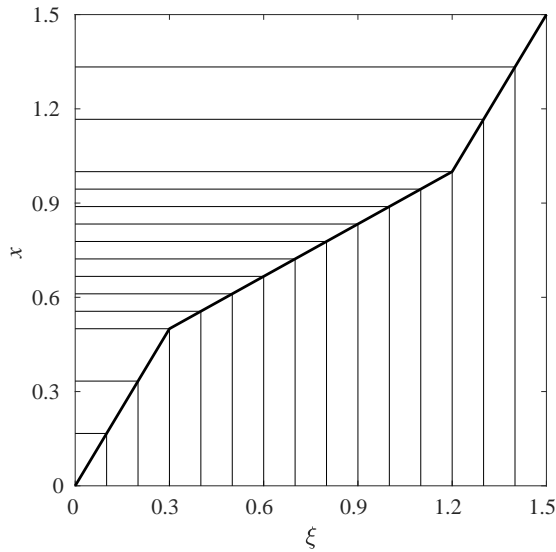


Figure 6.4: Piecewise-linear mesh transformation.

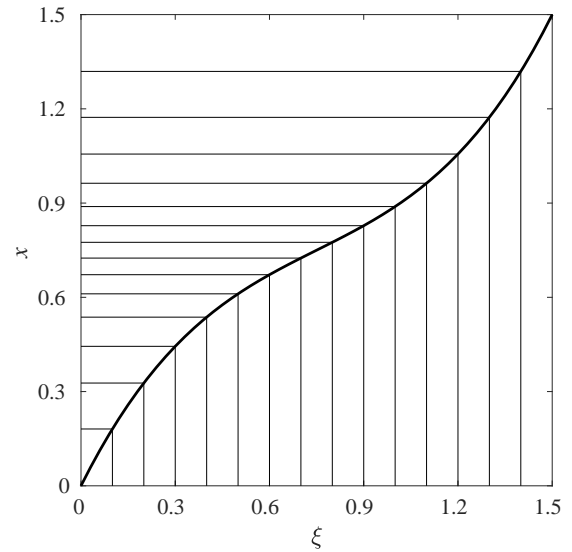


Figure 6.5: Polynomial mesh transformation.

on the assumption that the numerical mesh should be fine enough to allow the formation of eddies that are responsible for the mixing. In general, eddy formation is limited by the largest dimension of a mesh cell, thus shrinking the mesh resolution in one or two directions may not necessarily lead to a better simulation if the third dimension is large. Transformations, in general, reduce the efficiency of the computation, with two coordinate transformations impairing efficiency more than a transformation in one coordinate direction. Experiment with different meshing strategies to see how much of a penalty you will pay.

Here is an example of how to do a mesh transformation. Suppose your mesh is defined

```
&MESH IJK=15,10,20, XB=0.0,1.5,1.2,2.2,3.2,5.2 /
```

and you want to alter the uniform spacing in the x direction. First, refer to the figures above. You need to define a function $x = f(\xi)$ that maps the uniformly-spaced *Computational Coordinate* (CC) $0 \leq \xi \leq 1.5$ to the *Physical Coordinate* (PC) $0 \leq x \leq 1.5$. The function has three mandatory constraints: it must be monotonic (always increasing), it must map $\xi = 0$ to $x = 0$, and it must map $\xi = 1.5$ to $x = 1.5$. The default transformation function is $f(\xi) = \xi$ for a uniform mesh, but you need not do anything in this case.

Two types of transformation functions are allowed. The first, and simplest, is a piecewise-linear function. Figure 6.4 gives an example of a piecewise-linear transformation. The graph indicates how 15 uniformly spaced mesh cells along the horizontal axis are transformed into 15 non-uniformly spaced cells along the vertical axis. In this case, the function is made up of straight line segments connecting points (CC,PC), in increasing order, as specified by the following lines in the input file:

```
&MESH ..., TRNX_ID='my trnx' /
&TRNX ID='my trnx', CC=0.30, PC=0.50 /
&TRNX ID='my trnx', CC=1.20, PC=1.00 /
```

Note that an ID may be applied to a set of TRNX lines and invoked using a TRNX_ID, etc., on the desired MESH lines. This strategy also works when a MULT_ID is applied to the MESH. Alternatively, you may indicate the integer MESH_NUMBER (based on the order of the MESH lines) for which you wish to apply the transformation (this approach is deprecated because it requires a set of transformation lines for each mesh, which can be

impractical for MPI calculations). To apply the transformation to all meshes, set `MESH_NUMBER` to 0. Thus, an equivalent representation of the previous example applied to all meshes would be:

```
&TRNX CC=0.30, PC=0.50, MESH_NUMBER=0 /
&TRNX CC=1.20, PC=1.00, MESH_NUMBER=0 /
```

The parameter `CC` refers to the Computational Coordinate, ξ , located on the horizontal axis; `PC` is the Physical Coordinate, x , located on the vertical axis. The slopes of the line segments in the plot indicate whether the mesh is being stretched (slopes greater than 1) or shrunk (slopes less than 1). The tricky part about this process is that you usually have a desired shrinking/stretching strategy for the Physical Coordinate on the vertical axis, and must work backward to determine what the corresponding points should be for the Computational Coordinate on the horizontal axis. Note that the above transformation is applied to the second mesh in a multiple mesh job. It should be also noted that the start and endpoints should not be specified in this linear grid transformation.

The second type of transformation is a polynomial function whose constraints are of the form

$$\frac{d^n f(CC)}{d\xi^n} = PC$$

Figure 6.5 gives an example of a polynomial transformation, for which the parameters are specified:

```
&MESH ..., TRNX_ID='my trnx' /
&TRNX ID='my trnx', IDERIV=0, CC=0.75, PC=0.75 /
&TRNX ID='my trnx', IDERIV=1, CC=0.75, PC=0.50 /
```

which correspond to the constraints $f(0.75) = 0.75$ and $\frac{df}{d\xi}(0.75) = 0.5$, or, in words, the function maps 0.75 into 0.75 and the slope of the function at $\xi = 0.75$ is 0.5. The transform function must also pass through the points (0,0) and (1.5,1.5), meaning that FDS must compute the coefficients for the cubic polynomial $f(\xi) = c_0 + c_1 \xi + c_2 \xi^2 + c_3 \xi^3$. More constraints on the function lead to higher order polynomial functions, so be careful about too many constraints which could lead to non-monotonic functions. The monotonicity of the function is checked by the program and an error message is produced if it is not monotonic.

Do not specify either linear transformation points or `IDERIV=0` points at coordinate values corresponding to the mesh boundaries. This is done automatically by FDS.

6.3.6 Mesh Resolution

A common question asked by new FDS users is, “What should my grid spacing be?” The answer is not easy because it depends considerably on what you are trying to accomplish. In general, you should build an FDS input file using a relatively coarse mesh, and then gradually refine the mesh until you do not see appreciable differences in your results. This is referred to as a mesh sensitivity study.

For simulations involving buoyant plumes, a measure of how well the flow field is resolved is given by the non-dimensional expression $D^*/\delta x$, where D^* is a characteristic fire diameter

$$D^* = \left(\frac{\dot{Q}}{\rho_\infty c_p T_\infty \sqrt{g}} \right)^{2/5} \quad (6.2)$$

and δx is the nominal size of a mesh cell². The quantity, \dot{Q} , is the total heat release rate of the fire. If it changes over time, you should consider the corresponding change in resolution. The quantity $D^*/\delta x$ can be

²The characteristic fire diameter is related to the characteristic fire size via the relation $Q^* = (D^*/D)^{5/2}$, where D is the physical diameter of the fire.

thought of as the number of computational cells spanning the characteristic (not necessarily the physical) diameter of the fire. The more cells spanning the fire, the better the resolution of the calculation. It is better to assess the quality of the mesh in terms of this non-dimensional parameter, rather than an absolute mesh cell size. For example, a cell size of 10 cm may be “adequate,” in some sense, for evaluating the spread of smoke and heat through a building from a sizable fire, but may not be appropriate to study a very small, smoldering source.

The FDS Validation Guide [5] contains a table of the values of $D^*/\delta x$ used in the simulation of the validation experiments. The table is near the end of the chapter that describes all the experiments. These values range over two orders of magnitude and were chosen based on a grid resolution study and the particular attributes of the given fire scenario. Taking any of these values as an “acceptable” minimum is inappropriate.

There are a number of special output quantities that provide local measures of grid resolution. See Sec. 22.10.28 for details.

Chapter 7

Model Geometry

A considerable amount of work in setting up a calculation lies in specifying the geometry of the space to be modeled and applying boundary conditions to the solid surfaces. The geometry is described in terms of rectangular obstructions that can heat up, burn, conduct heat, and so on, and vents from which air or fuel can be either injected into, or drawn from, the flow domain. A boundary condition needs to be assigned to each obstruction and vent describing its thermal properties. A fire is just one type of boundary condition. This chapter describes how to build the model.

7.1 Specifying Boundary Conditions

Before describing how to build up the geometry, it is first necessary to explain how to describe what comprises the bounding surfaces of the geometry. The `SURF` namelist group defines the structure of all solid surfaces or openings within or bounding the flow domain. Boundary conditions for obstructions and vents are prescribed by referencing the appropriate `SURF` line(s) whose parameters are described in this section.

The default boundary condition for all solid surfaces is that of a smooth inert wall with the surface temperature fixed at `TMPA`, and is referred to as '`INERT`'. Do not confuse this with a surface for which heat transfer does not occur (this is an adiabatic surface which will change in temperature to maintain zero heat transfer, refer to Sec. 8.2.3). To maintain the surface temperature at `TMPA` when exposed to a gas temperature, which may be above or below `TMPA`, FDS calculates the required heat transfer coefficient, and hence the heat transfer between the surface and the gas phase. You can think of `INERT` as the ultimate ambient temperature heat sink; such as a water-cooled panel.

If only this boundary condition is needed, there is no need to add any `SURF` lines to the input file. If additional boundary conditions are desired, they are to be listed one boundary condition at a time. Each `SURF` line consists of an identification string `ID='...'` to allow references to it by an obstruction or vent. Thus, on each `OBST` and `VENT` line that are to be described below, the character string `SURF_ID='...'` indicates the `ID` of the `SURF` line containing the desired boundary condition parameters. If a particular `SURF` line is to be applied as the default boundary condition, set `DEFAULT=T` on the `SURF` line.

7.2 Creating Rectilinear Obstructions

The namelist group `OBST` contains parameters used to define obstructions. The entire geometry of the model is made up entirely of rectangular solids, each one introduced on a single line in the input file.

7.2.1 Basics

Each `OBST` line contains the coordinates of a rectangular solid within the flow domain. This solid is defined by two points (x_1, y_1, z_1) and (x_2, y_2, z_2) that are entered on the `OBST` line in terms of the real sextuplet `XB`. In addition to the coordinates, the boundary conditions for the obstruction can be specified with the parameter `SURF_ID`, which designates which `SURF` line (Sec. 7.1) to apply at the surface of the obstruction. If the obstruction has different properties for its top, sides and bottom, do not specify only one `SURF_ID`. Instead, use `SURF_IDS`, an array of three character strings specifying the boundary condition `IDS` for the top, sides and bottom of the obstruction, respectively. If the default boundary condition is desired, then `SURF_ID` or `SURF_IDS` need not be set. However, if at least one of the surface conditions for an obstruction is the inert default, it can be referred to as `'INERT'`, but it does not have to be explicitly defined. For example:

```
&SURF ID='FIRE', HRRPUA=1000.0 /  
&OBST XB=2.3,4.5,1.3,4.8,0.0,9.2, SURF_IDS='FIRE','INERT','INERT' /
```

puts a fire on top of the obstruction. This is a simple way of prescribing a burner.

In addition to `SURF_ID` and `SURF_IDS`, you can also use the sextuplet `SURF_ID6` as follows:

```
&OBST XB=2.3,4.5,1.3,4.8,0.0,9.2,  
SURF_ID6='FIRE','INERT','HOT','COLD','BLOW','INERT' /
```

where the six surface descriptors refer to the planes $x = 2.3$, $x = 4.5$, $y = 1.3$, $y = 4.8$, $z = 0.0$, and $z = 9.2$, respectively. Note that `SURF_ID6` should not be used on the same `OBST` line as `SURF_ID` or `SURF_IDS`.

Obstructions may be created or removed during a simulation. See Sec. 18.4.1 for details.

7.2.2 Thin Obstructions

An obstruction that is relatively thin compared to the gas phase numerical grid spacing is approximated as an infinitely thin sheet. These obstructions, like window panes, form a flow and (partial) radiation barrier, but if the numerical mesh is coarse relative to its thickness, the obstruction might be unnecessarily large if it is assumed to be one layer of mesh cells thick. Smokeview renders this obstruction as a thin sheet, but it is allowed to have thermally-thick boundary conditions.

This obstruction has the drawback that it is not possible to specify a normal velocity component on its surface. A thin sheet obstruction can only have one velocity vector on its face, thus a gas cannot be injected or extracted reliably from a thin obstruction because whatever is pushed from one side is necessarily pulled from the other. If you want to create a simple fan, for example, the obstruction should be specified to be at least one mesh cell thick. To prevent FDS from allowing thin sheet obstructions, set `THICKEN_OBSTRUCTIONS=T` on the `MISC` line, or `THICKEN=T` on each `OBST` line for which the thin sheet assumption is not allowed.

Another drawback occurs when using the default FFT-based pressure solver. Thin obstructions are more prone to errors in the normal velocity at the obstruction. If thin obstructions see large pressure gradients across the obstruction or are impacted by high normal velocities, then tighter tolerances for the FFT-based solver or the use of another pressure solver may be warranted. For details see the discussion in Sec. 21.1.

Thin obstructions may generate a mass source without specification of a normal velocity component. The gas cell next to the obstruction is given a volumetric source term, but the momentum at the surface remains zero.

7.2.3 Specified Versus Actual Areas

The `OBST` dimensions specified by `XB` in the input file may not (usually do not) perfectly align with the underlying Cartesian mesh. FDS then “snaps” the geometry to fill (or void) the volume of the nearest cell. Obstructions that are too small relative to the underlying numerical mesh are rejected. That is, if at least two of the three obstruction dimensions are less than half of the corresponding cell dimensions, the obstruction is ignored. As mentioned above, if you always want the cell to be filled, then specify `THICKEN=T` on the `OBST` line.

When the `OBST` snaps to the Cartesian mesh, the area of the cell face is changed, and therefore the total area of a `SURF` may slightly change. If a specified mass flux boundary (including `HRRPUA`) is given, then FDS will automatically adjust this mass flux so that total of the *specified area* times the *specified mass flux* is maintained.

7.2.4 Overlapping Obstructions

If the faces of two obstructions overlap each other, FDS will choose the surface properties of the obstruction that is specified second in the input file. If this is unwanted, add `OVERLAY=F` to the `OBST` line of the second obstruction, in which case the surface properties of the first obstruction will be applied. The default value of `OVERLAY` is `T`.

When obstructions overlap, Smokeview renders both obstructions independently of each other, often leading to an unsightly cross-hatching of the two surface colors where there is an overlap. A simple remedy for this is to “shrink” the obstruction you do not wish to take precedence by slightly by adjusting its coordinates (`XB`) accordingly. Then, in Smokeview, toggle the “q” key to show the obstructions as you specified them, rather than as FDS rendered them.

The simulation will be stopped with an error if all of the following occur: (1) surfaces of two obstructions overlap, (2) the surface properties are different, and (3) one or both of the obstructions can be removed or created during the simulation.

7.2.5 Preventing Obstruction Removal

Obstructions can be protected from the `HOLE` punching feature. Sometimes creating a door or window using a `HOLE` is convenient. For example, suppose a `HOLE` is punched in a wall to represent a door or window. An obstruction can be defined to fill this hole (presumably to be removed or colored differently or whatever) so long as the phrase `PERMIT_HOLE=F` is included on the `OBST` line. In general, any obstruction can be made impenetrable to a `HOLE` using this phrase. By default, `PERMIT_HOLE=T`, meaning that an obstruction is assumed to be penetrable unless otherwise directed. Note that if a penetrable obstruction and an impenetrable obstruction overlap, the obstruction with `PERMIT_HOLE=F` should be listed first.

If the obstruction is not to be removed or rejected for any reason, set `REMOVABLE=F`. This is sometimes needed to stop FDS from removing the obstruction if it is embedded within another, like a door within a wall.

In rare cases, you might not want to allow a `VENT` to be attached to a particular obstruction, in which case set `ALLOW_VENT=F`.

7.2.6 Transparent or Outlined Obstructions

Obstructions can be made semi-transparent by assigning a `TRANSPARENCY` on the `OBST` line. This real parameter ranges from 0 to 1, with 0 being fully transparent. The parameter should always be set along with either `COLOR` or an `RGB` triplet. It can also be specified on the appropriate `SURF` line, along with a color indicator. If you want the obstruction to be invisible, set `COLOR='INVISIBLE'`.

Obstructions are typically drawn as solids in Smokeview. To draw an outline representation, set `OUTLINE` equal to `T`.

7.2.7 Method for avoiding mesh interfaces in blockage outline view

FDS splits a blockage into separate parts, one part for each mesh. As a result when Smokeview draws blockages as outlines it draws extra lines where blockages intersect mesh interfaces. The following procedure can be used to eliminate these extra lines for a case named say `casename.fds`.

1. Create a one mesh version of `casename.fds`. Name it `casename_single.fds` (or anything different than `casename.fds`. Set `T_END` to 0 and `CHID` to `casename_single`.
2. After running `casename_single.fds` with `fds`, rename `casename_single.smv` to `casename.smo`.
3. Issue the command: `smokeview casename`.
4. Select the `Show/Hide>Geometry>Obstacles>Outline only` menu item to view the blockages as outlines.
5. press the `q` key to toggle between viewing blockages split at mesh boundaries (how `fds` handles them) and how blockages are defined in the input file.

Figure 7.1 shows an example where blockages are drawn as specified in the input file and drawn split at mesh interfaces.

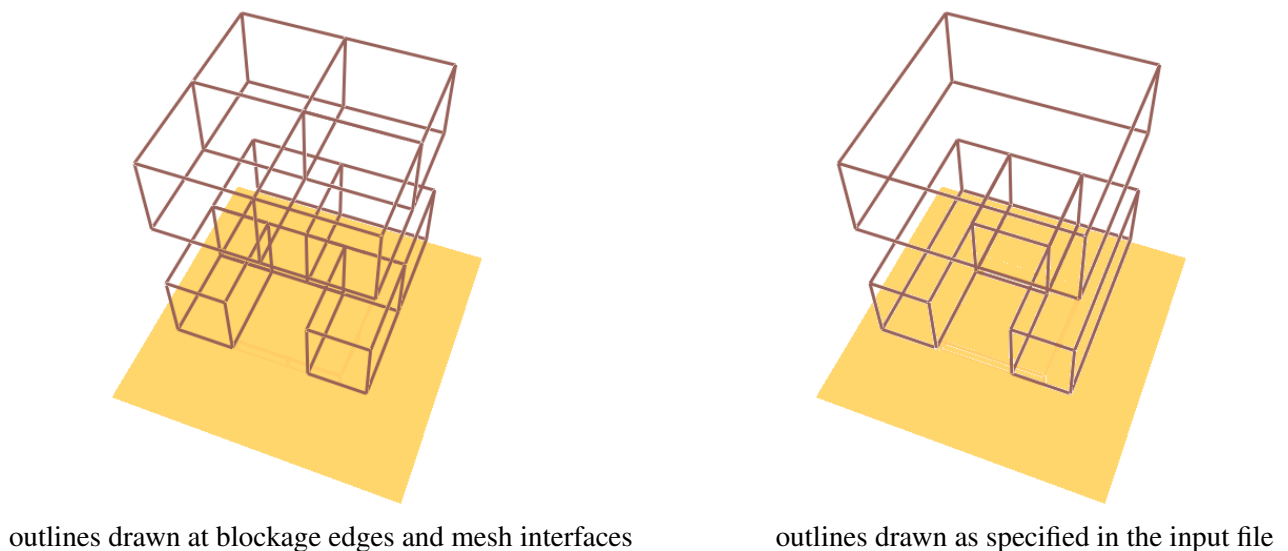


Figure 7.1: Blockage outlines drawn at mesh interfaces and as specified in the input file.

7.2.8 Creating Holes in Obstructions

The `HOLE` namelist group defines parameters that carve a hole out of an existing obstruction or set of obstructions. To do this, add lines of the form

```
&HOLE XB=2.0,4.5,1.9,4.8,0.0,9.2 /
```

Any solid mesh cells within the volume $2.0 < x < 4.5$, $1.9 < y < 4.8$, $0.0 < z < 9.2$ are removed. Obstructions intersecting the volume are broken up into smaller blocks. If the hole represents a door or window, a good rule of thumb is to punch more than enough to create the hole. This ensures that the hole is created through the entire obstruction. For example, if the `OBST` line denotes a wall 0.1 m thick:

```
&OBST XB=1.0,1.1,0.0,5.0,0.0,3.0 /
```

and you want to create a door, add this:

```
&HOLE XB=0.99,1.11,2.0,3.0,0.0,2.0 /
```

The extra centimeter added to the x coordinates of the hole make it clear that the hole is to punch through the entire obstruction.

When a `HOLE` is created, the affected obstruction(s) are either rejected, or created or removed at pre-determined times. See Sec. 18.4.1 for details. To allow a hole to be controlled with either the `CTRL` or `DEVC` namelist groups, you will need to add the `CTRL_ID` or `DEVC_ID` parameter respectively, to the `HOLE` line¹. When the state of the `HOLE` evaluates to `F`, an obstruction will be placed in the `HOLE`. By default the obstruction filling the `HOLE` will take the color of the surrounding `OBST` that the `HOLE` was punched through. To make the obstruction filling the `HOLE` a different color than the original obstruction, set the `COLOR` or integer triplet `RGB` on the `HOLE` line (see Sec. 7.5). If you want the obstruction filling the `HOLE` to be invisible, then set `COLOR='INVISIBLE'`. Additionally, you may use the keyword `TRANSPARENCY`, real number from 0 to 1, to make the obstruction filling the `HOLE` transparent. See Sec. 18.4.1 for an example.

If an obstruction is not to be punctured by a `HOLE`, add `PERMIT_HOLE=F` to the `OBST` line. Note that a `HOLE` has no effect on a `VENT` or a mesh boundary. It only applies to `OBSTRUCTIONS`.

It is a good idea to inspect the geometry by running either a setup job (`T_END=0` on the `TIME` line) or a short-time job to test the operation of devices and control functions.

¹If you add a `CTRL_ID` or `DEVC_ID` to the `HOLE` line, do not overlap this `HOLE` with another. The control logic can fail.

7.3 Creating Unstructured Geometry (Beta)

The namelist `GEOM` describes one or more unstructured closed geometric surfaces that enclose solid portions of the fluid domain. These surfaces consist of a collection of triangular faces, where each face is built from three vertices. The user can assign a specific boundary condition to each of the faces.

7.3.1 My First Geometry

To get you acquainted with the new geometry features we allow for some of `OBST` namelist parameters to be applied to `GEOM`. Therefore, subject to the limitations listed at the end of this section (see Sec. 7.3.11), you should be able to take a case built from `OBST` and simply do a *replace* with `GEOM` to begin using unstructured geometry. For example, if you have a burner built like this

```
&OBST XB=-1,1,-1,1,0,0.5, SURF_IDS='FIRE','INERT','INERT' /
```

You can replace this with

```
&GEOM XB=-1,1,-1,1,0,0.5, SURF_IDS='FIRE','INERT','INERT' /
```

Further, the `GEOM` may be *transformed* using a `MOVE` line as discussed in Sec. 7.3.8.

7.3.2 Unstructured Geometry Basics

A simple example of an unstructured solid is given by

```
&GEOM ID='My Solid'
      SURF_ID='FIRE','INERT'
      VERTS= -1.0, -1.0,  0.0,
              1.0, -1.0,  0.0,
              0.0,  1.0,  0.0,
              0.0,  0.0,  1.0,
      FACES= 1,3,2, 2,
              1,4,3, 1,
              3,4,2, 1,
              2,4,1, 0 /
```

`ID` specifies the name, 'My Solid', `VERTS` specify the coordinates of each vertex ($x_1, y_1, z_1; x_2, y_2, z_2, \dots$), and `FACES` define each triangular face—three vertices and one boundary condition index: (v_1, v_2, v_3, b) . The vertex indices range from 1 to the number of vertices listed in `VERTS`. The order of the three vertices defines which side of the triangle is outward facing. Observed from the outside, the vertices of each triangle should be ordered counter-clockwise, following the right-hand rule convention.

The boundary condition index b ranges from 0 to the number of entries in `SURF_ID`. $b = 0$ applies the default boundary condition. In the example above, the first face has an index, $b = 2$, which corresponds to 'INERT', the second and third faces have $b = 1$ corresponding to 'FIRE', and the fourth corresponds to the default.

7.3.3 Triangulated Surfaces Quality

In order for FDS to correctly detect the solid portion of the volume from the rest of the fluid domain, some tests are performed at start on the quality of the triangulated surfaces.

The following conditions are enforced by FDS on the collection of vertices and faces of each `GEOM` namelist:

1. Each vertex must be unique, shared by edges, be part of an only well-formed surface so that its neighboring ring of faces could in theory be continuously deformed to a disk: a vertex that abides to these conditions is called *non-degenerate* and *manifold*. Examples of good and bad vertices are presented in Figure 7.4.
2. Each edge connecting vertices must have a non-zero length, and must always be shared by exactly two faces. An edge that abides to these conditions is called *non-degenerate* and *manifold*. See examples of good and bad edges in Figure 7.5.
3. Each face cannot intersect the others and must have a non-zero area; such a face is called *non-intersecting* and *non-degenerate*.
4. The vertex orderings for the faces shall be chosen so that adjacent faces have outward consistent normals, as illustrated in Figure 7.2.
5. Each `GEOM` namelist may define one or more distinct volumes (*unconnected* volumes) that should not mutually intersect or self-intersect. See an example of a bad intersecting geometry in Figure 7.8.

If the previous conditions are respected, a well-formed `GEOM` namelist is obtained that represents one or more *closed, manifold, orientable, non intersecting* triangulated surfaces, that enclose well-defined volumes. An example of a well-formed triangulated surface is presented in Figure 7.3.

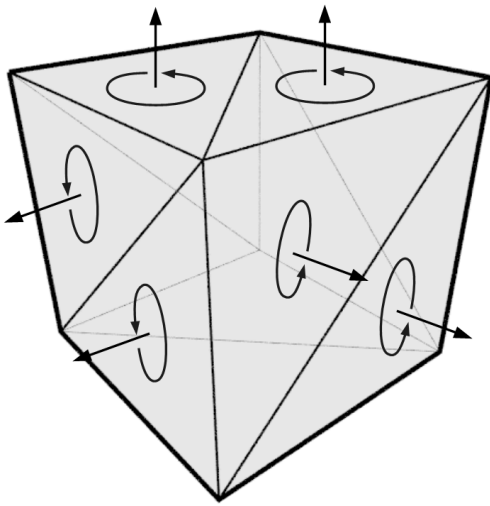


Figure 7.2: **Good GEOM:** consistent normals on a manifold orientable surface.

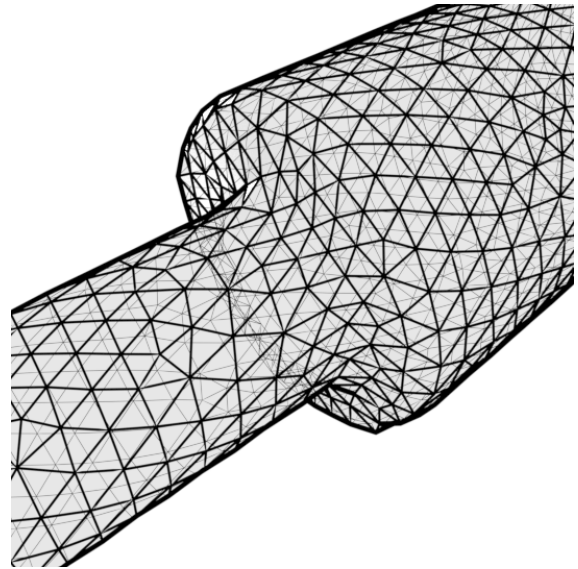


Figure 7.3: **Good GEOM:** a well formed triangulated surface.

Whenever one of the checks on these conditions fails, FDS outputs an error message describing the problem and its location in the domain. For example:

```
ERROR: GEOM ID='Cube': Non manifold geometry at edge: ...
```

7.3.4 Intersections

Several types of intersections can happen between GEOMS and between GEOMS and other namelists:

- **GEOM - GEOM and GEOM - OBST:** whenever these types of intersection happen, FDS attempts to perform a boolean operation uniting the objects, respecting the boundary conditions of the surfaces. When a geometry intersects with another geometry, the vertices and edges defining that intersection have not been input by the user. The exact intersection may be very complicated to discern in the general case, and we employ a linearization that generates the approximate boundary faces within the cell from edges defined on its Cartesian faces. An example for two intersecting spheres is shown below in Fig. 7.9. If the geometry is too complex, this operation can fail. In such a case the cell will be left void. The user is recommended to apply geometry unions in their preprocessor of choice if possible.
- **GEOM - HOLE:** currently there is no interaction.

7.3.5 Coloring and Texture Maps

Coloring and texture maps for GEOM objects are specified in the same way as for OBST (see Sec. 7.5).

7.3.6 Self-Generated Geometries

The GEOM namelist allows the quick definition of particular kinds of geometric objects, blocks using XB, spheres using SPHERE_ORIGIN and SPHERE_RADIUS and a 2D terrain elevations using ZVALS. FDS generates

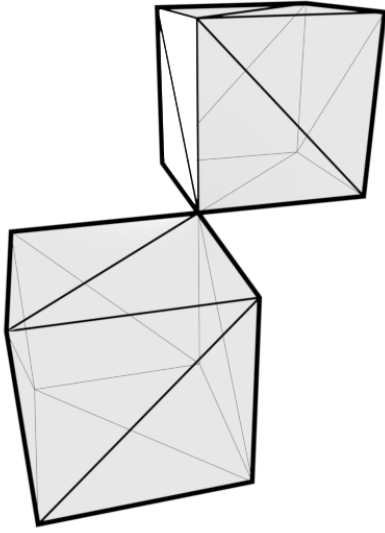


Figure 7.4: **Bad GEOM:** the vertex shared by the two cubes is not manifold, because it is connected to two rings of faces, that cannot be continuously deformed to a single disk. The other vertices are manifold.

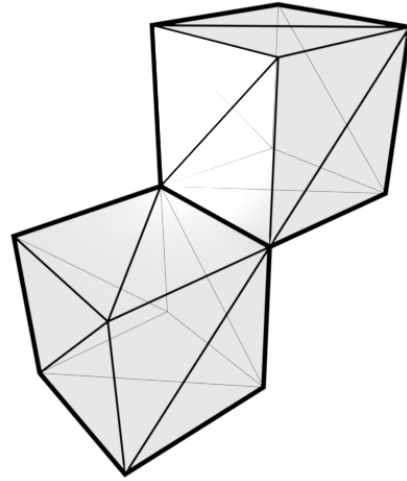


Figure 7.5: **Bad GEOM:** the edge shared by the two cubes is not manifold, because more than two faces are incident on it. The other edges are manifold.

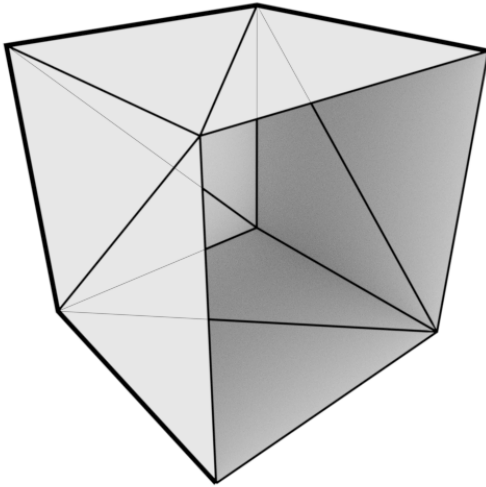


Figure 7.6: **Bad GEOM:** FDS cannot differentiate the internal volume from the rest of the domain, because the triangulated surface is open.

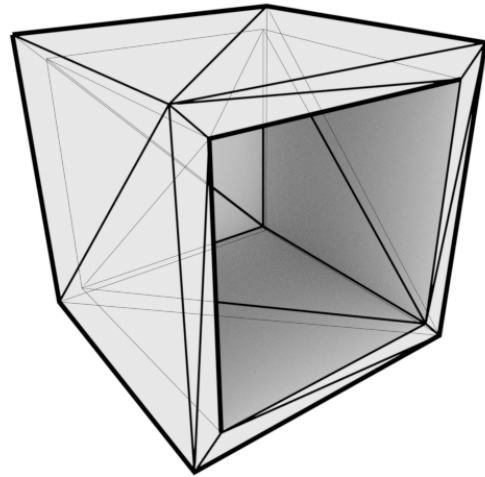


Figure 7.7: **Good GEOM:** by adding a thickness FDS can differentiate the internal volume from the rest of the domain.

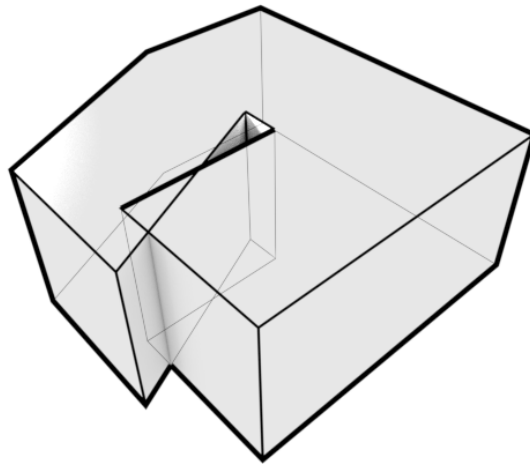


Figure 7.8: **Bad GEOM:** This volume is self-intersecting.

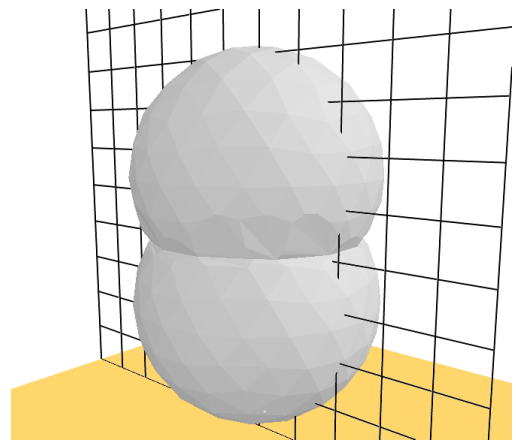
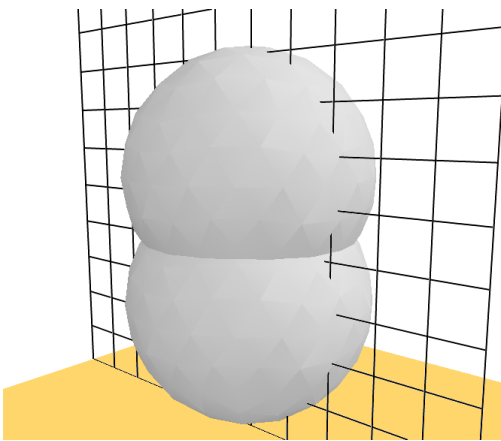


Figure 7.9: Linearization at intersections: sphere example. (Left) Two intersecting spheres as input to FDS. (Right) The final linearized geometry without hanging vertices.

vertices and faces to represent these objects, equivalent to what would have been defined using `VERTS` and `FACES`.

Blocks

A `GEOM` namelist defining a block is specified by

```
&GEOM ID='block'  
      SURF_ID='S1'  
      XB=0.0,1.0,0.0,1.0,0.0,1.0/
```

where `XB=xmin,xmax,ymin,ymax,zmin,zmax` defines the min and max bounds of the block. The `XB` parameter is used in the same way as on an `OBST` or `VENT` line. A block may be refined into many parts by specifying the `IJK` parameter. For example, `IJK=8,6,4` would split the block into 8 parts along the x dimension, 6 parts along the y dimension and 4 parts along the z dimension.

The `SURF_ID` parameter assigns the specified boundary condition to all the generated faces. Similar to `OBST`, you can also use `SURF_IDS` and `SURF_ID6` lists to specify surface conditions (see Sec. 7.2.1).

Spheres

A `GEOM` namelist defining a sphere centered at $(0,0,0)$ with radius 1 is specified by

```
&GEOM ID='sphere'  
      SURF_ID='S1'  
      SPHERE_ORIGIN=0.0,0.0,0.0, SPHERE_RADIUS=1.0/
```

Spheres are discretized by default so that each face is consistent in size with the grid resolution. One may specify a `N_LEVEL` parameter which defines the number of times the sphere is split.

Cylinders

A `GEOM` namelist defining a cylinder geometry with centroid at `CYLINDER_ORIGIN = (0,0,0)`, axis `CYLINDER_AXIS = (1.,0,0)`, length `CYLINDER_LENGTH = 2.` and radius `CYLINDER_RADIUS = 0.5` is specified by

```
&GEOM ID='cylinder', SURF_ID='S1',  
      CYLINDER_LENGTH=2.,  
      CYLINDER_RADIUS=.5,  
      CYLINDER_ORIGIN=0.,0.,0.,  
      CYLINDER_AXIS=1.,0.,0.,  
      CYLINDER_NSEG_AXIS=6,  
      CYLINDER_NSEG_THETA=48 /
```

where the cylinder is discretized using 6 segments along its axis and 48 segments across its circumference. The keyword `SURF_ID` can be replaced by `SURF_IDS='S1','S2','S3'`, where the strings `'S1'`, `'S2'`, `'S3'` correspond to the surface IDs of top, sides and bottom surfaces respectively.

2D Terrain Elevations

A `GEOM` namelist defining a 2D terrain elevation is given by

```
&GEOM ID='terrain'  
      SURF_ID='S1'
```

```

IJK=ivals, jvals, XB=xmin, xmax, ymin, ymax,
EXTEND_TERRAIN=etlog, ZVAL_HORIZON=zvhr,
ZVALS=.../

```

where `XB` defines a rectangular region bounded by (x_{\min}, y_{\min}) and (x_{\max}, y_{\max}) and where `ZVALS` defines elevation data in this region. `IJK` specifies how many vertices occur in this region along each dimension. In this example `ivals` values occur along the x dimension and `jval` values occur along the y dimension. The `ZVALS` keyword is used to specify elevations at each (x, y) location. The elevation data specified after the `ZVALS` keyword is arranged in row major order. The first row contains `ivals` elevation values occurring at the y_{\min} position from x_{\min} to x_{\max} . Similarly, the last row contains `ivals` elevation values for the y_{\max} position again from x_{\min} to x_{\max} . There are then `jvals` rows and `ivals` columns of elevation data. The optional fields `EXTEND_TERRAIN` and `ZVAL_HORIZON` are used in the following form. If the computational domain is set larger than the `XB` defined terrain patch, such that it contains the terrain completely, setting `EXTEND_TERRAIN=T` allows for the terrain to be extended to the domain boundaries. This can be helpful when trying to reduce boundary effects on a calculation. If, additionally, the real number `ZVAL_HORIZON` is set on the `GEOM` line, this elevation will be used on the domain boundaries when extending the domain. As with the blocks and spheres, FDS uses the information provided by these keywords to construct vertices and triangular faces. The bottom elevation of the geometry (used only for visualization purposes in Smokeview) may be set with the parameter `ZMIN`.

Additionally, terrain geometries can be specified defining a triangulation with a convex regular boundary, as seen from the top. This triangulation will represent the terrain of interest and will define the top boundary of a terrain `GEOM`. The terrain surface is specified by `VERTS` and `FACES` as in a basic geometry (Sec. 7.3.2), and the field `IS_TERRAIN` needs to be set to `T`, such that FDS knows this is a terrain unstructured surface. Surface types per faces are specified in the same manner as in basic geometries. Also, the `EXTEND_TERRAIN` logic can also be applied.

Geometries Extruded From Simple Planar Polygons

In this case, a geometry can be defined specifying the `VERTS`, connectivity `POLY`, and extrusion distance `EXTRUDE`. For example:

```

&GEOM ID='ExtPoly', SURF_ID='Poly1',
VERTS =
  -1.0, -1.0,  1.0,
    0.5, -1.0,  1.75,
    1.0, -1.0,  2.0,
    0.0,  0.0,  1.5,
    1.0,  1.0,  2.0,
   -1.0,  1.0,  1.0,
POLY = 1, 2, 4, 3, 5, 6,
EXTRUDE=0.5 /

```

The polygon defined by `POLY` must be simple, non degenerate (no segment intersections, no repeated vertices) and all segments must lay on the same plane in three dimensional space. It can have a maximum of 1000 vertices. The value of `EXTRUDE` can be a positive or negative real, but not zero. A single surface ID can be provided, or top, sides and bottom surface IDs through a `SURF_IDS` field, in the same manner as the cylinders of Sec. 7.3.6. Note that the bottom surface in `SURF_IDS` always refers to the polygon defined in `POLY`, from where the extrusion operation is performed.

7.3.7 Generating Complex Geometries

Generating well-formed complex geometries is a time-consuming and error-prone process. Several third-party pre-processing tools are provided by the FDS-SMV community for automatically translating CAD files to GEOM namelists.

These tools should always enforce quality checks on the triangulated surfaces and produce well-formed GEOM namelists. Links to these tools can be found at the project home page at <https://pages.nist.gov/fds-smv/> under “Third-Party Tools and Training”.

7.3.8 Transforming Objects

A geometry may be translated, rotated and scaled. For example, in the following GEOM namelist,

```
&MOVE ID='ChairMove', AXIS=..., ..., ROTATION_ANGLE=..., DX=, /
&GEOM ID='chair'
  SURF_ID='woodSID', 'apholsterySID',
  VERTS=X1,Y1,Z1, ..., XM,YM,ZM,
  FACES=F1_1,F1_2,F1_3,F1_SID, ..., FN_1,FN_2,FN_3,FN_SID
  MOVE_ID='ChairMove' /
```

the `chair` object is translated and rotated using a reference to the `MOVE` namelist, defined and explained in Sec. 11.4.

7.3.9 Reading Geometry Node Locations And Connectivity Data From Binary

In cases where geometries have very large numbers of nodes and surface triangles, greater setup time efficiency can be achieved by reading this data from an FDS produced binary file. This binary file can be obtained running the case in setup only mode (`T_END=0.0`). Consider the chair geometry defined in the previous section. When a geometry is read by FDS, the `XYZ` and face connectivity data `FACES` is automatically written into a binary file with name `CHID_GEOM_ID.binggeom` on the run directory (here, `GEOM_ID='chair'`). If an `ID` is not provided, the name of the binary file will be `CHID_N_GEOM.binggeom`, where `N_GEOM` is the number defining the order of appearance of the geometry line on the FDS input file.

Then, supposing the case `ID` is `CHID='DiningRoom'`, the next time the case is run, the geometry line can be modified to:

```
&GEOM ID='chair',
  SURF_ID='woodSID', 'apholsterySID',
  BINARY_FILE='DiningRoom_chair.binggeom',
  MOVE_ID='ChairMove' /
```

where the presence of `BINARY_FILE` instructs FDS to read nodes and faces from the binary geometry file. Note we have deleted the `VERTS`, `FACES` arrays from the input line. This manner of loading large geometries avoids the testing and reading of all lines related to these arrays on the FDS input file. This approach is particularly efficient in large parallel calculations, where parallel file systems are used. See Sec. 27.19 for more information on geometry I/O binary format. If a different name than the default is used, it can be specified in the field `BINARY_FILE`. The location directory can also be added in the `BINARY_FILE` string. Terrains can also be read and written in binary format, and in this case only their top surface triangulation is recorded. The binary contains their geometry information and terrain type, but the `SURF_ID` array has to be provided in the geometry line as shown above. The same requirement is needed for `EXTEND_TERRAIN` if terrain extension is desired. The number of surface IDs defined in the binary file connectivities must match the number of surfaces defined in `SURF_ID`.

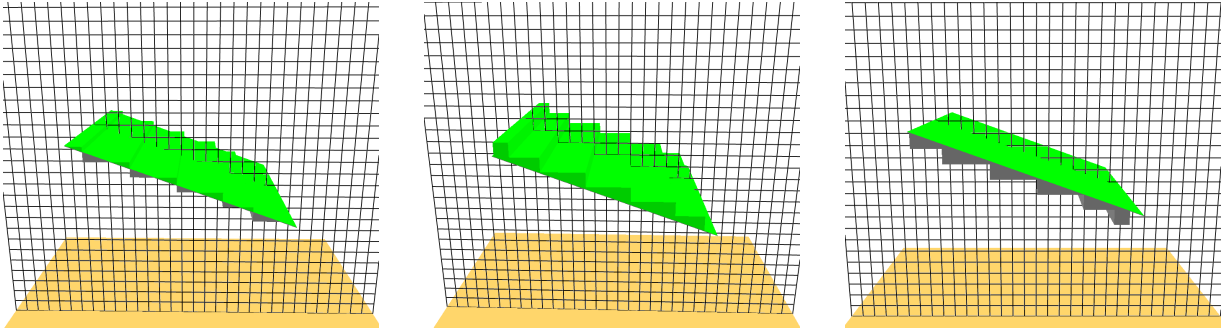


Figure 7.10: Handling thin geometry: (left) Cell blocking on either side; (center) `CELL_BLOCK_IOR= 3` ; (right) `CELL_BLOCK_IOR=-3` .

7.3.10 Handling Split Cells and Thin Geometries

One current limitation of the unstructured geometry algorithm is that a Cartesian cell cannot be split by a thin geometry or corner, potentially creating two background pressure zones within one Cartesian cell. By default, if a cell is split by a thin geometry, FDS will fill in the smaller of the two disjoint volumes. However, it may be desired to leave one side perfectly smooth while allowing the other to absorb the geometry change. For example, you want the inside wall of a tunnel or ceiling to be smooth but you are not as concerned with the outer portion for your problem. In this scenario, you can control which side of the geometry gets filled by setting either `CELL_BLOCK_IOR` or `CELL_BLOCK_ORIENTATION` on the `GEOM` line. The former is an integer indicating a Cartesian axis while the latter allows the specification of a vector in any direction. An example of how this works is shown in Fig. 7.10. Currently, this functionality is limited to geometries where `IOR` can define the direction. For example, this would not work for a hollow sphere. For terrain geometries the default is `CELL_BLOCK_IOR=-3`.

7.3.11 GEOM Limitations

- Currently no control logic is available on `GEOM` namelists. That is, adding or removing a `GEOM` is not possible.
- The `MULT` construct cannot currently be applied to geometries.
- Thin geometries and corners cannot split cells, one side must be blocked. See Sec. 7.3.10.
- Backing external is allowed only on a single mesh. If the back surface is on another mesh the backing will be set to `VOID` (quietly).
- A `VENT` cannot be applied to a `GEOM`. Surface properties must be applied face by face via `SURF_ID`.
- No `VENT` implies `HVAC` cannot be applied to a `GEOM` surface. This is on our priority list and will be available in the future.
- Imposing a tangential velocity in geometries cannot be done. `NO_SLIP` conditions assume a zero tangential velocity. Allowing this capability is also a development priority.
- Neither 3D heat transfer nor `BURN_AWAY` are available in `GEOM`.
- The `HOLE` namelist does not apply to `GEOM`.

- Subgrid geometries (smaller than a grid cell) are not recommended. They are currently not included in the treatment of momentum equations. Similarly, gaps should be defined over a cell size thick.
- Using triangular faces that are much smaller than the Cartesian grid cell is not recommended. Each face will be assigned boundary conditions at unnecessary resolution compared to Cartesian cells.
- Particles will interact with the `GEOM`, but will not flow underneath a geometry due to surface tension. That is, if particles are falling on top of a sphere, once they reach the equator they fall directly to the floor. They will behave as pellets, not droplets.
- Be cautious with soot deposition. While the functionality for deposition exists with geometry, it has not undergone thorough testing.

7.3.12 The Big Picture, Run Time, and Accuracy

Big picture The long-term goal for complex geometry in FDS is to allow for relatively seamless workflow with Building Information Management (BIM) models and complex CAD geometries. However, the past several years have been spent developing a formulation that is conservative and stable. In so doing, for the time being we have had to abandon the immersed boundary method and move to a cut cell scheme that requires an unstructured pressure solver (ULMAT; see Sec. 21.1).

Run time When using complex geometry you may notice slower runtimes for two reasons. First, the setup time will be noticeably slower, especially if you are not importing geometry from binary (see Sec. 7.3.9). This is inevitable, because identifying and processing cutcells is time-consuming. Second, the use of ULMAT, while more accurate at solid boundaries, is slower than the FFT pressure solver (per iteration). Further, ULMAT gets progressively slower as the size of the mesh increases. Therefore, for larger cases using ULMAT it is important to break the domain up into meshes with less than 150,000 cells (roughly 53^3).

Accuracy The rationale for using curvilinear geometry should be that the accuracy of the Cartesian geometry is not sufficient. For example, you may have a sloped ceiling or some detail of a room that requires higher fidelity. When higher accuracy is required, this usually also means that better grid resolution is required. Keep the fidelity of the geometry and the Cartesian grid consistent with each other. Do not expect a highly detailed geometry to give improved results if the Cartesian grid is disproportionately coarse.

7.4 Applying Surface Properties

Whereas the `OBST` group is used to specify obstructions within the computational domain, the `VENT` group (Table 23.37) is used to prescribe planes adjacent to obstructions or external walls. Note that the label `VENT` is used for historical reasons—this group of parameters has evolved well beyond its initial role as simply allowing for air to be blown into, or sucked out of, the computational domain. See Sec. 10.1 for a discussion of ventilation strategies that employ the `VENT` parameters.

7.4.1 Basics

Vents are specified in a similar manner to obstructions, with the sextuplet `XB` denoting a plane abutting a solid surface. Two of the six coordinates must be the same, denoting a plane as opposed to a solid. Note that only one `VENT` may be specified for any given solid surface. If `VENTs` overlap, FDS will output a warning message and ignore redundant `VENT` lines.

The term `VENT` is somewhat misleading. Taken literally, a `VENT` can be used to model components of the ventilation system in a building, like a diffuser or a return. In these cases, the `VENT` coordinates form a plane on a solid surface forming the boundary of the duct. No holes need to be created through the solid; it is assumed that air is pushed out of or sucked into duct work within the wall. Less literally, a `VENT` is used simply as a means of applying a particular boundary condition to a rectangular patch on a solid surface. A fire, for example, is usually created by first generating a solid obstruction via an `OBST` line, and then specifying a `VENT` somewhere on one of the faces of the solid with a `SURF_ID` with the characteristics of the thermal and combustion properties of the fuel. For example, the lines

```
&OBST XB=0.0,5.0,2.0,3.0,0.0,4.0, SURF_ID='big block' /  
&VENT XB=1.0,2.0,2.0,2.0,1.0,3.0, SURF_ID='hot patch' /
```

specify a large obstruction (with the properties given elsewhere in the file under the name 'big block') with a “patch” applied to one of its faces with alternative properties under the name 'hot patch'. This latter surface property need not actually be a “vent,” like a supply or return duct, but rather just a patch with different boundary conditions than those assumed for the obstruction. Note that the surface properties of a `VENT` over-ride those of the underlying obstruction.

A `VENT` must always be attached to a solid obstruction. Sometimes, a portion of a `VENT` might extend beyond a solid, in which case it is either ignored or will trigger an error message. Only the portion of the `VENT` that abuts the solid surface is considered. Thus, if the `VENT` is given a `SURF_ID` that specifies a value of `HRRPUA` (Heat Release Rate Per Unit Area), this rate will only be applied to the portion of the `VENT` for which there is a solid beneath.

Also, a `VENT` derives its orientation by having a solid cell on one side and a gas cell on the other. Do not extend a `VENT` to have multiple orientations.

An easy way to specify an entire external wall is to replace `XB` with `MB` (Mesh Boundary), a character string whose value is one of the following: 'XMAX', 'XMIN', 'YMAX', 'YMIN', 'ZMAX' or 'ZMIN' denoting the planes $x = XMAX$, $x = XMIN$, $y = YMAX$, $y = YMIN$, $z = ZMAX$ or $z = ZMIN$, respectively. Like an obstruction, the boundary condition index of a vent is specified with `SURF_ID`, indicating which of the listed `SURF` lines to apply. If the default boundary condition is desired, then `SURF_ID` need not be set.

Be careful when using the `MB` shortcut when doing a multiple mesh simulation; that is, when more than one rectangular mesh is used. The plane designated by the character string `MB` may be mistakenly applied to more than one mesh, possibly leading to confusion about whether a plane is a solid wall or an open boundary. Check the geometry in Smokeview to assure that the `VENTs` are properly specified. Use color as much as possible to double-check the set-up. More detail on color in Sec. 7.5 and Table 7.1. Also, the parameter `OUTLINE=T` on the `VENT` line causes the `VENT` to be drawn as an outline in Smokeview.

A safer option for multi-mesh calculations might be to use `DB` (Domain Boundary), which takes the same options as `MB` but applies the `SURF_ID` to the domain boundary.

7.4.2 Special Vents

There are four reserved `SURF_ID`'s that may be applied to a `VENT` – `'OPEN'`, `'MIRROR'`, `'PERIODIC'`, and `HVAC`. The term *reserved* means that these `SURF_IDS` should not be explicitly defined by you. Their properties are predefined.

Open Vents

The first special `VENT` is invoked by the parameter `SURF_ID='OPEN'`. This is used only if the `VENT` is applied to the exterior boundary of the computational domain, where it denotes a passive opening to the outside. By default, FDS assumes that the exterior boundary of the computational domain (the `XBS` on the `MESH` line) is a solid wall. To create a totally or partially open domain, use `OPEN` vents on the exterior mesh boundaries. It is sometimes convenient to specify doors or windows that open out to the exterior of the computational domain by simply specifying it to be `OPEN`. However, keep in mind that the pressure boundary condition on such an opening is imperfect, and it is recommended that if the flow through the doorway or window is important, you should extend the domain a few meters rather than use an `OPEN` boundary. You would still have to use the `OPEN` boundary to open up one or more sides of the computational domain, but these openings would be far enough away from the modeled door or window that they would not affect the flow pattern.

By default, it is assumed that ambient conditions exist beyond the `'OPEN'` vent. However, in some cases, you may want to alter this assumption, for example, the temperature. If you assume a temperature other than ambient, specify `TMP_EXTERIOR` along with `SURF_ID='OPEN'`. You can modify the time history of this parameter using a ramp function, `TMP_EXTERIOR_RAMP` (see Section 11.1 for details about time/temperature ramps). Use this option cautiously – in many situations if you want to describe the exterior of a building, it is better to include the exterior explicitly in your calculation because the flow in and out of the doors and windows will be more naturally captured. See Sec. 16.5.1 for more details. If you want to specify a non-ambient pressure at the `OPEN` boundary, see Sec. 10.4.

The `OPEN` pressure boundary condition is most stable for flows that are predominantly normal to the vent, either mostly in or mostly out. This is because the prescribed pressure at an `OPEN` boundary is ill-conditioned (a small perturbation to the input may lead to large change in the output) if the flow is parallel to the vent. Suppose, for example, that an outdoor flow is 10 m/s in the x direction and ± 0.001 m/s in the z direction with an `OPEN` top boundary. The kinetic energy of this flow is roughly $k = 50 \text{ m}^2/\text{s}^2$. When the vertical velocity is positive (+0.001 m/s) then the prescribed boundary condition for the stagnation pressure is set to $H = k = 50 \text{ m}^2/\text{s}^2$. But when the vertical velocity is negative (-0.001 m/s) then $H = 0$ (see [1]). For this reason, `OPEN` vents should be used with care in outdoor applications. See Section 16.2 for an alternative approach.

Vents to the outside of the computational domain (`OPEN` vents) *can* be opened or closed during a simulation. It is best done by creating or removing a thin obstruction that covers the `OPEN VENT`. See Sec. 18.4.2 for details.

Mirror Vents

A `VENT` with `SURF_ID='MIRROR'` denotes a symmetry plane. Usually, a `MIRROR` spans an entire face of the computational domain, essentially doubling the size of the domain with the `MIRROR` acting as a plane of symmetry. The flow on the opposite side of the `MIRROR` is exactly reversed². From a numerical point of

²Note that the mirror image of a scene is *not* shown in Smokeview.

view, a `MIRROR` is a no-flux, free-slip boundary. As with `OPEN`, a `MIRROR` can only be prescribed at an exterior boundary of the computational domain. Often, `OPEN` or `MIRROR VENTS` are prescribed along an entire side of the computational domain, in which case the “`MB`” notation is handy.

In conventional RANS (Reynolds-Averaged Navier-Stokes) models, symmetry boundaries are often used as a way of saving on computation time. However, because FDS is an LES (Large Eddy Simulation) model, the use of symmetry boundaries should be considered carefully. The reason for this is that an LES model does not compute a time-averaged solution of the N-S equations. In other words, for a RANS model, a fire plume is represented as an axially-symmetric flow field because that is what you would expect if you time-averaged the actual flow field over a sufficient amount of time. Thus, for a RANS model, a symmetry boundary along the plume centerline is appropriate. In an LES model, however, there is no time-averaging built into the equations, and there is no time-averaged, symmetric solution. Putting a `MIRROR` boundary along the centerline of a fire plume will change its dynamics entirely. It will produce something very much like the flow field of a fire that is adjacent to a vertical wall. For this reason, a `MIRROR` boundary condition is not recommended along the centerline of a turbulent fire plume. If the fire or burner is very small, and the flow is laminar, then the `MIRROR` boundary condition makes sense. In fact, in 2-D calculations, `MIRROR` boundary conditions are employed in the third coordinate direction (this is done automatically, you need not specify it explicitly).

Periodic Vents

A `VENT` with `SURF_ID='PERIODIC'` may be used in combination with another periodic vent on the opposite side of the domain. If the domain consists of only a single mesh, the lines:

```
&VENT MB='XMIN', SURF_ID='PERIODIC' /  
&VENT MB='XMAX', SURF_ID='PERIODIC' /
```

designate that the simulation is periodic in the x -direction. For multi-mesh domains where `PERIODIC` boundary conditions are applied, the entire domain must be a single large block and the `VENT` planes must be specified using `PBX`, `PBY`, or `PBZ`. If $x_{\min} = 0$ and $x_{\max} = 1$, for example, use

```
&VENT PBX=0, SURF_ID='PERIODIC' /  
&VENT PBX=1, SURF_ID='PERIODIC' /
```

By default, in order to handle the most general case of a periodic domain with multiple meshes (imagine a periodic channel divided into several meshes), FDS treats the pressure boundary condition as what we call an “interpolated boundary.” This means that the matrix for the pressure Poisson equation is arranged for Dirichlet boundary conditions (the value of the solution is specified at the boundary). This can lead to small errors in the solution and sometimes it is desirable to use the true periodic matrix for the Poisson equation. But with the current solver (Fishpak) this is only possible for a single mesh. If you want to implement true periodic boundaries for a single mesh case, set the appropriate `FISHPAK_BC` value to zero on the `PRES` line. For example,

```
&PRES FISHPAK_BC(1:3)=0,0,0 /
```

Periodic vents may not be used to connect offset vents or vents in different coordinate directions. For such cases, you must employ HVAC capabilities (see Sec. 10.2).

HVAC Vents

A VENT with SURF_ID='HVAC' denotes that the vent is connected to an HVAC system. See Sec. 10.2 for a description of inputs for HVAC systems.

Circular Vents

Circular or semi-circular vents may be specified as the intersection of a rectangle with coordinates XB and a circle with center XYZ and radius RADIUS. The rectangular surface cells that are assigned the corresponding SURF_ID will be those whose centroid falls within the intersection. In the example case called Fires/circular_burner.fds, the following two lines create a circular vent that is 1 m in diameter and flows propane gas at a rate of 0.02 kg/m²/s:

```
&SURF ID='BURNER', MASS_FLUX(1)=0.02, SPEC_ID(1)='PROPANE', TAU_MF(1)=0.01 /  
&VENT XB=-0.6,0.6,-0.6,0.6,0,0, XYZ=0,0,0, RADIUS=0.5, SURF_ID='BURNER',  
      SPREAD_RATE=0.05/
```

The XB coordinates designate the orientation of the vent. In this case, the extent of the area specified by XB is large enough to contain the entire circle. Note also in this example that the parameter SPREAD_RATE causes the fire to spread outward at a rate of 0.05 m/s. The mass flux of propane through the vent is plotted in Fig. 7.11. Notice that the mass flux increases following a “t-squared” profile. This is what is expected of a fire which spreads radially at a linear rate. In this case, the fire reaches the RADIUS of the circle in 10 s, as expected. Note also that the parameter TAU_MF indicates that the fuel should ramp up quickly once the flame front reaches a given grid cell. In other words, TAU_MF controls the local ramp-up of fuel; the SPREAD_RATE controls the global ramp-up. Following the ramp-up, the fuel flows at a rate equal to the area of the circle times the mass flux of fuel per unit area. Even if the circle is crudely resolved on a coarse grid, the fuel flow rate will be adjusted to produce the desired value governed by the circular vent.

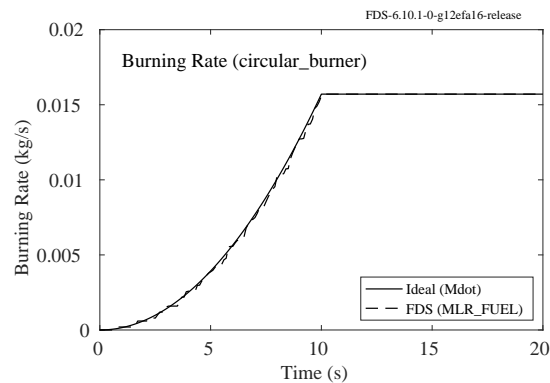


Figure 7.11: Results of the circular_burner test case.

7.4.3 Controlling Vents

VENT functionality can be controlled in some cases using “devices” and “controls,” specified via a DEVC_ID or a CTRL_ID. See Sec. 18.4.2 for details.

7.4.4 Trouble-Shooting Vents

Unlike most of the entries in the input file, the order that you specify `VENTS` can be important. There might be situations where it is convenient to position one `VENT` atop another. For example, suppose you want to designate the ceiling of a compartment to have a particular set of surface properties, and you designate the entire ceiling to have the appropriate `SURF_ID`. Then, you want to designate a smaller patch on the ceiling to have another set of surface properties, like an air supply. In this case, you must designate the supply `VENT` *first* because for that area of the ceiling, FDS will ignore the ceiling properties and apply the supply properties. FDS processes the first `VENT`, not the second as it did in versions prior to FDS 5. Now, the rule for `VENTS` is “first come, first served.” Keep in mind, however, that the second `VENT` is not rejected entirely – only where there is overlap. FDS will also print out a warning to the screen (or to standard error) saying which `VENT` has priority. Also, be careful if any of the `VENTS` are applying `OPEN` boundary conditions – `OPEN` boundaries disable pressure `ZONES`, which will change the solution of the problem.

Smokeview can help identify where two `VENTS` overlap, assuming each has a unique `COLOR`. Because Smokeview draws `VENTS` on top of each other, areas of overlap will have a grainy, awkward appearance that changes pattern as you move the scene. In situations where you desire the overlap for the sake of convenience, you might want to slightly adjust the coordinates of the preferred `VENT` so that it is slightly offset from the solid surface. Make the offset less than about a tenth of a cell dimension so that FDS snaps it to its desired location. Then, by toggling the “q” key in Smokeview, you can eliminate the grainy color overlap by showing the `VENT` exactly where you specified it, as opposed to where FDS repositioned it. This trick also works where the faces of two obstructions overlap.

If an error message appears requesting that the orientation of a vent be specified, first check to make sure that the vent is a plane. If the vent is a plane, then the orientation can be forced by specifying the parameter `IOR`. If the normal direction of the `VENT` is in the positive x direction, set `IOR=1`. If the normal direction is in the negative x direction, set `IOR=-1`. For the y and z direction, use the number 2 and 3, respectively. Setting `IOR` may sometimes solve the problem, but it is more likely that if there is an error message about orientation, then the `VENT` is buried within a solid obstruction, in which case the program cannot determine the direction in which the `VENT` is facing.

7.5 Coloring Obstructions, Geometry, Vents, Surfaces, and Meshes

Assigning objects a meaningful color or pattern is useful when visualizing the results of a simulation. There are two ways to do this in FDS. Either assign a single color, or assign a texture map, which is essentially an image of your choosing.

7.5.1 Colors

Colors for many items within FDS can be prescribed in two ways; a triplet of integer color values, `RGB`, or a character string, `COLOR`. The three `RGB` integers range from 0 to 255, indicating the amount of Red, Green and Blue that make up the color. If you define the `COLOR` by name, it is important that you type the name *exactly* as it is listed in the color tables. Color parameters can be specified on a `SURF` line, in which case all surfaces of that type will have that color, or color parameters can be applied directly to obstructions immersed geometries, or vents. For example, the lines:

```
&SURF ID='UPHOLSTERY', ..., RGB=0,255,0 /  
&OBST XB=..., COLOR='BLUE' /  
&GEOM GEOM_ID=..., COLOR='BLUE' /
```

will color all `UPHOLSTERY` green and this particular obstruction or geometry blue.

Table 7.1 provides a small sampling of `RGB` values and `COLOR` names for a variety of colors³. Assigning unique colors to surfaces via the `SURF` line is very useful for determining if the desired surface properties have been assigned throughout the geometry.

Obstructions, geometries and vents may be colored individually, over-riding the color designated by the `SURF` line. The special case `COLOR='INVISIBLE'` causes the vent, geometry or obstruction not to be drawn by Smokeview. Another special case `COLOR='RAINBOW'` causes the color of the vent, obstruction or mesh to be randomly selected from the full range of `RGB` values; this can be useful when using the `MULT` namelist group and want to differentiate between the multiplied obstruction, vent or mesh.

7.5.2 Texture Maps

There are various ways of prescribing the color of various objects within the computational domain, but there is also a way of pasting images onto the obstructions for the purpose of making the Smokeview images more realistic. This technique is known as “texture mapping.” For example, to apply a wood paneling image to a wall, add to the `SURF` line defining the physical properties of the paneling the line:

```
&SURF ID='wood paneling', ..., TEXTURE_MAP='paneling.jpg', TEXTURE_WIDTH=1.,  
TEXTURE_HEIGHT=2. /
```

Assuming that a JPEG file called `paneling.jpg` exists in the working directory, Smokeview loads and displays the image wherever the paneling is used. PNG images may also be used for texture maps. Note that the image does not appear when Smokeview is first invoked. It is an option controlled by the `Show/Hide` menu. The parameters `TEXTURE_WIDTH` and `TEXTURE_HEIGHT` are the physical dimensions of the image. In this case, the JPEG image is of a 1 m wide by 2 m high piece of paneling. Smokeview replicates the image as often as necessary to make it appear that the paneling is applied where desired. Consider carefully how the image repeats itself when applied in a scene. If the image has no obvious pattern, there is no problem with the image being repeated. If the image has an obvious direction, the real triplet `TEXTURE_ORIGIN` should be added to the `VENT` or `OBST` line to which a texture map should be applied. For example,

³A complete listing of all 500+ colors can be found by searching the FDS source code file `data.f90`.

Table 7.1: A sample of color definitions.

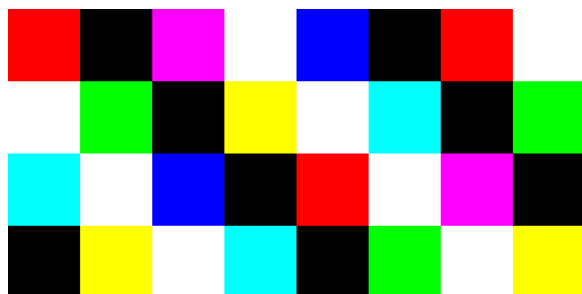
Name		R	G	B	Name		R	G	B
AQUAMARINE		127	255	212	MAROON		128	0	0
ANTIQUE WHITE		250	235	215	MELON		227	168	105
BEIGE		245	245	220	MIDNIGHT BLUE		25	25	112
BLACK		0	0	0	MINT		189	252	201
BLUE		0	0	255	NAVY		0	0	128
BLUE VIOLET		138	43	226	OLIVE		128	128	0
BRICK		156	102	31	OLIVE DRAB		107	142	35
BROWN		165	42	42	ORANGE		255	128	0
BURNT SIENNA		138	54	15	ORANGE RED		255	69	0
BURNT UMBER		138	51	36	ORCHID		218	112	214
CADET BLUE		95	158	160	PINK		255	192	203
CHOCOLATE		210	105	30	POWDER BLUE		176	224	230
COBALT		61	89	171	PURPLE		128	0	128
CORAL		255	127	80	RASPBERRY		135	38	87
CYAN		0	255	255	RED		255	0	0
DIM GRAY		105	105	105	ROYAL BLUE		65	105	225
EMERALD GREEN		0	201	87	SALMON		250	128	114
FIREBRICK		178	34	34	SANDY BROWN		244	164	96
FLESH		255	125	64	SEA GREEN		84	255	159
FOREST GREEN		34	139	34	SEPIA		94	38	18
GOLD		255	215	0	SIENNA		160	82	45
GOLDENROD		218	165	32	SILVER		192	192	192
GRAY		128	128	128	SKY BLUE		135	206	235
GREEN		0	255	0	SLATEBLUE		106	90	205
GREEN YELLOW		173	255	47	SLATE GRAY		112	128	144
HONEYDEW		240	255	240	SPRING GREEN		0	255	127
HOT PINK		255	105	180	STEEL BLUE		70	130	180
INDIAN RED		205	92	92	TAN		210	180	140
INDIGO		75	0	130	TEAL		0	128	128
IVORY		255	255	240	THISTLE		216	191	216
IVORY BLACK		41	36	33	TOMATO		255	99	71
KELLY GREEN		0	128	0	TURQUOISE		64	224	208
KHAKI		240	230	140	VIOLET		238	130	238
LAVENDER		230	230	250	VIOLET RED		208	32	144
LIME GREEN		50	205	50	WHITE		255	255	255
MAGENTA		255	0	255	YELLOW		255	255	0

```
&OBST XB=1.,2.,3.,4.,5.,7., SURF_ID='wood paneling', TEXTURE_ORIGIN=1.,3.,5. /
```

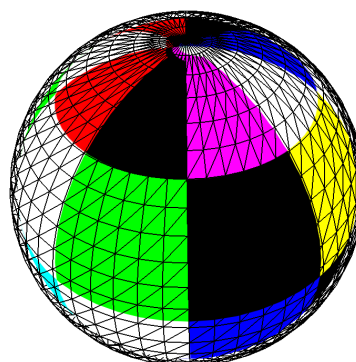
applies paneling to an obstruction whose dimensions are 1 m by 1 m by 2 m, such that the image of the paneling is positioned at the point (1,3,5). The default value of `TEXTURE_ORIGIN` is (0,0,0), and the global default can be changed by added a `TEXTURE_ORIGIN` statement to the `MISC` line.

For geometries that are spheres, the parameter `TEXTURE_MAPPING='SPHERICAL'` on the `GEOM` line lets smokeview know to wrap the texture around the sphere rather than to lay it flat from above. The following FDS input lines from the example test case `geom_texture3a.fds` defines a sphere using the image `sphere_cover_03.png` as a texture map (this image is found in the smokeview installation directory). Figure 7.12 shows a texture mapped sphere using these input lines along with rectangular texture map that was applied.

```
&SURF ID='surf1'
  TEXTURE_MAP='sphere_cover_03.png',COLOR='BLUE',
  TEXTURE_WIDTH=1.0,TEXTURE_HEIGHT=1.0,TEXTURE_ORIGIN=0.0,0.0,0.0,
  TEXTURE_MAPPING='SPHERICAL' /
&MOVE ID='SPHERE_ROT', AXIS=0.,0.,1., ROTATION_ANGLE=270. /
&GEOM ID='sphere1',SURF_ID='surf1',SPHERE_RADIUS=0.5,
N_LAT=25,N_LONG=50,SPHERE_ORIGIN=0.0,0.0,0.0,MOVE_ID='SPHERE_ROT'
```



texture map



texture mapped sphere

Figure 7.12: Texture mapped sphere.

7.6 Repeated Objects: The MULT Namelist Group (Table 23.20)

Sometimes obstructions, holes and vents are repeated over and over in the input file. This can be tedious to create and make the input file hard to read. However, if a particular set of objects repeats itself in a regular pattern, you can use a utility known as a multiplier. If you want to repeat an obstruction, for example, create a line in the input file as follows:

```
&MULT ID='m1', DX=1.2, DY=2.4, I_LOWER=-2, I_UPPER=3, J_LOWER=0, J_UPPER=5 /
&OBST XB=x1,x2,y1,y2,z1,z2, MULT_ID='m1' /
```

This has the effect of making an array of obstructions according to the following formulae:

$$\begin{aligned}x1' &= x1 + DX0 + i DX & ; & \quad I_LOWER \leq i \leq I_UPPER \\x2' &= x2 + DX0 + i DX & ; & \quad I_LOWER \leq i \leq I_UPPER \\y1' &= y1 + DY0 + j DY & ; & \quad J_LOWER \leq j \leq J_UPPER \\y2' &= y2 + DY0 + j DY & ; & \quad J_LOWER \leq j \leq J_UPPER \\z1' &= z1 + DZ0 + k DZ & ; & \quad K_LOWER \leq k \leq K_UPPER \\z2' &= z2 + DZ0 + k DZ & ; & \quad K_LOWER \leq k \leq K_UPPER\end{aligned}$$

In situations where the position of the obstruction needs shifting prior to the multiplication, use the parameters DX0, DY0, and DZ0.

A variation of this idea is to replace the parameters, DX, DY, and DZ, with a sextuplet called DXB. The six entries in DXB increment the respective values of the obstruction coordinates given by XB. For example, the x coordinates are transformed as follows:

$$\begin{aligned}x1' &= x1 + DX0 + n DXB(1) & ; & \quad N_LOWER \leq n \leq N_UPPER \\x2' &= x2 + DX0 + n DXB(2) & ; & \quad N_LOWER \leq n \leq N_UPPER\end{aligned}$$

N_LOWER and N_UPPER is used to denote the range of N. This more flexible input scheme allows you to create, for example, a slanted roof in which the individual roof segments shorten as they ascend to the top. This feature is demonstrated by the following short input file that creates a hollowed out pyramid using the four perimeter obstructions that form the outline of its base:

```
&HEAD CHID='pyramid', TITLE='Simple demo of multiplier function' /
&MESH IJK=100,100,100, XB=0.0,1.0,0.0,1.0,0.0,1.0 /
&TIME T_END=0. /
&MULT ID='south', DXB=0.01,-.01,0.01,0.01,0.01,0.01, N_LOWER=0, N_UPPER=39 /
&MULT ID='north', DXB=0.01,-.01,-.01,-.01,0.01,0.01, N_LOWER=0, N_UPPER=39 /
&MULT ID='east', DXB=-.01,-.01,0.01,-.01,0.01,0.01, N_LOWER=0, N_UPPER=39 /
&MULT ID='west', DXB=0.01,0.01,0.01,-.01,0.01,0.01, N_LOWER=0, N_UPPER=39 /
&OBST XB=0.10,0.90,0.10,0.11,0.10,0.11, MULT_ID='south', COLOR='RED' /
&OBST XB=0.10,0.90,0.89,0.90,0.10,0.11, MULT_ID='north', COLOR='BLUE' /
&OBST XB=0.10,0.11,0.11,0.89,0.10,0.11, MULT_ID='west', COLOR='GREEN' /
&OBST XB=0.89,0.90,0.11,0.89,0.10,0.11, MULT_ID='east', COLOR='CYAN' /
&MULT ID='holes', DX=0.15, DZ=0.1, I_UPPER=1, K_UPPER=1 /
&HOLE XB=0.40,0.45,0.00,1.00,0.15,0.20, MULT_ID='holes' /
&TAIL /
```

The end result of this input file is to create a pyramid by repeating long, rectangular obstructions at the base of each face in a stair-step pattern. Note in this case the use of N_LOWER and N_UPPER which automatically cause FDS to repeat the obstructions in sequence rather than as an array.

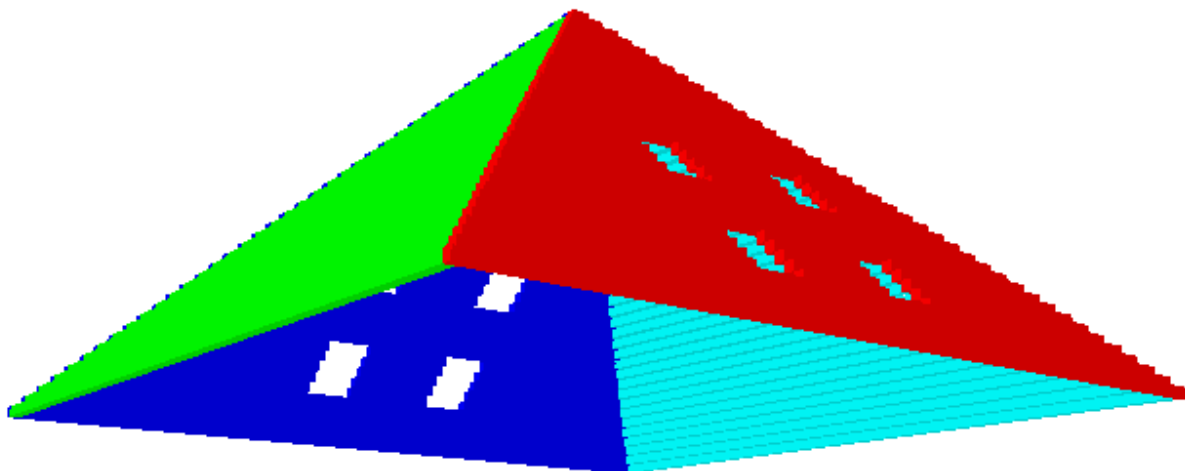


Figure 7.13: An example of the multiplier function.

Note that the `MULTIPLICATION` functionality works for `MESH`, `OBST`, `HOLE`, `VENT`, and `INIT` lines. For a `MESH`, it only applies to the bounds (`XB`) of the mesh, not the number of cells.

Note also that if a `COLOR` is specified on a line that includes a `MULT_ID`, this color will be applied to all replicates of the object. However, you can set `COLOR='RAINBOW'` which will instruct FDS to randomly choose a color for each replicate object.

7.6.1 Using `MULT` for Mesh Refinement

Note that objects and meshes in the `MULT` array may be skipped using `I_LOWER_SKIP` and `I_UPPER_SKIP`, etc. The i from the equation above is skipped within this interval. As an example of how this may be useful, consider the multi-mesh case below with refinement of the interior meshes. The resulting mesh arrangement is shown in Fig. 7.14.

```
&MULT ID='interior fine mesh array',
  DX=1.0,I_LOWER=1,I_UPPER=2,
  DZ=1.0,K_LOWER=1,K_UPPER=2 /

&MULT ID='exterior coarse mesh array',
  DX=1.0,I_LOWER=0,I_LOWER_SKIP=1,I_UPPER_SKIP=2,I_UPPER=3,
  DZ=1.0,K_LOWER=0,K_LOWER_SKIP=1,K_UPPER_SKIP=2,K_UPPER=3 /

&MESH IJK=8,1,8, XB=0.0,1.0,-0.5,0.5,0.0,1.0, MULT_ID='interior fine mesh array' /
&MESH IJK=4,1,4, XB=0.0,1.0,-0.5,0.5,0.0,1.0, MULT_ID='exterior coarse mesh array' /
```

7.6.2 Using `MULT` to make shapes out of obstructions

Block obstructions made using an `OBST` line are constrained to the box defined by `XB`. However, `MULT` can be used to create an array of obstructions—each of which can be as small as a single grid cell—that can be manipulated into one of four basic shapes: sphere, cylinder, cone or box. The flow obstruction is created from the intersection of the `OBST` array, created using a `MULT_ID`, and the `SHAPE` defined on the `OBST` line.

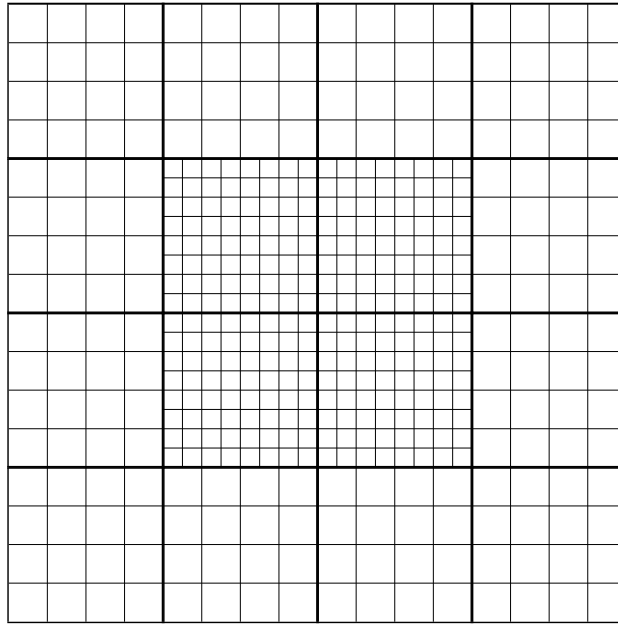


Figure 7.14: Using MULT for mesh refinement.

Note that this method may be memory intensive since each grid cell can be its own OBST, similar to the way OBST are decomposed when using BURN_AWAY for pyrolysis.

The first step in carving out the shape is to create a MULT line that defines the replication of the OBST. Then a SHAPE is entered on the OBST line together with whatever parameters are required; parameters are listed in Table 7.2.

Table 7.2: OBST SHAPE parameters.

SHAPE	Parameters (default values in Table 23.21)
'SPHERE'	XYZ, RADIUS
'CYLINDER'	XYZ, RADIUS, HEIGHT, ORIENTATION
'CONE'	XYZ, RADIUS, HEIGHT, ORIENTATION
'BOX'	XYZ, LENGTH, WIDTH, HEIGHT, ORIENTATION, THETA

An example of a sphere is given below. By default, the center of the sphere is at (0,0,0), so only the RADIUS is required. If the RADIUS is larger than the half-width the OBST array, this is not a problem, but the end result will not be a sphere. If the RADIUS is smaller than the half-width, this is also permissible, but it is inefficient.

```
&MESH IJK=50,50,50, XB=-0.125,0.125,-0.125,0.125,-0.125,0.125/
&SURF ID='shape', COLOR='ORANGE'/
&MULT ID='cube array', DX=0.005,DY=0.005,DZ=0.005, I_UPPER=41,J_UPPER=41,K_UPPER=41/
&OBST XB=-0.105,-0.100,-0.105,-0.100,-0.105,-0.100
MULT_ID='cube array'
SURF_ID='shape'
SHAPE='SPHERE'
RADIUS=0.1/
```

To create a cylinder we need a height and orientation in addition to the radius. The position of the center of the bottom face of the cylinder XYZ defaults to (0,0,0). An example of a vertically oriented (default) cylinder is shown below in Fig. 7.16. Note that the magnitude of the orientation vector is not important; the

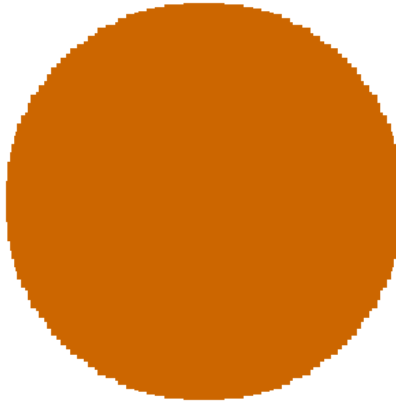


Figure 7.15: Creating an OBST sphere using MULT and SHAPE.

vector is only used to specify the direction.

A word of caution about precise area adjustments: Changing the `ORIENTATION` from the default (0,0,1) or using `SHAPE` with multiple meshes voids the area adjustment algorithm. As a result, FDS cannot achieve the exact mass flux out of the surface as prescribed by the `SURF` line and the precise area given by the geometry parameters for the `SHAPE` on the `OBST` line. The mass flux FDS generates will be determined by `SURF` line and the size of the grid cells the `OBST` snaps to. If either reorientation or a multiple mesh calculation is required, first run a sample case to see what total flow rate the surface is achieving (usually found in the `CHID_hrr.csv` file). Then manually adjust your mass flux (or `HRRPUA`) on the appropriate `SURF` line.

```
&SURF ID='shape', COLOR='DARK GRAY'/
&MULT ID='cube array', DX=0.005,DY=0.005,DZ=0.005, I_UPPER=41,J_UPPER=41,K_UPPER=41/
&OBST XB=-0.105,-0.100,-0.105,-0.100,-0.105,-0.100
      MULT_ID='cube array'
      SURF_ID='shape'
      SHAPE='CYLINDER'
      RADIUS=0.05
      HEIGHT=0.2
      XYZ=0,0,-0.1
      ORIENTATION=0,0,1/
```

A cone needs basically the same input parameters as a cylinder. The position of the center of the bottom face is specified using `XYZ` and the `RADIUS` of the bottom face, the `HEIGHT`, and the `ORIENTATION` must be specified. An example is shown below in Fig 7.17.

```
&SURF ID='shape', COLOR='FOREST GREEN'/
&MULT ID='cube array', DX=0.005,DY=0.005,DZ=0.005, I_UPPER=41,J_UPPER=41,K_UPPER=41/
&OBST XB=-0.105,-0.100,-0.105,-0.100,-0.105,-0.100
      MULT_ID='cube array'
      SURF_ID='shape'
      SHAPE='CONE'
      RADIUS=0.05
      HEIGHT=0.2
      XYZ=0,0,-0.1
```



Figure 7.16: Creating an OBST cylinder using MULT and SHAPE.

```
ORIENTATION=0,0,1/
```

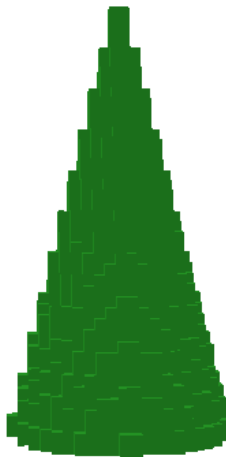


Figure 7.17: Creating an OBST cone using MULT and SHAPE.

Finally, the box shape allows to define rotated cuboids. A box shape is defined locally by its HEIGHT, WIDTH and LENGTH. The location of the box center is provided in XYZ. The angle THETA in degrees specifies a rotation of the box respect to the global z axis, following right hand rule. Subsequently, the ORIENTATION direction vector allows for an orientation change respect to the THETA rotated reference frame. This direction vector changes the box orientation such that the local HEIGHT direction matches it. A simple example is shown Fig 7.18.

```
&SURF ID='shape', COLOR='GRAY'/
&MULT ID='cube array', DX=0.005,DY=0.005,DZ=0.005, I_UPPER=41,J_UPPER=41,K_UPPER=41/
```

```
&OBST XB=-0.105,-0.100,-0.105,-0.100,-0.105,-0.100  
MULT_ID='cube array'  
SURF_ID='shape'  
SHAPE='BOX'  
LENGTH=0.075  
WIDTH=0.025  
HEIGHT=0.05  
XYZ=0,0,0.01  
ORIENTATION=0.5,0,1  
THETA = 45. /
```

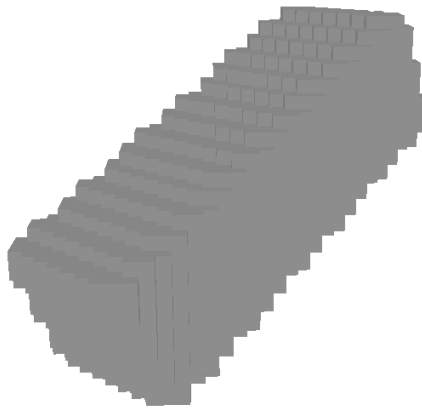


Figure 7.18: Creating an OBST rotated box using MULT and SHAPE.

Chapter 8

Thermal Boundary Conditions

This chapter describes how to specify the thermal properties of solid objects.

8.1 Basics

By default, the outer boundary of the computational domain is assumed to be a solid boundary that is maintained at ambient temperature. The same is true for any obstructions that are added to the scene. To specify the properties of solids, use the namelist group `SURF` (Sec. 7.1). Solids are assumed to consist of layers that can be made of different materials. The properties of each material required are designated via the `MATL` namelist group (Sec. 8.3). These properties indicate how rapidly the materials heat up, and how they burn. Each `MATL` entry in the input file must have an `ID`, or name, so that they may be associated with a particular `SURF` via the parameter `MATL_ID`. For example, the input file entries:

```
&MATL ID='BRICK', CONDUCTIVITY=0.69, SPECIFIC_HEAT=0.84, DENSITY=1600. /  
&SURF ID='BRICK WALL', MATL_ID='BRICK', COLOR='RED', BACKING='EXPOSED',  
      THICKNESS=0.20 /  
&OBST XB=0.1,5.0,1.0,1.2,0.0,1.0, SURF_ID='BRICK WALL' /
```

define a brick wall that is 4.9 m long, 1 m high, and 20 cm thick. Note that the thickness of the wall indicated by the `OBST` line is independent of the `THICKNESS` specified by the `SURF` line. The `OBST` line defines the geometry of the obstruction (i.e., how the obstruction is seen by the flow solver). The `SURF` line defines the heat transfer characteristics of the obstruction (i.e., how the obstruction is seen by the 1D solid phase solver). This allows an obstruction to snap to the local grid but still have the heat transfer solution reflect the actual thickness.

8.2 Surface Temperature and Heat Flux

This section describes how to specify simple thermal boundary conditions. These are often used when there is little or no information about the properties of the solid materials. If the properties of the materials are known, it is better to specify these properties and let the model compute the heat flux to, and temperature of, the walls and other solid surfaces.

8.2.1 Specified Solid Surface Temperature

Typically, the surface temperature of a solid boundary is predicted by FDS using thermal properties specified via the `MATL` namelist group, which are in turn invoked by the `SURF` entry via the character string `MATL_`

ID. However, the input `TMP_FRONT`, the surface temperature in units of °C, can be used to specify a fixed temperature boundary condition:

```
&SURF ID='HOT WALL', COLOR='RED', TMP_FRONT=200. /
```

Note that there is no need to specify a `MATL_ID` or `THICKNESS`. Because the wall is to be maintained at the given temperature, there is no need to say anything about its material composition or thickness.

8.2.2 Convective Heat Transfer Options

This section is labeled as a special topic because normally modifying the convective heat transfer model in FDS is not needed. However, there are special cases for which the default model may not be adequate, and this section describes some options.

Default Convective Heat Transfer Model

In an LES calculation, the convective heat transfer coefficient, h , is based on the larger of the Nusselt (Nu) numbers for free (natural) and forced convection:

$$\dot{q}_c'' = h(T_g - T_w) \quad ; \quad h = \frac{k}{L} \max \left(\text{Nu}_{\text{free}}, \text{Nu}_{\text{forced}}, \frac{2L}{\delta n} \right) \quad (8.1)$$

where T_g is the temperature at the center of the gas phase cell adjacent to the surface, T_w is the surface (wall) temperature, L is a characteristic length, and k is the thermal conductivity of the gas. For planar surfaces, L is taken as 1 m and for spheres and cylinders, L is taken as the diameter, D . The last argument on the right-hand side corresponds to the limit for DNS cases. For Lagrangian particles the assumption is that $D \ll \delta n$ and so the equation for h becomes:

$$h = \frac{k}{L} \max (\text{Nu}_{\text{free}}, \text{Nu}_{\text{forced}}). \quad (8.2)$$

For free (natural) convection, the Nusselt number is a function of the Rayleigh number:

$$\text{Ra} = \frac{2g|T_g - T_w|L^3}{(T_g + T_w)\nu\alpha} \quad ; \quad \nu = \frac{\mu}{\rho} \quad ; \quad \alpha = \frac{k}{\rho c_p} \quad ; \quad \text{Pr} = \frac{\nu}{\alpha} \quad (8.3)$$

The following expressions are simplifications of those given in Ref. [9] under the assumption that $\text{Pr} = 0.7$.

$$\text{Nu}_{\text{free}} = \begin{cases} (0.825 + 0.324\text{Ra}^{1/6})^2 & \text{Vertical plate or cylinder} \\ 0.54\text{Ra}^{1/4} & \text{Horizontal hot plate facing up or cold plate facing down, } \text{Ra} \leq 10^7 \\ 0.15\text{Ra}^{1/3} & \text{Horizontal hot plate facing up or cold plate facing down, } \text{Ra} > 10^7 \\ 0.52\text{Ra}^{1/5} & \text{Horizontal hot plate facing down or cold plate facing up} \\ (0.60 + 0.321\text{Ra}^{1/6})^2 & \text{Horizontal cylinder} \\ 2 + 0.454\text{Ra}^{1/4} & \text{Sphere} \end{cases} \quad (8.4)$$

For forced convection, the Nusselt number takes the form:

$$\text{Nu}_{\text{forced}} = C_0 + (C_1 \text{Re}^m - C_2) \text{Pr}^{1/3} \quad ; \quad \text{Re} = \frac{\rho|\mathbf{u}|L}{\mu} \quad (8.5)$$

The values of the coefficients are given in Table 8.1. Alternatively, specifying custom coefficients by setting the values of `NUSSELT_C0`, `NUSSELT_C1`, `NUSSELT_C2`, and `NUSSELT_M` on the `SURF` line is possible. If using this feature, you must define the value of all four coefficients.

The length scale, L , is specified by `CONVECTION_LENGTH_SCALE` on the `SURF` line. By default, it is 1 m for plates, and the diameter of a sphere or cylinder.

Table 8.1: Coefficients used for forced convection heat transfer correlations [9]

Geometry	C_0	C_1	C_2	m	Re
Flat Plate	0	0.0296	0	0.8	$\leq 10^8$
Cylinder	0	0.989	0	0.330	0.4 – 4
Cylinder	0	0.911	0	0.385	4 – 40
Cylinder	0	0.683	0	0.466	40 – 4000
Cylinder	0	0.193	0	0.618	4000 – 40000
Cylinder	0	0.027	0	0.805	40000 – 400000
Sphere	2	0.6	0	0.5	

Output for Convective Heat Transfer Regime

It may be useful to visualize which convective heat transfer correlation is being exercised to compute the heat transfer coefficient. This can be accomplished by using the solid phase output quantity 'CONVECTIVE HEAT TRANSFER REGIME' on BNDF or DEVC. For wall surfaces (not available for particles), the regime is mapped to an integer value, 1 = NATURAL, 2 = FORCED, 3 = IMPACT, 4 = RESOLVED, which is color coded for boundary visualization.

Specified Convective Heat Transfer Coefficient

To specify the convective heat transfer coefficient, set it to a constant using HEAT_TRANSFER_COEFFICIENT on the SURF line in units of $W/(m^2 K)$ with optional time dependent ramp using RAMP_HEAT_TRANSFER_COEFFICIENT. If the back side of the solid obstruction faces the exterior of the computational domain and the solid conducts heat, the heat transfer coefficient of the back side may be specified using HEAT_TRANSFER_COEFFICIENT_BACK with optional time dependent ramp using RAMP_HEAT_TRANSFER_COEFFICIENT_BACK. This back side condition is appropriate for a SURF line with BACKING='VOID' or if the solid obstruction backs to the exterior of the computational domain.

Impinging Jet Heat Transfer Model

The forced convection heat transfer correlations generally apply to flow parallel to a surface. When the flow is an impinging jet (normal and toward the surface) then the tangential components of velocity are not well resolved at the impingement point and consequently a fictitiously low value of heat transfer coefficient will be predicted if special consideration is not given to the surface. If the surface may be subject to an impinging jet or stagnation point flow (in fire, usually this pertains to ceilings above a fire source), consider using impinging jet heat transfer model, applied on SURF using HEAT_TRANSFER_MODEL='IMPINGING JET'.

The impinging jet model is an extension of the user-specified h that allows you to specify a Gaussian radial profile parameterized by a center point, XYZ(1:3), a width in meters (m), HEAT_TRANSFER_COEFFICIENT_SIGMA, and the peak value, h_0 , using HEAT_TRANSFER_COEFFICIENT, all on the SURF line. You may determine these parameters using correlations in [9], for example. Take note that the correlations usually give the average heat transfer coefficient over an area, \bar{h} . It can be shown that the ratio of the peak to average for a Gaussian profile over area $A = \pi\sigma^2$ is $h_0/\bar{h} \approx 1.3$. An example SURF line is given below.

```
&SURF ID='WALL', HEAT_TRANSFER_MODEL='IMPINGING JET', XYZ=0,0,1, HEAT_TRANSFER_
    COEFFICIENT=50, HEAT_TRANSFER_COEFFICIENT_SIGMA=0.5 /
```

Specifying the Heat Flux at a Solid Surface

Instead of altering the convective heat transfer coefficient, a fixed heat flux may be specified directly. Two methods are available to do this. The first is to specify a `NET_HEAT_FLUX` in units of kW/m^2 . When this is specified FDS will compute the surface temperature required to ensure that the combined radiative and convective heat flux from the surface is equal to the `NET_HEAT_FLUX`. The second method is to specify the `CONVECTIVE_HEAT_FLUX`, in units of kW/m^2 . The radiative flux is then determined based on the `EMISSIVITY` on the `SURF` line and the wall temperature needed to get the desired `CONVECTIVE_HEAT_FLUX`. Setting the `EMISSIVITY` to zero will result in there being only a convective heat flux from a surface. If `NET_HEAT_FLUX` or `CONVECTIVE_HEAT_FLUX` is positive, the wall heats up the surrounding gases. If `NET_HEAT_FLUX` or `CONVECTIVE_HEAT_FLUX` is negative, the wall cools the surrounding gases. You cannot specify `TMP_FRONT` with `NET_HEAT_FLUX` since `NET_HEAT_FLUX` combines radiative and convective flux, and FDS must compute the wall temperature to get the desired flux. Specifying `TMP_FRONT` and `CONVECTIVE_HEAT_FLUX` will result in FDS using the gas temperature adjacent to a wall cell to set the convection heat transfer coefficient to achieve the desired `CONVECTIVE_HEAT_FLUX` for that wall cell. If a droplet lands surface with a specified heat flux, no heat transfer will be predicted between the droplet and the surface. This is because the high heat transfer rates associated with droplets can lead to nonsensical wall temperatures when trying to enforce the specific flux which are then likely to result in runtime errors or numerical instabilities.

Logarithmic Law of the Wall

Near-wall treatments, such as wall models or wall functions, aim to mimic the sudden change from molecular to turbulent transport close to the walls using algebraic formulations without the need of resolving the otherwise computationally expensive region of the near-wall flow-field. The main theory follows dimensional analysis based on the idea that shear at the wall is constant. Accordingly, the non-dimensional velocity u^+ is calculated using a wall function [1].

By analogy, we define the non-dimensional temperature $T^+ \equiv (T_g - T_w)/T_\tau$, where T_g is the gas temperature of the first off-wall grid cell and T_τ is defined with the wall heat flux, \dot{q}_w'' , as $T_\tau = \dot{q}_w''/\rho_w u_\tau c_p$. The local heat transfer coefficient is then obtained from

$$h = \frac{\dot{q}_w''}{T_g - T_w} = \frac{\rho_w c_p u_\tau}{T^+} \quad (8.6)$$

Refer to the FDS Tech Guide [1] for further details of the formulation. To specify this heat transfer model for a particular surface, set `HEAT_TRANSFER_MODEL` equal to 'LOGLAW' on the `SURF` line. Note that the loglaw model is not well-suited for buoyant flows—it requires a well-resolved “wind” near the surface and is therefore mainly applicable to forced convection type flows with high grid resolution.

Blowing Heat Transfer

If a surface is emitting (“blowing”) or removing (“sucking”) gas, the flow normal to the surface disrupts the thermal boundary layer. Blowing tends to decrease the heat transfer coefficient while sucking tends to increase it. Adding `BLOWING=T` to the `SURF` line will account for this effect, except for DNS simulations where empirical heat transfer correlations are not used. When `BLOWING=T`, the heat transfer coefficient is adjusted as follows [10, 11]:

$$h_{\text{blowing}} = \underbrace{\left[\frac{\Phi_h}{\exp(\Phi_h) - 1} \right]}_{\text{BLOWING CORRECTION}} h \quad ; \quad \Phi_h = \frac{\dot{m}'' c_p}{h} \quad (8.7)$$

where h is the unadjusted heat transfer coefficient, \dot{m}'' is the mass flow rate per unit area (positive for blowing), and c_p is the specific heat of the gas.,

8.2.3 Adiabatic Surfaces

A common boundary condition for diagnostic simulations is to force all surfaces to be perfectly insulated, that is, there is no net heat transfer (radiative and convective) from the gas to the solid. For these cases, specify `ADIABATIC=T` on the `SURF` line. FDS will compute a wall temperature so that the sum of the net convective and radiative heat flux is zero. Specifying a surface to be `ADIABATIC` is equivalent to `NET_HEAT_FLUX=0` and `EMISSIVITY=1`. In such cases, it is also common to initialize the gas and surface temperatures for the purpose of measuring the heat absorbed or generated by objects within the domain, for example:

```
&INIT XB=..., TEMPERATURE=2000. /  
&SURF ID='...', ADIABATIC=T, TMP_FRONT_INITIAL=2000. /
```

Note that `TMP_FRONT_INITIAL` sets the initial temperature of the surface and has few other uses except for this particular application. If the surface is not `ADIABATIC` and it has material properties, use `TMP_INNER` to set the surface and interior temperature.

Also note that no solid surface is truly adiabatic; thus, the specification of an adiabatic boundary condition should be used for diagnostic purposes only. Actual fire simulations can develop numerical instabilities if `ADIABATIC` surfaces are used because the wall temperature and radiation fields are not updated simultaneously. If a droplet lands on an `ADIABATIC` surface, no heat transfer will be predicted between the droplet and the surface. This is because the high heat transfer rates associated with droplets can lead to nonsensical wall temperatures when trying to enforce the adiabatic condition which are then likely to result in runtime errors or numerical instabilities.

8.3 One-Dimensional Heat Conduction in Solids

Specified temperature or heat flux boundary conditions are easy to apply, but only of limited usefulness in real fire scenarios. In most cases, walls, ceilings and floors are made up of several layers of lining materials. In most fire scenarios, it is sufficient to compute the conduction of heat normal to the solid surface using a 1-D solver. There are some limited cases for which three-dimensional heat conduction is possible; see Sec. 8.4.

The `MATL` namelist group is used to define the properties of the materials that make up bounding solids. A solid boundary can consist of multiple layers¹ of different materials, and each layer can consist of multiple material components.

8.3.1 Structure of Solid Boundaries

Material layers and components are specified on the `SURF` line via the array called `MATL_ID(IL, IC)`. The argument `IL` is an integer indicating the layer index, starting at 1, the layer at the exterior boundary. The argument `IC` is an integer indicating the component index. For example, `MATL_ID(2, 3) = 'BRICK'` indicates that the third material component of the second layer is `BRICK`. In practice, the materials are often listed as in the following example:

```
&MATL ID          = 'INSULATOR'
  CONDUCTIVITY     = 0.041
  SPECIFIC_HEAT    = 2.09
  DENSITY          = 229. /

&SURF ID          = 'BRICK WALL'
  MATL_ID          = 'BRICK', 'INSULATOR'
  COLOR            = 'RED'
  BACKING          = 'EXPOSED'
  THICKNESS        = 0.20, 0.10 /
```

Without arguments, the parameter `MATL_ID` is assumed to be a list of the materials in multiple layers, with each layer consisting of only a single material component.

When a set of `SURF` parameters is applied to the face of an `OBST`, the first `MATL_ID` defines the first layer of solid material. The other `MATL_IDS` are applied in succession. If `BACKING = 'EXPOSED'`, the last `MATL_ID` is applied to the opposite face of the solid. If in the example above, `BRICK WALL` was applied to the entire `OBST` using `SURF_ID`, then when doing a heat transfer calculation from the $+x$ face to the $-x$ face, FDS would consider the `OBST` to be `BRICK` followed by `INSULATOR` and the same for a heat transfer calculation from the $-x$ face to the $+x$ face. To avoid this, specify a second `SURF` that has the reverse `MATL_ID` and use `SURF_ID6` to apply the two `SURF` definitions to opposite faces of the `OBST`.

Mixtures of solid materials within the same layer can be defined using the `MATL_MASS_FRACTION` keyword. This parameter has the same two indices as the `MATL_ID` keyword. For example, if the brick layer contains some additional water, the input could look like this:

```
&MATL ID          = 'WATER'
  CONDUCTIVITY     = 0.60
  SPECIFIC_HEAT    = 4.19
  DENSITY          = 1000. /

&SURF ID          = 'BRICK WALL'
  MATL_ID(1, 1:2) = 'BRICK', 'WATER'
```

¹The maximum number of material layers is 20. The maximum number of material components is 20.

```

MATL_MASS_FRACTION(1,1:2) = 0.95,0.05
MATL_ID(2,1)              = 'INSULATOR'
COLOR                     = 'RED'
BACKING                   = 'EXPOSED'
THICKNESS                 = 0.20,0.10 / <--- for layers 1 and 2

```

In this example, the first layer of material, Layer 1, is composed of a mixture of brick and water. This is given by the `MATL_ID` array which specifies Component 1 of Layer 1 to be brick, and Component 2 of Layer 1 to be water. The mass fraction of each is specified via `MATL_MASS_FRACTION`. In this case, brick is 95 %, by mass, of Layer 1, and water is 5 %.

Importantly, the components of the solid mixtures are treated as pure substances with no voids. The density of the mixture is

$$\rho = \left(\sum_i \frac{Y_i}{\rho_i} \right)^{-1} \quad (8.8)$$

where Y_i are the material mass fractions and ρ_i are the material bulk densities defined on the `MATL` lines. In the example above, the resulting density of the wall would be about 1553 kg/m^3 . The fact that the wall density is smaller than the density of pure brick may be confusing, but can be explained easily. If the wall can contain water, the whole volume of the wall can not be pure brick. Instead there are voids (pores) that are filled with water. If the water is taken away, there is only about 1476 kg/m^3 of brick left. To have a density of 1600 kg/m^3 for a partially void wall, a higher density should be used for the pure brick.

8.3.2 Thermal Properties

For any solid material, specify its thermal `CONDUCTIVITY` ($\text{W}/(\text{mK})$), `DENSITY` (kg/m^3), `SPECIFIC_HEAT` ($\text{kJ}/(\text{kgK})$), and `EMISSIVITY`² (0.9 by default). Both `CONDUCTIVITY` and `SPECIFIC_HEAT` can be functions of temperature. `DENSITY` and `EMISSIVITY` cannot. Temperature-dependence is specified using the `RAMP` convention. As an example, consider Marinite, a wall material suitable for high temperature applications:

```

&MATL ID              = 'MARINITE'
  EMISSIVITY          = 0.8
  DENSITY              = 737.
  SPECIFIC_HEAT_RAMP  = 'c_ramp'
  CONDUCTIVITY_RAMP   = 'k_ramp' /
&RAMP ID='k_ramp', T= 24., F=0.13 /
&RAMP ID='k_ramp', T=149., F=0.12 /
&RAMP ID='k_ramp', T=538., F=0.12 /
&RAMP ID='c_ramp', T= 93., F=1.172 /
&RAMP ID='c_ramp', T=205., F=1.255 /
&RAMP ID='c_ramp', T=316., F=1.339 /
&RAMP ID='c_ramp', T=425., F=1.423 /

```

Notice that with temperature-dependent quantities, the `RAMP` parameter `T` means Temperature, and `F` is the value of either the specific heat or conductivity. In this case, neither `CONDUCTIVITY` nor `SPECIFIC_HEAT` is given on the `MATL` line, but rather the `RAMP` names.

The solid material can be given an `ABSORPTION_COEFFICIENT` (1/m) that allows the radiation to penetrate and absorb into the solid. Correspondingly, the emission of the material is based on the internal temperatures, not just the surface. Materials with an `ABSORPTION_COEFFICIENT` can only be applied to planar surfaces, not cylinders or spheres.

²The values of `EMISSIVITY` that are specified on the various `MATL` lines that define that material components of the solid are used to compute an effective emissivity for the front and back surfaces. However, if you specify an `EMISSIVITY` or `EMISSIVITY_BACK` on the `SURF` line, these values will then take precedence.

8.3.3 Back Side Boundary Conditions

There are several options for defining the back surface boundary condition of a thermally-thick obstruction. The default, `BACKING='EXPOSED'` on the `SURF` line, assumes that the back side is exposed to the thermal environment behind the solid. FDS calculates the heat transfer through the solid into the space behind the wall and vis versa. This heat conduction calculation in the solid is based on the `THICKNESS` of the material layers specified on the `SURF` line, not the dimension of the `OBST` or other solid to which the `SURF` is applied. For example, when modeling a steel plate that is 5 mm thick, if the `OBST` is approximated as a zero-cell thick sheet because 5 mm is less than half the grid dimension of 5 cm, then FDS will still compute the heat transfer through a 5 mm thick plate of steel. If a `SURF` with `BACKING='EXPOSED'` is applied to the surface of a solid whose dimension in the normal direction is at least one gas phase grid cell length greater than `THICKNESS`, FDS will then ignore `BACKING='EXPOSED'` and apply `BACKING='VOID'` instead. This latter boundary condition is described below.

If a `VENT` is applied to a thermally-thick obstruction with 3-D heat conduction, the `SURF_ID` associated with this `VENT` must describe a thermally-thick, 3-D solid as well; that is, the `SURF` line must have a `MATL_ID` and `HT3D=T`.

An alternative back side boundary condition, `BACKING='INSULATED'`, assumes that the solid backs up to a perfectly insulated material, in which case no heat is lost to the back side. Use of this condition means specifying properties of the inner insulating material is not necessary because it is assumed to be perfectly insulated.

If the wall is assumed to always back up to the ambient, then the attribute `BACKING='VOID'` should be set. You can also set the back side exposing gas temperature to be something other than ambient using `TMP_GAS_BACK`. A `RAMP` for `TMP_GAS_BACK` can be specified with `RAMP_TMP_GAS_BACK`. The backside heat transfer coefficient is computed using a natural convection correlation (see the FDS Tech Guide [3]). Similarly, you can set the front side gas temperature using `TMP_GAS_FRONT` and ramp `RAMP_TMP_GAS_FRONT`. This is useful mainly for diagnostic cases where you want to control the exposing temperature.

The back side emissivity of the surface can be controlled by specifying `EMISSIVITY_BACK` on the `SURF` line. If not specified, the back side emissivity will be calculated during the simulations as a mass-weighted sum of the `MATL` emissivities.

8.3.4 Initial Solid Temperature

By default, the initial temperature of the solid material is set to ambient (`TMPA` on the `MISC` line). Use `TMP_INNER` on the `SURF` line to specify a different initial temperature of the solid. Note that setting this temperature causes `TMP_FRONT` and/or `TMP_BACK` to be over-ridden. One can also specify an initial temperature profile using `RAMP_T_I` where the `T` value of the ramp is the depth (m) from the face, and the `F` value is temperature at that depth. `RAMP_T_I` profile cannot be used with 3D heat transfer.

8.3.5 Walls with Different Materials Front and Back

If an `OBST` does not border the edge of the computational domain, FDS calculates the heat transfer through the specified solid `THICKNESS`, and it uses the gas phase temperature and heat flux on the front and back sides for boundary conditions. A redundant calculation is performed on the opposite side of the obstruction. The first layer listed is applied to the exposed front face and the last layer to the opposite face. Take, for example, a `SURF` line that is applied to a solid `OBST`. On the $-x$ side of the `OBST`, layer 1 is `MATERIAL A`, layer 2 is `MATERIAL B`, and layer 3 is `MATERIAL A`. On the $+x$ side the `SURF` is applied in the same manner.

```
&SURF ID          = 'SYMMETRIC'
      BACKING      = 'EXPOSED'
```

```

MATL_ID(1:3,1)      = 'MATERIAL A','MATERIAL B','MATERIAL A'
THICKNESS(1:3)      = 0.1,0.2,0.1 /

```

Now consider the `SURF` definition below. On the $-x$ side of the `OBST`, layer 1 is `MATERIAL A`, layer 2 is `MATERIAL B`, and layer 3 is `MATERIAL C`. On the $+x$ side, the `SURF` is applied in the same manner, layer 1 is `MATERIAL A`, layer 2 is `MATERIAL B`, and layer 3 is `MATERIAL C`. This means that both sides of the `OBST` will compute heat transfer assuming `MATERIAL A` is the first layer but this is not possible.

```

&SURF ID              = 'NON-SYMMETRIC'
BACKING               = 'EXPOSED'
MATL_ID(1:3,1)        = 'MATERIAL A','MATERIAL B','MATERIAL C'
THICKNESS(1:3)        = 0.1,0.2,0.1 /

```

Care is needed when specifying multiple layers. If the layering is symmetric, the same `SURF` line can be applied to both sides. However, if the layering is not symmetric, two separate `SURF` lines must be created and applied one to each side. For example, a hollow box column that is made of steel and covered on the outside by a layer of insulation material and a layer of plastic on top of the insulation material, would have to be described with two `SURF` lines like the following:

```

&SURF ID              = 'COLUMN EXTERIOR'
BACKING               = 'EXPOSED'
MATL_ID(1:3,1)        = 'PLASTIC','INSULATION','STEEL'
THICKNESS(1:3)        = 0.002,0.036,0.0063 /

&SURF ID              = 'COLUMN INTERIOR'
BACKING               = 'EXPOSED'
MATL_ID(1:3,1)        = 'STEEL','INSULATION','PLASTIC'
THICKNESS(1:3)        = 0.0063,0.036,0.002 /

```

If, in addition, the insulation material and plastic are combustible, and their burning properties are specified on the appropriate `MATL` lines, then indicating which side of the column would generate the fuel vapor is needed. In this case, the steel is impermeable. Adding the the parameter `LAYER_DIVIDE=2.0` to the `SURF` line labeled 'COLUMN EXTERIOR' would result in fuel vapors formed by the heating of the two first layers ('PLASTIC' and 'INSULATION') being driven out of that surface. Similarly, for the `SURF` line labeled 'COLUMN INTERIOR', specifying `LAYER_DIVIDE=0.0` would indicate that no fuel vapors are to driven into the interior of the column. In fact, values from 0.0 to 1.0 would work equally because the material 'STEEL' would not generate any fuel vapors. By default, `LAYER_DIVIDE` is 0.5 times the number of layers for surfaces with `EXPOSED` backing, and equal to the number of layers for other surfaces. See Section 9.2.2 for more information.

8.3.6 Specified Internal Heat Source

The condensed phase heat conduction equation has a source term that describes the internal sources and sinks of energy. There are three types of sources that contribute to this term: heats of reaction for the pyrolysis (see Sec. 9.2), internal absorption and emission of radiation (see Sec. 9.2.7), and the source specified by the user. An example of the case where specified heat source could be needed is the heating of electrical cables due to internal current.

The internal source term for each layer of the surface is specified using `INTERNAL_HEAT_SOURCE` on the `SURF` line. Its units are kW/m^3 and the default value is zero. An optional time ramp can be specified for each layer's heat source using `RAMP_IHS`. In the example below, the cylindrical surface describing a cable consists of an outer plastic layer and inner core of metal. The metal core is heated with a power of 300 kW/m^3 .

```

&SURF ID                      = 'Cable'
      THICKNESS                = 0.002,0.008
      MATL_ID(1,1)             = 'PLASTIC'
      MATL_ID(2,1)             = 'METAL'
      GEOMETRY                  = 'CYLINDRICAL'
      LENGTH                    = 0.1
      INTERNAL_HEAT_SOURCE      = 0.,300. /

```

8.3.7 Non-Planar Walls and Targets

All obstructions in FDS are assumed to conform to the rectilinear mesh, and all bounding surfaces are assumed to be flat planes. However, many objects, like cables, pipes, and ducts, are not flat. Even though these objects have to be represented in FDS as “boxes,” you can still perform the internal heat transfer calculation as if the object were really cylindrical or spherical. For example, the input lines:

```

&OBST XB=0.0,5.0,1.1,1.2,3.4,3.5, SURF_ID='CABLE' /
&SURF ID='CABLE', MATL_ID='PVC', GEOMETRY='CYLINDRICAL', THICKNESS=0.01 /

```

can be used to model a power cable that is 5 m long, cylindrical in cross section, 2 cm in diameter. The heat transfer calculation is still one-dimensional; that is, it is assumed that there is a uniform heat flux all about the object. This can be somewhat confusing because the cable is represented as an obstruction of square cross section, with a separate heat transfer calculation performed at each face, and no communication among the four faces. Obviously, this is not an ideal way to do solid phase heat transfer, but it does provide a reasonable bounding surface temperature for the gas phase calculation. More detailed assessment of a cable would require a two or three-dimensional heat conduction calculation, which is not included in FDS.

To represent the *inner* surface of a cylinder, specify `GEOMETRY='INNER CYLINDRICAL'` along with an `INNER_RADIUS (m)`.

Use `GEOMETRY='SPHERICAL'` to describe a spherical object. Add `HORIZONTAL=T` to the `SURF` line if you have specified a cylindrical geometry and you want to ensure that the heat transfer coefficient is appropriate for a horizontal cylinder.

8.3.8 Solid Phase Numerical Gridding Issues

Inside solids, FDS solves the one-dimensional heat transfer equation numerically in the direction normal to the surface. The node spacing for the numerical solver is not uniform, in general. The size of first cell at the surface is automatically chosen to be less than or equal to $\sqrt{\tau k / \rho c}$, where τ is a time constant set to 1 s and $k / \rho c$ is the thermal diffusivity. By default, the node spacing is less dense in the center of the layer and more dense at the boundaries to better resolve steep gradients in the temperature.

The default parameters governing the node spacing are appropriate for simple heat transfer calculations, but sometimes pyrolysis reactions cause spurious fluctuations in temperature and burning rate. The numerical accuracy and stability of the solid phase solution may be improved as follows:

Make the node spacing more uniform inside the material by setting `STRETCH_FACTOR(NL)=1.` on the `SURF` line, where `NL` designates a particular layer. This will generate a perfectly uniform mesh. Values between 1 and 2 give different levels of stretching. The default value of 2 indicates that the second cell is twice as thick as the first, the third is twice the second, and so on until the mid-depth of the layer is reached, at which point the cells shrink following the same pattern.

Make the mesh cells smaller by setting `CELL_SIZE_FACTOR(NL)` less than 1, where `NL` designates a particular layer. For example, a value of 0.5 makes the mesh cells in the layer half the size.

Specify the desired cell size by specifying `CELL_SIZE(NL)` directly, where `NL` designates a particular layer. This parameter over-rides `STRETCH_FACTOR` and `CELL_SIZE_FACTOR`.

Improve the time resolution by setting `WALL_INCREMENT=1` on the `TIME` line. This forces the solid phase solution to be updated every time step instead of the default every 2 time steps. If this is still not sufficient, you can direct that FDS use smaller time steps to update the solid phase heat conduction calculation than that used by the gas phase solver. This can be done specifically for a selected surface (`SURF`) type using the parameter `TIME_STEP_FACTOR`. Its default value is 10, meaning that the solid phase time step for that particular `SURF` type can be sub-divided by *at most* a factor of 10. The decision to sub-divide the time step is based on the criterion that the internal temperature of the solid should not change by more than `DELTA_TMP_MAX` °C during that sub-step. The default value of `DELTA_TMP_MAX` is 10 °C, and this is also a `SURF` parameter. You can choose `QUANTITY='SUBSTEPS'` on either a `DEVC` or `BNDF` output line to see how many sub-steps are being used by a particular surface cell or the entire domain, respectively. You can also ask FDS to use a solid phase time step that obeys the Fourier number constraint, $Fo = \delta t \alpha / \delta x^2 = 1$; this is done by setting `CHECK_FO=T` on `MISC`.

Limit the number of cells in any layer by setting `N_LAYER_CELLS_MAX(NL)` (default 1000), where `NL` designates a particular layer. Reducing this value does not necessarily improve accuracy, but it does save computing time. However, rarely does the solid phase require this many cells. The output file `CHID.out` contains the coordinates of the solid phase nodes.

Change when a wall cell is renoded by setting `REMESH_RATIO` on the `SURF` line and/or `REAC_RATE_DELTA` on the `MATL` line. If nodes changes size during pyrolysis, FDS will try and renode layers with changing cell size to optimize the number of nodes. This renoding process is costly if it is done frequently. The parameter `REMESH_RATIO` sets the fractional change in the size of a wall cell before FDS will check for renoding. For example, with the default value of 0.15, FDS will not attempt the first renode of a wall cell that is initially all 0.1 m nodes unless one node has either increased to 0.115 m or decreased to 0.085 m. A second potential challenge with renoding occurs when there are large cell-to-cell temperature differences inside a pyrolyzing layer. Renoding will combine cells causing changes in temperature. If there is a large enough temperature difference, this can cause large swings in the pyrolysis rate. The parameter `REAC_RATE_DELTA`, default value of 0.05, sets the fractional change in the pyrolysis rate for a material that is considered acceptable. If renoding of a layer could cause a temperature change large enough to achieve that change in pyrolysis rate, then FDS will not renode that layer.

If all the material components react and leave no solid residue, the thickness of the solid will shrink. Each of the shrinking layers will vanish from the computation when its thickness gets smaller than a prescribed limiting value. This value can be set on a `SURF` line using `MINIMUM_LAYER_THICKNESS(N)`, where `N` is the index of the layer. If you do not specify a layer index, the value shall be applied to all layers. The default value is 0.0001 m or 0.1 times the specified `THICKNESS`, whichever is less. When all the material of a shrinking surface is consumed but `BURN_AWAY` is not prescribed, the surface temperature is set to `TMP_GAS_BACK`, convective heat flux to zero and burning rate to zero.

See Sec. 9.3 for ways to check and improve the accuracy of the solid phase calculation.

8.3.9 Specified Front and Back Surface Temperature

If a solid obstruction is thermally-thick, typically specifying its surface temperature is undesired because the surface temperature should establish itself based on the heat flux to, and heat penetration within, the solid. However, there may be situations where this is desired. One example is for testing the heat conduction algorithm against analytical solutions for which a specified boundary temperature is typical. An example is as follows:

```
&SURF ID='...', MATL_ID='...', TMP_FRONT=500., TMP_BACK=20., THICKNESS=0.01 /
```

This line fixes the front temperature of a 1 cm thick plate to 500 °C and its back side to 20 °C. You do not need to specify the temperature of both surfaces. If the back side of the obstruction is exposed, you will need to assign another `SURF` to the back with the opposite boundary conditions. The front and back temperatures can be respectively varied with time using the ramps `RAMP_T` and `RAMP_TMP_BACK`.

8.4 3-D Heat Conduction (Beta)

The 1-D heat conduction model used in FDS is appropriate for most building materials that have a relatively low thermal conductivity; that is, they are insulating. However, materials like steel have a relatively high conductivity, and the 1-D conduction model may not be appropriate. FDS does have a limited 3-D heat conduction capability that is appropriate for structural steel or other solid objects where lateral heat transfer may be important. The 3-D conduction algorithm in FDS makes use of the 1-D solver. Every third solid phase time step, the temperature in one of the three coordinate directions is updated in time, and between each time step, this updated temperature field is transferred to the other two directions. In this way, the 3-D temperature field is updated over the course of three calls to the solid phase solver.

8.4.1 Basics

For the 3-D algorithm, each obstruction (OBST) that makes up the solid is assumed to consist of a mixture of different materials (MATLS). Multiple layers can be constructed from multiple obstructions, but these obstructions are snapped to the gas phase grid. Thus, you cannot specify arbitrarily thick layers like you can with the 1-D solver.

To invoke the 3-D solver, specify HT3D=T on a SURF line as in this example:

```
&SURF ID='STEEL SLAB', HT3D=T, COLOR='BLACK' /  
&OBST XB=..., SURF_ID='STEEL SLAB', MATL_ID='STEEL', CELL_SIZE=0.003 /  
&MATL ID='STEEL', DENSITY=7500, SPECIFIC_HEAT=0.5, CONDUCTIVITY=50 /
```

Parameters that are used to control the solid phase gridding are preferably given on the OBST line instead of the SURF line. In this example, the MATL_ID is specified on the OBST line. You can specify a uniformly-spaced grid using the parameter CELL_SIZE (m) or you can stretch the grid using the parameters STRETCH_FACTOR and CELL_SIZE_FACTOR. You can also limit the number of grid cells using N_LAYER_CELLS_MAX. These parameters are explained in Sec. 8.3.8, but when specified on an OBST line, these parameters are scalars and refer to the single layer formed by that particular obstruction. If these parameters are not specified on the OBST line, they will be taken from the first layer specified on the SURF line, and if not specified there, they will take on the default values listed in Sec. 8.3.8.

An appropriate CELL_SIZE, δ , for 3-D heat conduction is based on the material thermal diffusivity, $\alpha = k/(\rho c)$:

$$\delta \approx \min \left(\delta x, \sqrt{\frac{\tau k}{\rho c}} \right) \quad ; \quad \tau = 1 \text{ s} \quad (8.9)$$

Here, τ is just a unity time scale for the purpose of unit consistency. The gas phase grid cell size is δx . In the example above, the 'STEEL' has a conductivity of 0.05 kW/(m K), specific heat 0.5 kJ/(kg K), and density 7500 kg/m³; thus, an appropriate CELL_SIZE would be on the order of 0.004 m. FDS will adjust this value so that an integral number of cells will span the width of the solid object. Choosing a CELL_SIZE much lower than this recommended value might lead to spurious results, especially if materials of very different thermal diffusivities are components of the solid object.

You can specify an internal source for the solid obstruction using INTERNAL_HEAT_SOURCE on the OBST line. Its units are kW/m³ and the default value is zero. An optional time ramp can be specified using RAMP_IHS.

8.4.2 Limitations

1. `HT3D` cannot be applied to an exterior boundary; it must be applied to an `OBST` for which at least one face in each coordinate direction is exposed.
2. Avoid contact between 3-D and 1-D solids. If two sides of a 3-D solid touch 1-D solids, there will be no lateral heat conduction computed in that particular direction.
3. If your 3-D obstruction extends beyond meshes that abut, FDS uses a special algorithm to identify all the meshes where this obstruction lives, and also those obstructions connected to it. However, if this algorithm fails to detect all the meshes and you receive an error stating that there is a problem, add the parameter `NEIGHBOR_SEPARATION_DISTANCE` to the `MISC` line. Any mesh within this distance of another mesh will share geometry information for use in the 3-D heat conduction calculation. If you set this parameter to a value larger than the width of the computational domain, then all meshes will establish communication channels for exchanging boundary information.
4. By default, the interior nodes are clustered near the surface and stretched out deeper within the solid. If you want to maintain uniform spacing, set `CELL_SIZE` on the `SURF` or `OBST` line to indicate the desired interior node spacing. The `CELL_SIZE` is typically chosen to be comparable to the gas phase cells. If the obstruction is thin; that is, less than one gas phase cell thick, the specified `CELL_SIZE` will only apply to the heat conduction in the transverse, not normal, direction. The normal direction gridding will be controlled by the parameters `STRETCH_FACTOR` and `CELL_SIZE_FACTOR`. This may be useful in cases where the specified `CELL_SIZE` is too coarse to resolve variations in surface definition along the normal dimension (see Section 8.4.4 for an example).
5. `HT3D` cannot be applied to an `OBST` that is to `BURN_AWAY`. In addition, if the solid undergoes significant shrinking or swelling, do not use `HT3D`. “Significant” means that the number of internal cells changes, in which case the 3-D nodal structure breaks down. To determine if this happens, use the `PROFILE` output feature (Sec. 22.3) to visualize profiles of internal temperature or other solid phase quantities. These output files contain the internal node coordinates as a function of time.
6. If a `SURF` line specifies either `HT3D=T` or `VARIABLE_THICKNESS=T` and also specifies an `HRRPUA`, `MLPUA`, or `MASS_FLUX`, you must specify a `MATL_ID` on the `SURF` line and the appropriate `OBST` lines. The reason for doubly specifying the `MATL_ID` is so that the `SURF` line can be set up properly. Note that the specification of `HRRPUA` or similar on the surface of a 3-D or variably thick solid means that no obstruction making up the solid can have specified internal reactions, i.e. pyrolysis.

Example: Steel Assembly

Figure 8.1 displays a simple test case in which a steel assembly is heated by a small hot object. The plot at right asserts that the net heat flux integrated over the surface is equal to the heat absorbed by the solid:

$$\int_0^t \int_{\partial\Omega} \dot{q}_{\text{net}}'' dS dt' = \int_{\Omega} \rho_s c_s \Delta T_s dV \quad (8.10)$$

The integral on the left hand side of Eq. (8.10) is evaluated in FDS using the device line:

```
&DEVC ID='...', XB=..., QUANTITY='NET HEAT FLUX', SURF_ID='STEEL SLAB'
      SPATIAL_STATISTIC='SURFACE INTEGRAL', TEMPORAL_STATISTIC='TIME INTEGRAL' /
```

The volume enclosed by `XB` encompasses the entire assembly. The integral on the right is evaluated by:

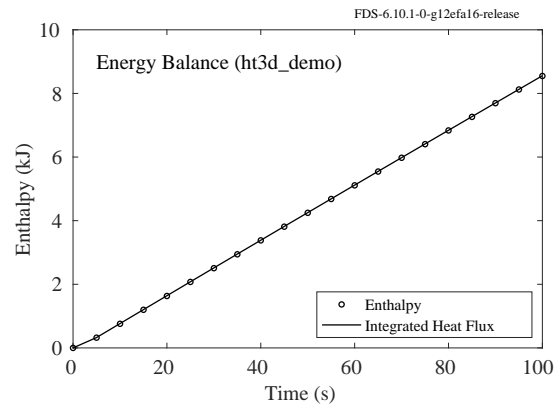
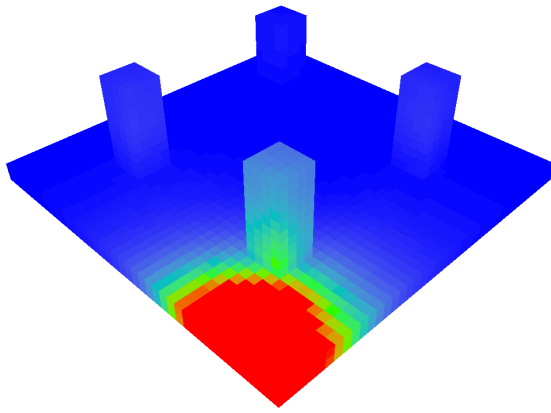


Figure 8.1: (Left) Surface temperature of a steel assembly exposed to a small hot object. (Right) Plot demonstrating that the increasing enthalpy of the steel is consistent with the integrated net heat flux over the surface.

```
&DEVC ID='...', XB=..., QUANTITY='WALL ENTHALPY', IOR=3, SURF_ID='STEEL SLAB'
      SPATIAL_STATISTIC='SURFACE INTEGRAL', RELATIVE=T, CONVERSION_FACTOR=10000, /
```

The quantity 'WALL ENTHALPY' is the total energy within the volume carved out by a wall cell from front to back. This is why the `IOR` is included because the integration need only be carried out for upward facing wall cells. Also, because the unit associated with this quantity is kJ rather than kJ/m², the `CONVERSION_FACTOR` is needed to cancel out the area of each wall cell (0.01 m by 0.01 m) after the summation. The parameter `RELATIVE=T` means that only the change in enthalpy is desired. .

Example: Unprotected Structural Steel

Structural steel members like I-beams are difficult to model for two reasons. First, the underlying grid must be reasonably well-resolved to capture the cross-sectional shape, and second, the lateral heat transfer can be important. At best, I-beams are typically modeled as a collection of thin (i.e. zero cell thick) obstructions representing the web and flanges.

```
&OBST XB=0.12,1.88,0.51,0.53,0.21,0.79, SURF_ID='STEEL SLAB', MATL_ID='STEEL' /
&OBST XB=0.12,1.88,0.41,0.63,0.76,0.79, SURF_ID='STEEL SLAB', MATL_ID='STEEL' /
&OBST XB=0.12,1.88,0.41,0.63,0.21,0.24, SURF_ID='STEEL SLAB', MATL_ID='STEEL' /
&SURF ID='STEEL SLAB', HT3D=T, COLOR='BLACK', CELL_SIZE=0.1 /
```

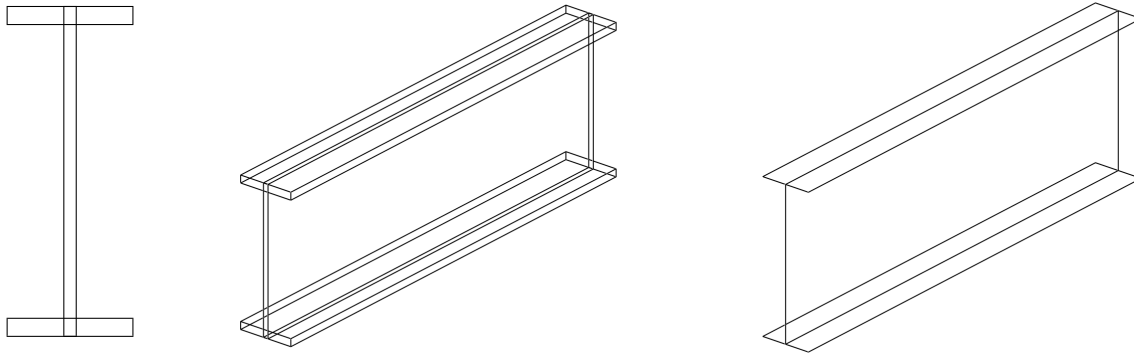


Figure 8.2: (Left) Cross section of an I-beam. (Center) 3-D representation of the beam. (Right) FDS approximated thin obstructions.

8.4.3 Surface Linings

In some applications, you might want to perform 3-D heat conduction within collections of abutting obstructions, each with a `MATL_ID`. The obstructions might have some outer lining, like fabric on sofa cushions, for example, and this lining is thinner than a single gas phase grid cell. In other words, this lining material cannot be represented by an `OBST`. However, you can specify one or more layers of multi-component materials using `MATL_ID`, `MATL_MASS_FRACTION`, and `THICKNESS` on the `SURF` line(s) that are assigned to the `OBST`s that also have specified `MATL_IDS` and `MATL_MASS_FRACTIONS`.

The way linings are handled depends on the size of the obstruction. If the obstruction is a flat plate whose depth is less than one-half a grid cell and is represented as a zero-cell thick sheet, then any layer materials are assumed to *add* to the thickness of the underlying plate. For example, if a plate of steel is 1 cm thick and it is assumed to be coated on both sides by layers of insulation that are 2 cm thick, the assembly is still treated as a thin sheet from the standpoint of the gas phase computation, but the depth is taken as 5 cm for the purpose of the heat transfer computation.

On the other hand, if the solid is made up of obstructions that are not thin plates, lining materials displace volume of the underlying solid. For example, the following lines of input define a single obstruction made of steel that is lined with two layers of insulation made up of three different components:

```
&OBST XB=-0.20, 0.20,-0.20, 0.20,-0.05, 0.05, SURF_ID='SLAB', MATL_ID='STEEL' /
&SURF ID='SLAB', HT3D=T, COLOR='BEIGE', THICKNESS=0.003,0.002,
      MATL_ID(1,1:2)='STUFF A','STUFF B', MATL_MASS_FRACTION(1,1:2)=0.3,0.7,
      MATL_ID(2,1:2)='STUFF B','STUFF C', MATL_MASS_FRACTION(2,1:2)=0.2,0.8 /
&MATL ID='STUFF A', DENSITY= 50, SPECIFIC_HEAT=1.0, CONDUCTIVITY=0.1 /
&MATL ID='STUFF B', DENSITY= 80, SPECIFIC_HEAT=1.5, CONDUCTIVITY=0.2 /
&MATL ID='STUFF C', DENSITY= 30, SPECIFIC_HEAT=2.5, CONDUCTIVITY=0.3 /
&MATL ID='STEEL', DENSITY=7500, SPECIFIC_HEAT=0.5, CONDUCTIVITY=50. /
```

The resulting solid has the same dimensions as the one you specify, but 5 mm of its surface material is replaced by the given layers of “stuff.”

Example: Insulated Structural Steel

An insulated steel beam can be modeled as a hybrid of 1-D and 3-D objects. The heat conducted through the insulation varies mainly in the direction normal to the surface, while the heat conducted along the steel beam varies mainly in the lateral direction. The following lines provide an example where a steel plate is

coated with insulation. The steel plate is entered as a relatively thin obstruction, and the insulation is applied to each side via a `SURF` line. This latter parameter ensures that the thin steel obstruction and the insulation obstruction are included in the calculation of solid overlap volumes. Note that all dimensions are exact to ensure that the overlap volumes are computed properly.

```
&OBST XB=..., SURF_ID='INSULATION', MATL_ID='STEEL', CELL_SIZE=0.1 /
&SURF ID='INSULATION', MATL_ID='STUFF', COLOR='BEIGE', THICKNESS=0.01, HT3D=T /
```

Note that the `CELL_SIZE` refers to heat transfer in the lateral direction. The heat transfer in the direction normal to the insulation and thin steel plate will be gridded according to parameters on the `SURF` line `'INSULATION'`.

8.4.4 Working with Thin Plates

The sample case called `Heat_Transfer/checkerboard.fds` demonstrates how to work with a thin plate of steel. Consider the following lines of input:

```
&MATL ID='steel', SPECIFIC_HEAT=0.515, CONDUCTIVITY=16.2, DENSITY=7900 /
&SURF ID='paint', COLOR='BLACK', EMISSIVITY=0.95, HT3D=T /
&SURF ID='no paint', COLOR='SILVER', EMISSIVITY=0.33, HT3D=T /
&OBST XB=0.00,1.20,0.00,0.0031,0.00,1.20, MATL_ID='steel', SURF_ID='no paint' /
&VENT XB=0.00,0.05,0.0000,0.0000,0.00,0.05, SURF_ID='paint', MULT_ID='M', IOR=-2 /
&VENT XB=0.00,0.05,0.0031,0.0031,0.00,0.05, SURF_ID='paint', MULT_ID='M', IOR= 2 /
&VENT XB=0.05,0.10,0.0000,0.0000,0.05,0.10, SURF_ID='paint', MULT_ID='M', IOR=-2 /
&VENT XB=0.05,0.10,0.0031,0.0031,0.05,0.10, SURF_ID='paint', MULT_ID='M', IOR= 2 /
&MULT ID='M', DX=0.10, DZ=0.10, I_UPPER=11, K_UPPER=11 /
```

There is a single thin `OBST`struction that is 3.1 mm thick and 1.2 m long and 1.2 m wide. The gas phase grid size is 25 mm, much larger than the plate thickness. From the standpoint of the gas phase fluid solver, this plate is just a barrier and has no thickness. However, its thickness of 3.1 mm is taken into account by the 3-D heat conduction solver. The plate is assigned a `MATL_ID` of `'steel'` and it is also given a default set of surface properties using the `SURF_ID='no paint'`. This `SURF_ID` invokes the 3-D solver and it assigns a color and an `EMISSIVITY` of 0.33. If no `EMISSIVITY` had been assigned, it would have taken on that of `'steel'` which in this case would have taken the default value of 0.9. The `VENT` lines apply a checkerboard pattern of “black” or high emissivity squares onto the plate. Figure 8.3 displays an image of the simulation along with a plot showing a comparison of internal enthalpy and net heat flux.

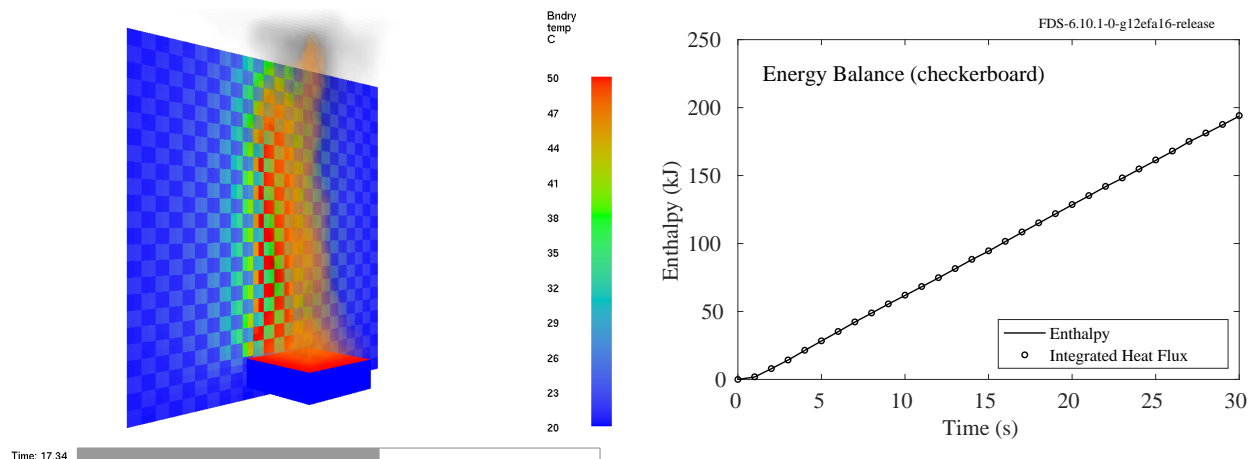


Figure 8.3: (Left) Temperature of a thin steel plate subjected to fire. The plate has a patchwork pattern of high and low emissivity. (Right) Plot comparing the enthalpy of the plate and the net surface heat flux.

8.4.5 1-D Heat Conduction in Solids of Varying Thickness

There are some scenarios that do not call for 3-D heat conduction, but do not have a well-defined `THICKNESS` which is needed by the default 1-D heat conduction algorithm. In these cases, the solid might have various layers of varying thicknesses. In such cases, you do not specify a `THICKNESS` or a `MATL_ID` on the `SURF` line. Instead, you specify `MATL_ID` on the obstructions (`OBSTS`) that make up the solid. Multiple obstructions can be used to create multiple layers, and each layer can be assigned one or more material components via `MATL_ID(:)` and `MATL_MASS_FRACTION(:)`. Unlike the way these parameters are specified on a `SURF` line, the single index represents a material component, not a layer. As an example, consider these input lines:

```
&SURF ID='SKIN', COLOR='BLACK', VARIABLE_THICKNESS=T /
&OBST XB=0.2,0.8,0.2,0.8,0.2,0.8, SURF_ID='SKIN',
      MATL_ID='Steel','Aluminum', MATL_MASS_FRACTION=0.8,0.2 /
&HOLE XB=0.3,0.7,0.3,0.7,0.3,0.7 /
&OBST XB=0.3,0.7,0.3,0.7,0.3,0.7, PERMIT_HOLE=F,
      MATL_ID='Aluminum', MATL_MASS_FRACTION=1. /
```

The `SURF` line serves only to color the surface black and to indicate, via the parameter `VARIABLE_THICKNESS`, that the layer thicknesses and material composition are to be conveyed via the `OBSTS` that make up the solid. In this case, the solid is a 0.6 m cube made up of steel and aluminum with a hollowed out core made up of pure aluminum. The material properties are conveyed via `MATL` lines as described above.

Chapter 9

Fire and Pyrolysis

FDS has several approaches for describing the pyrolysis of solids and liquids. The approach to take depends largely on the availability of material properties and the appropriateness of the underlying pyrolysis model. Note that all pyrolysis models in FDS require that you explicitly define the gas phase reaction. See Chapter 13 for details. A given surface may only use only one pyrolysis model at a time.

9.1 Specified Heat Release Rate

Solids and liquid fuels can be modeled by specifying their relevant properties via the `MATL` namelist group. However, no material properties are needed to simply obtain a fire of a given heat release rate (HRR). A specified fire is modeled as the ejection of gaseous fuel from a solid surface or vent. This is essentially a burner, with a specified Heat Release Rate Per Unit Area, `HRRPUA`, in units of kW/m^2 . For example

```
&SURF ID='FIRE', HRRPUA=500. /
```

applies $500 \text{ kW}/\text{m}^2$ to any surface with the attribute `SURF_ID='FIRE'`. See the discussion of time-dependent quantities in Chapter 11 to learn how to ramp the heat release rate up and down.

An alternative to `HRRPUA` with the exact same functionality is `MLRPUA`, except this parameter specifies the Mass Loss Rate of fuel gas Per Unit Area in $\text{kg}/(\text{m}^2 \text{ s})$. Do not specify both `HRRPUA` and `MLRPUA` on the same `SURF` line.

If either `HRRPUA` or `MLRPUA` are specified, but no `SPEC_ID` is given, then FDS will use the `FUEL` for the first `REAC` input. If a single `SPEC_ID` is given, then FDS will use that as the fuel. This can only be done if the species is used as the `FUEL` in only one `REAC` input; otherwise use `SPEC_ID` and `MASS_FLUX`. If multiple `SPEC_ID` along with `MASS_FRACTION` are given, then FDS will assume that the `HRRPUA` or `MLRPUA` refers to only species corresponding to `FUEL` inputs on a `REAC` line. For `MLRPUA`, FDS will just multiply the `MLRPUA` by the `MASS_FRACTION` of the fuel species to obtain the fuel species mass flux. If there are any non-fuel species, then the mass flux of those species will be set to preserve their specified `MASS_FRACTION`. For `HRRPUA`, FDS will compute the average heat of combustion for all the fuel species. This average heat of combustion is used to define the mass flux of the fuel species, i.e., the effective `MLRPUA`. Non-fuel species are then treated as with `MLRPUA`.

In the example below, six fuel species are used for four different surfaces with `HRRPUA`. Four of the fuel species are predefined with no `HEAT_OF_COMBUSTION` on their associated `REAC` inputs; their heats of combustion can be determined using their heats of formation. One species has a `HEAT_OF_COMBUSTION` on its `REAC` input. The sixth fuel species is user defined and used in a `REAC` with no `HEAT_OF_COMBUSTION`. Its heat of combustion can be determined using `EPUMO2`. surface `S1` has `HRRPUA` but no `SPEC_ID`; therefore, it

will use the `FUEL` species for the first `REAC` input (`METHANE`) for the fuel. The fuel mass flux for `S1` will be given by the `HRRPUA` divided by the `METHANE` heat of combustion. Surface `S2` specifies that the fuel species is `MYFUEL`. Surface `S3` will have a total mass flux computed using the `MASS_FLUX` weighted average of the heats of combustion for `PROPANE` and `ETHANE`. Surface `S4` will use species `MYFUEL2` whose heat of combustion will be the weighted average of the `TOLUENE` and `HYDROGEN` heats of combustion using the `SPEC` input's `MASS_FRACTION`. Results are shown in Fig. 9.1.

```
&SPEC ID='TOLUENE', LUMPED_COMPONENT_ONLY=T/
&SPEC ID='HYDROGEN', LUMPED_COMPONENT_ONLY=T/
&SPEC FUEL='MYFUEL', C=1, H=1, O=1/
&SPEC ID='MYFUEL2', SPEC_ID='TOLUENE', 'HYDROGEN', MASS_FRACTION=0.5, 0.5/

&REAC FUEL='METHANE' /
&REAC FUEL='PROPANE' /
&REAC FUEL='ETHANE', HEAT_OF_COMBUSTION=40000/
&REAC FUEL='MYFUEL2' /

&SURF ID='S1', COLOR='RED', HRRPUA=1000/
&SURF ID='S2', COLOR='ORANGE', HRRPUA=1000, SPEC_ID='MYFUEL' /
&SURF ID='S3', COLOR='YELLOW', HRRPUA=1000, SPEC_ID='PROPANE', 'ETHANE',
      MASS_FRACTION=0.2, 0.8/
&SURF ID='S4', COLOR='GREEN', HRRPUA=1000, SPEC_ID='MYFUEL2' /
```

If the second surface was specified as:

```
&SURF ID='S2', COLOR='ORANGE', HRRPUA=1000, SPEC_ID='MYFUEL', 'NITROGEN',
      MASS_FRACTION=0.25, 0.75/
```

then the fuel mass flux would be the same as before, but now it would be diluted with three times the mass flux of `NITROGEN`.

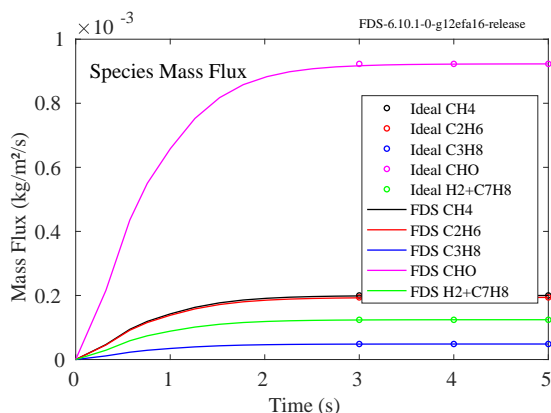


Figure 9.1: Demonstration of specifying `HRRPUA` with multiple reactions.

9.1.1 A Radially-Spreading Fire

Sometimes modeling a fire that spreads radially at some specified rate is desired. Rather than trying to obtain material properties to directly model the ignition and spread of the fire, the fire spread rate can be specified directly. First, a `SURF` line with a specified heat release rate, `HRRPUA`, is added with any desired optional time history parameter, `RAMP_Q` or `TAU_Q` (see Section 11.1). Then, specify `XYZ` and `SPREAD_RATE` on either

a VENT or the same SURF line. The fire is directed to start at the point XYZ (if unspecified, default is the center point of the VENT) and spread radially at a rate of SPREAD_RATE (m/s). The optional ramp-up of the HRR begins at the time when the fire arrives at a given point. For example, the lines

```
&SURF ID='FIRE', HRRPUA=500.0, RAMP_Q='fireramp' /
&RAMP ID='fireramp', T= 0.0, F=0.0 /
&RAMP ID='fireramp', T= 1.0, F=1.0 /
&RAMP ID='fireramp', T=30.0, F=1.0 /
&RAMP ID='fireramp', T=31.0, F=0.0 /
&VENT XB=0.0,5.0,1.5,9.5,0.0,0.0, SURF_ID='FIRE', XYZ=1.5,4.0,0.0, SPREAD_RATE=0.03 /
```

create a rectangular area via the VENT line on which the fire starts at the point (1.5,4.0,0.0) and spreads outward at a rate of 0.03 m/s. Each surface cell burns for 30 s as the fire spreads outward, creating a widening ring of fire. Note that the RAMP_Q is used to turn the burning on and off to simulate the consumption of fuel as the fire spreads radially. It should not be used to mimic a t -squared fire growth rate – the whole point of the exercise is to mimic this curve in a more natural way. Eventually, the fire goes out as the ring grows past the boundary of the rectangle. Some trial and error is probably required to find the SPREAD_RATE that leads to a desired time history of the heat release rate.

In cases where the fire spreads over an area that is a rectangular plane, specify XYZ and SPREAD_RATE on the SURF line directly and then apply that SURF line to the obstructions or particles over which you want the fire to spread. This technique can be useful for simulating the spread of fire through a cluttered space when the detailed properties of the materials are unknown, or when the uncertainties associated with modeling the pyrolysis of the solid fuels directly are too great.

If the starting time of the simulation, T_BEGIN, is not zero, be aware that the default start time of the radially spreading fire is T_BEGIN, not zero. This is also true of TAU_Q, but it is not true of RAMP_Q. Because this might be confusing, if you start the calculation at a time other than zero, do a quick test to ensure that the ramps or fire spread behave as expected.

9.1.2 Compensating for the unresolved surface area

If the obstruction or vent with the pyrolysis model represents an object that has more complex shape than a rectangular box, or if the mesh resolution does not allow resolving the real surface area, adding the parameter AREA_MULTIPLIER on the SURF line will manually adjust the burning rate with a constant factor. The default value is 1.0. This factor will also affect the rate of heat transfer between the object and gas phase.

9.1.3 Thermally-Thick Solids that Burn at a Specified Rate

Real objects, like furnishings, office equipment, and so on, are often difficult to describe via the SURF and MATL parameters. Sometimes the only information about a given object is its bulk thermal properties, its “ignition” temperature, and its subsequent burning rate as a function of time from ignition. For this situation, add lines similar to the following:

```
&MATL ID                                = 'stuff'
CONDUCTIVITY                           = 0.1
SPECIFIC_HEAT                           = 1.0
DENSITY                                = 900.0 /

&SURF ID                                = 'my surface'
COLOR                                   = 'GREEN'
MATL_ID                                 = 'stuff'
HRRPUA                                  = 1000.
```

```

IGNITION_TEMPERATURE = 500.
RAMP_Q                = 'fire_ramp'
THICKNESS              = 0.01 /

&RAMP ID='fire_ramp', T= 0.0, F=0.0 /
&RAMP ID='fire_ramp', T= 10.0, F=1.0 /
&RAMP ID='fire_ramp', T=310.0, F=1.0 /
&RAMP ID='fire_ramp', T=320.0, F=0.0 /

```

An object with surface properties defined by 'my surface' shall burn at a rate of 1000 kW/m² after a linear ramp-up of 10 s following its “ignition” when its surface temperature reaches 500 °C. Burning shall continue for 5 min, and then ramp-down in 10 s. Note that the time *T* in the *RAMP* means time from ignition, not the time from the beginning of the simulation. Note also that now the “ignition temperature” is a surface property, not material property.

After the surface has ignited, the heat transfer into the solid is still calculated, but there is no coupling between the burning rate and the surface temperature. As a result, the surface temperature may increase too much. To account for the energy loss due to the vaporization of the solid fuel, *HEAT_OF_VAPORIZATION* can be specified for the surface. For example, when using the lines below, the total heat flux at the material surface is reduced by a factor 1000 kJ/kg times the instantaneous burning rate.

```

&SURF ID              = 'my surface'
COLOR                 = 'GREEN'
MATL_ID               = 'stuff'
HRRPUA                = 1000.
IGNITION_TEMPERATURE = 500.
HEAT_OF_VAPORIZATION = 1000.
RAMP_Q                = 'fire_ramp'
THICKNESS              = 0.01 /

```

Finally, if you desire that the burning stop if the surface temperature drops below a specified value, set *EXTINCTION_TEMPERATURE* on the *SURF* line. This value should be less than or equal to the *IGNITION_TEMPERATURE*.

The parameters *HRRPUA*, *IGNITION_TEMPERATURE*, *EXTINCTION_TEMPERATURE*, and *HEAT_OF_VAPORIZATION* are all telling FDS that you want to control the burning rate yourself, but you still want to simulate the heating up and “ignition” of the fuel. When these parameters appear on the *SURF* line, they are acting in concert. If *HRRPUA* appears alone, the surface will begin burning at the start of the simulation, like a piloted burner. The addition of an *IGNITION_TEMPERATURE* delays burning until your specified temperature is reached. The addition of *HEAT_OF_VAPORIZATION* tells FDS to account for the energy used to vaporize the fuel. For any of these options, if a *MATL* line is invoked by a *SURF* line containing a specified *HRRPUA*, then that *MATL* ought to have only thermal properties. The *MATL* line should have no reaction parameters, product yields, and so on, like those described in the previous sections. By specifying *HRRPUA*, the burning rate is being controlled rather than letting the material pyrolyze based on the conditions of the surrounding environment. Also note that this simple model assumes that the solid acts like a typical *thermoplastic* material, i.e. it pyrolyzes near the surface leaving relatively little char.

9.1.4 Scaling Pyrolysis (SPyro) Model: Scaled Burning Rate from Cone Data

FDS has a simple scaling-based pyrolysis model (SPyro) that scales burning rate data collected from a cone calorimeter or similar device to a different material thickness and accounts for the dynamic heat feedback during an FDS simulation. The inputs for this model are similar to Section 9.1.3 with some additional parameters. The frequently used parameters for this model are:

- **HRRPUA (Required)** Specified heat release rate per unit area (HRRPUA) for the `SURF` which is multiplied by `RAMP_Q`. Typically, HRRPUA is set to 1.0 kW/m^2 using this model.
- **IGNITION_TEMPERATURE (Required)** Specifies ignition temperature of the material in degrees Celsius.
- **INERT_Q_REF** Specifies whether the test data represents an inert test device where pyrolysis occurs without combustion. This flag applies to all specified test data if an array of experimental data is provided. (Default False)
- **RAMP_Q (Required)** Specifies the `ID` of a `RAMP` which contains the experimental data with `T` as time and `F` as the measured HRRPUA in kW/m^2 . The test data should be adjusted so that 0 s in the ramp represents the time the sample started burning/pyrolyzing in the test.
- **REFERENCE_HEAT_FLUX (Required)** Specifies the radiant heat flux in kW/m^2 imposed by the test device to the sample prior to the start of pyrolysis, i.e., what a water-cooled heat flux gauge would measure.
- **REFERENCE_THICKNESS** Specifies the thickness of the sample in the experiment. If not specified, FDS will assume the experiment was at the same `THICKNESS(1)` defined on the `SURF` line. Note that this is just the combustible portion of the sample and not any insulation or other inert backing materials. (Default `THICKNESS(1)`)

Data from up to ten experiments can be specified by specifying arrays for `REFERENCE_HEAT_FLUX`, `RAMP_Q`, and `REFERENCE_THICKNESS`. If data from multiple experiments are provided, FDS will interpolate between experimental data when incident heat fluxes are within the range of tested data. Values for `REFERENCE_HEAT_FLUX` and `REFERENCE_THICKNESS` must be grouped by increasing thickness with increasing flux in each group of thicknesses. Typically, when samples are acquired for cone testing there are slight differences in thickness due to manufacturing or sample preparation. If, within a group of tests, the `REFERENCE_HEAT_FLUX` values are intended to be applied to the same thickness, then `REFERENCE_THICKNESS` should be specified as the same, nominal value.

There are other optional parameters which can be used to adjust the model behavior, which are described below.

- **MAXIMUM_SCALING_HEAT_FLUX** By default the model does not set an upper limit on the predicted heat flux used to scale the test data (Default $100,000 \text{ kW/m}^2$).
- **MINIMUM_SCALING_HEAT_FLUX** By default the model does not set a lower limit on the predicted heat flux used to scale the test data. Setting a lower bound to the flux used for scaling may help the model predictions in some cases such as weakly burning sources on coarse grids which may have poorly resolved heat feedback (Default 0 kW/m^2).
- **REFERENCE_HEAT_FLUX_TIME_INTERVAL** The instantaneous heat feedback in FDS can vary greatly from time step to time step. In an actual fire the burning rate is tied to the sample temperature, and thermal inertia means the burning rate changes more slowly than then instantaneous flux. This natural smoothing can be approximated by setting a smoothing window using `REFERENCE_HEAT_FLUX_TIME_INTERVAL` in s (Default 1 s).

The example, `spyro_cone_demo.fds`, demonstrating the scaling behavior is shown in Fig. 9.2. In this example there is a material with cone test data at 25, 50, and 75 kW/m^2 . Using just the 50 kW/m^2 , the sample is exposed to fluxes of 25, 50, and 75 kW/m^2 . It can be seen that at 25 kW/m^2 the test data is stretched out in time by a factor of 2 with a reduction in burning rate of a factor of 2. Similarly at 75 kW/m^2 , the curve is collapsed by 50 % with the burning rate increased by 50 %. The 50 kW/m^2 simply

returns the test data curve. A fourth surface is modeled that uses the 25 and 75 kW/m² with the sample exposed to a flux of 50 kW/m². The resulting burning rate is an average of the 25 and 75 kW/m² test data. Both extrapolation and interpolation do not perfectly reproduce the test data; however, they do capture the general behavior in terms of burning duration, peak burning rate, and time of peak.

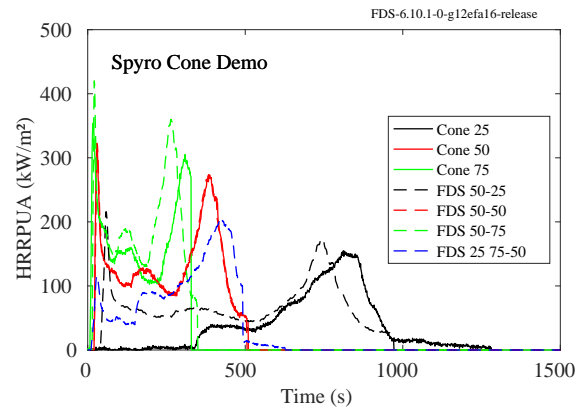


Figure 9.2: Demonstration of extrapolating cone test data to other heat fluxes.

9.2 Complex Pyrolysis Models

This section describes the parameters that describe the reactions that occur within solid materials when they are burning. It is strongly recommended before reading this section that you read some background material on solid phase pyrolysis, for example “Thermal Decomposition of Polymeric Materials,” by Witkowski, Stec, and Hull, or “Flaming Ignition of Solid Fuels,” by Torero, both of which are in the 5th edition of the *SFPE Handbook of Fire Protection Engineering*.

9.2.1 Reaction Mechanism

A solid surface in FDS may consist of multiple layers with multiple material components per layer. The material components are described via `MATL` lines and are specified on the `SURF` line that describes the structure of the solid. Each `MATL` can undergo several reactions that may occur at different temperatures. It may not undergo any – it may just heat up. However, if it is to change form via one or more reactions, designate the number of reactions with the integer `N_REACTIONS`. `N_REACTIONS` *must* be set or else FDS will ignore all parameters associated with reactions. Note that experimental evidence of multiple reactions does not imply that a single material is undergoing multiple reactions, but rather that multiple material components are undergoing individual reactions at distinct temperatures. Currently, the maximum number of reactions for each material is 10 and the chain of consecutive reactions may contain up to 20 steps.

For a given material, the j th reaction can produce other solid materials whose names are designated with `MATL_ID(i, j)`, gas species whose names are designated with `SPEC_ID(i, j)`, and particles whose particle classes are designated with `PART_ID(i, j)`. Note that the index, i , runs from 1 to the number of materials, gaseous species, or particle classes. This index does *not* correspond to the order in which the `MATL` or `SPEC` lines are listed in the input file. For a given reaction, the relative amounts of solid, gaseous, or particle products are input to FDS via the *yields*: `NU_MATL(i, j)`, `NU_SPEC(i, j)`, and `NU_PART(i, j)`, respectively. The yields are all zero by default. If `NU_MATL(i, j)` is non-zero, then you *must* indicate what the solid residue is via `MATL_ID(i, j)`, the ID of another `MATL` that is also listed in the input file. If `NU_SPEC(i, j)` is non-zero, then you *must* indicate what the gas species is via `SPEC_ID(i, j)`, the ID of another `SPEC` that is also listed in the input file. If `NU_PART(i, j)` is non-zero, then you *must* indicate what the particle class is via `PART_ID(i, j)`, the ID of another `PART` that is also listed in the input file. If particles are specified, the insertion rate and number of particles is controlled via the particle inputs on the `SURF` containing the material. Ideally, the sum of the yields should add to 1, meaning that the mass of the reactant is conserved. However, there are instances when it is convenient to have the yields that sum to less than one. For example, the spalling or ablation of concrete can be described as a “reaction” that consumes energy but does not produce any “product” because the concrete is assumed to have either fallen off the surface in chunks or pulverized powder. The concrete’s mass is not conserved *in the model* because it has essentially disappeared from that particular surface. Producing more mass (yields more than 1) will result in an error message.

For consistency, the `HEAT_OF_COMBUSTION(i, j)` can also be specified for each species, i , in each reaction, j . These values are used only if the corresponding heats of combustion for the gaseous species are greater than zero. Note that `HEAT_OF_COMBUSTION(i, j)` discussed here is not necessarily the same as the `HEAT_OF_COMBUSTION` of the surrogate `FUEL` used for gas phase combustion (e.g., 'PROPANE'). When the heats of combustion differ, the mass flux of the surrogate `FUEL` in the gas phase is adjusted so that the correct heat release rate is attained. This is further discussed in the next section.

In the example below, the pyrolysis of wood is included within a simulation that uses a finite-rate reaction instead of the default mixing-controlled model. Notice in this case that all of the gas species (except for the background nitrogen) are explicitly defined, and as a result, FDS needs to be told explicitly what gaseous species are produced by the solid phase reactions. In this case, 82 % of the mass of wood is converted to gaseous 'PYROLYZATE' and 18 % is converted to solid 'CHAR'.

```

&SPEC ID = 'PYROLYZATE', MW=53.6 /
&SPEC ID = 'OXYGEN', MASS_FRACTION_0 = 0.23 /
&SPEC ID = 'WATER VAPOR' /
&SPEC ID = 'CARBON DIOXIDE' /

&MATL ID                                = 'WOOD'
    EMISSIVITY                          = 0.9
    CONDUCTIVITY                        = 0.2
    SPECIFIC_HEAT                       = 1.3
    DENSITY                             = 570.
    N_REACTIONS                         = 1
    A(1)                               = 1.89E10
    E(1)                               = 1.51E5
    N_S(1)                             = 1.0
    MATL_ID(1,1)                       = 'CHAR'
    NU_MATL(1,1)                       = 0.18
    SPEC_ID(1:4,1)                     = 'OXYGEN', 'WATER VAPOR', 'CARBON DIOXIDE', 'PYROLYZATE'
    NU_SPEC(1:4,1)                     = 0,0,0,0.82
    HEAT_OF_REACTION(1)                 = 430.
    HEAT_OF_COMBUSTION(4,1)             = 14500. /

```

Note that the indices associated with the parameters are not needed *in this case*, but they are shown to emphasize that, in general, there can be multiple reactions with corresponding kinetic parameters and products.

9.2.2 Solid Phase Gas Transport

The solid phase conduction/reaction algorithm does not have an explicit transport mechanism for pyrolyzed gases. Rather, the pyrolyzates are assumed to appear instantaneously at the surface. Which surface is controlled by the `SURF` line parameter `LAYER_DIVIDE`, a real number that specifies the number of layers whose gaseous pyrolyzates are to be applied to the surface with the given `SURF` label. For example, if `LAYER_DIVIDE=1.5`, the gases generated by the first layer and half of the second layer shall be applied at the given surface. This same partitioning holds whether or not the solid is shrinking or swelling. Be careful to ensure that the specified values of `LAYER_DIVIDE` are consistent for the `SURF` lines that control opposing surfaces. If two different `SURF` lines govern the front and back of a solid obstruction, the specified values of `LAYER_DIVIDE` should sum to the total number of layers. Otherwise, the mass of evolved gases may not be correct.

If `LAYER_DIVIDE` is not set, it is assumed that for a solid with `BACKING='EXPOSED'`, the gases generated within a depth of half the total thickness are to be applied at the front surface and the other half at the back. If the `BACKING` is not `'EXPOSED'`, all of the gases will be applied at the front surface.

Suppose, for example, that the solid is composed of a layer of insulation on top of a layer of plastic on top of a layer of steel. The steel is impermeable. By setting `LAYER_DIVIDE=2.0` to the `SURF` line whose first layer is the insulation directs the vapors generated by the insulation and plastic to be driven out of the exterior surface of the insulation. Similarly, for the `SURF` line that is applied to the steel, specifying `LAYER_DIVIDE=0.0` would indicate that no fuel vapors are to escape the steel surface. Note that in this instance, the sum of the values of `LAYER_DIVIDE` is not equal to the number of layers, but this is not a problem because the layer of steel does not generate any gases.

The solid phase output `QUANTITY='LAYER DIVIDE DEPTH'` can check if the `LAYER_DIVIDE` is being applied correctly. For example, the input line

```
&BNDF QUANTITY='LAYER DIVIDE DEPTH', CELL_CENTERED=T /
```

shows the distance from the surface where the pyrolyzates are directed toward that surface, as opposed to the opposite side of the solid.

9.2.3 Reaction Rates

The mass per unit volume of material component i , $\rho_{s,i}(x,t)$, is a function of the depth into the solid, x , and time, t . It evolves in time according to the following equation:

$$\frac{\partial \rho_{s,i}}{\partial t} = - \sum_{j=1}^{N_{r,i}} r_{ij} + \sum_{i'=1}^{N_m} \sum_{j=1}^{N_{r,i'}} v_{s,i'j} r_{i'j} \quad (i' \neq i) \quad (9.1)$$

where

$$r_{ij} = A_{ij} \rho_{s,i}^{n_{s,ij}} T_s^{n_{t,ij}} \exp\left(-\frac{E_{ij}}{RT_s}\right) X_{O_2}^{n_{O_2,ij}} \quad (9.2)$$

The term, r_{ij} , defines the rate of reaction at the temperature, T_s , of the i th material undergoing its j th reaction. The second term on the right of the equation (9.1) represents the contributions of other materials producing the i th material as a residue with a yield of $v_{s,i'j}$. This term is denoted by `NU_MATL(:,j)` on the i' -th `MATL` line. $\rho_{s,i}$ is the density of the i th material component of the layer, defined as the mass of the i th material component divided by the volume of the layer. Thus, $\rho_{s,i}$ is a quantity that increases if the i th material component is produced as a residue of some other reaction, or decreases if the i th component decomposes. $n_{s,ij}$ is the reaction order and prescribed under the name `N_S(j)`, and is 1 by default. If the value of n_s is not known, it is a good starting point to assume it is 1. $n_{t,j}$ is prescribed under the name `N_T(j)`. By default, $n_{t,j}$ is zero.

The pre-exponential factor¹, A_{ij} , is prescribed under the name `A(j)` on the `MATL` line of the i th material, with units of s^{-1} . E_{ij} , the activation energy, is prescribed via `E(j)` in units of J/mol. Remember that 1 kcal is 4.184 kJ, and be careful with factors of 1000. For a given reaction, specify both A and E , or neither. Do not specify only one of these two parameters. Typically, these parameters only have meaning when both are derived from a common set of experiments, like TGA (thermogravimetric analysis).

The fourth term of the reaction rate equation (9.2) can be used to simulate oxidation reactions. If the heterogeneous reaction order $n_{O_2,ij}$ is greater than zero, the reaction rate is affected by the local oxygen volume fraction, X_{O_2} . It is calculated from the gas phase (first grid cell) oxygen volume fraction $X_{O_2,g}$ by assuming simultaneous diffusion and consumption so that the concentration profile is in equilibrium. The concentration at depth x is given by

$$X_{O_2}(x) = X_{O_2,f} \exp(-x/L_g) \quad (9.3)$$

where L_g is the gas diffusion length scale and $X_{O_2,f}$ is computed to satisfy simultaneous mass transfer and solid phase reaction rates (see FDS Tech Guide [3]). $n_{O_2,ij}$ is prescribed under the name `N_O2(j)` on the `MATL` line of the i th material. It is zero by default. L_g is prescribed under the name `GAS_DIFFUSION_DEPTH(j)`, and it is 0.001 m by default. The heat release rate per unit area associated with oxidation reactions can be output using the solid phase `QUANTITY` called 'OXIDATIVE HRRPUA'. This is an integral in depth of the contributions from all reactions and all materials for a given solid surface.

You can specify `MAX_REACTION_RATE(j)` ($kg/m^3 s$) to limit the reaction rate, r_{ij} , below a specified value.

¹For versions of FDS up to and including 6.7.7, the form of the reaction rate expression was slightly different than the one currently used. In these older versions, the density term on the right hand side of Eq. (9.2) was divided by the initial density of the layer. For first-order reactions ($n_{s,ij} = 1$), this older form is equivalent to the current form. However, for reactions that are not first-order ($n_{s,ij} \neq 1$), the new value of $A_{ij} = A_{ij}^{old} / \rho_s(0)^{n_{s,ij}-1}$, where $\rho_s(0)$ is the initial density of the layer.

Estimating Kinetic Parameters

The kinetic constants, A and E , are typically not available for most real materials. However, there is a way to model material decomposition using a simplified reaction scheme. The key assumption is that each material component undergoes only one reaction. If, for example, the composite material undergoes three distinct reactions, it must be represented by three material components, each of which undergoes one reaction. If there is an additional residue left over, then a fourth material component is needed. Any or all of the material components can leave behind a residue.

In lieu of specifying A and E , there are several parameters that can be used by FDS to derive effective values, the most important of which is the `REFERENCE_TEMPERATURE` (°C). To understand this parameter, consider the plot shown in Fig. 9.3. These curves represent the results of a hypothetical TGA experiment in which a single component material undergoes a single reaction that converts the solid into a gas. The Normalized Mass (blue curve, Y_s) decreases as the sample is slowly heated, in this case at a rate of 5 K/min. The Normalized Mass Loss Rate (red curve) is the rate of change of the normalized mass as a function of time ($-dY_s/dt$). Where this curve peaks is referred to in FDS as the `REFERENCE_TEMPERATURE`, which is not necessarily equivalent to an ignition temperature, nor is it necessarily the surface temperature of the burning solid. Rather, it is simply the temperature at which the mass fraction of the material decreases at its maximum rate within the context of a TGA or similar experimental apparatus.

The kinetic constants for component i of a multi-component solid are given by²:

$$E_{i,1} = \frac{e r_{p,i}}{Y_{s,i}(0)} \frac{RT_{p,i}^2}{\dot{T}} \quad ; \quad A_{i,1} = \frac{e r_{p,i}}{Y_{s,i}(0)} e^{E/RT_{p,i}} \quad (9.4)$$

where $T_{p,i}$ and $r_{p,i}/Y_{s,i}(0)$ are the reference temperature and rate, respectively. The `REFERENCE_RATE` is the normalized mass loss rate, in units of s^{-1} , at the given `REFERENCE_TEMPERATURE` divided by the mass fraction, $Y_{s,i}(0)$, of the material component in the original sample undergoing the reaction. For a single component, single reaction material, $Y_{s,1}(0) = 1$. The `HEATING_RATE` (\dot{T}) is the rate at which the temperature of the TGA (or equivalent) test apparatus was increased. It is input into FDS in units of K/min (in the formula, it is expressed in K/s). Its default value is 5 K/min. In Fig. 9.3, the area under the red curve (Normalized Mass Loss Rate) is equal to the heating rate (in units of K/s).

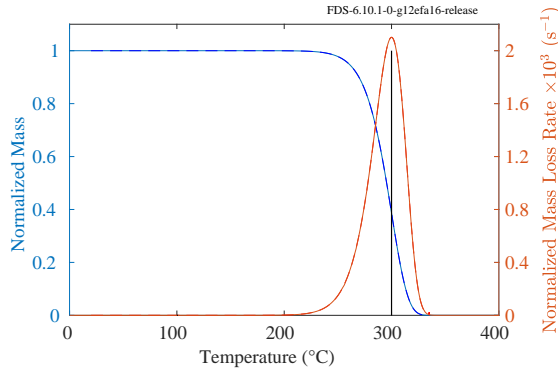
There are many cases where it is only possible to estimate the `REFERENCE_TEMPERATURE` (T_p) of a particular reaction because micro-scale calorimetry data is unavailable. In such cases, an additional parameter can be specified to help fine tune the shape of the reaction rate curve, assuming some sort of measurement or estimate has been made to indicate at what temperature, and over what temperature range, the reaction takes place. The `PYROLYSIS_RANGE` (ΔT) is the approximate width (in degrees Celsius or Kelvin) of the normalized mass loss rate curve, assuming its shape to be roughly triangular. Its default value is 80 °C. Using these input parameters, an estimate is made of the peak reaction rate, $r_{p,i}$, with which $E_{i,1}$, then $A_{i,1}$, are calculated.

$$\frac{r_{p,i}}{Y_{s,i}(0)} = \frac{2\dot{T}}{\Delta T} (1 - v_{s,i}) \quad (9.5)$$

The parameter, $v_{s,i}$, is the yield of solid residue.

When in doubt about the values of these various parameters, just specify the `REFERENCE_TEMPERATURE`. Note that FDS will automatically calculate A and E using the above formulae. Do not specify A and E if you specify the `REFERENCE_TEMPERATURE`, and do not specify `PYROLYSIS_RANGE` if you specify `REFERENCE_RATE`. For the material decomposition shown in Fig. 9.3, the `MATL` would have the form:

²These formulas have been derived from an analysis that considers a first-order reaction. When using this method, do not specify a non-unity value for the reaction order `N_S` on the `MATL` line.



$$\frac{dY_s}{dt} = -A Y_s \exp(-E/RT_s) \quad Y_s(0) = 1$$

$$\begin{aligned} T_p &= 300 \text{ }^\circ\text{C} \\ r_p &= 0.002 \text{ s}^{-1} \\ \dot{T} &= 5 \text{ K/min} \\ v_s &= 0 \end{aligned}$$

Figure 9.3: The blue curve represents the normalized mass, $Y_s = \rho_s/\rho_s(0)$, of a solid material undergoing heating at a rate of 5 K/min. The red curve represents the reaction rate, $-dY_s/dt$. The ordinary differential equation that describes the transformation is shown at right. Note that the parameters T_p , r_p , and v_s represent the “reference” temperature, reaction rate, and residue yield of the single reaction. From these parameters, values of A and E can be estimated using the formulae in (9.4). The full set of parameters for this case are listed in `pyrolysis_1.fds`.

```
&MATL ID              = 'My Fuel'
...
N_REACTIONS           = 1
SPEC_ID(1,1)          = '...'
NU_SPEC(1,1)          = 1.
REFERENCE_TEMPERATURE(1) = 300.
REFERENCE_RATE(1)      = 0.002
HEATING_RATE(1)       = 5.
HEAT_OF_COMBUSTION(1)  = ...
HEAT_OF_REACTION(1)   = ... /
```

Note that the indices have been added to the reaction parameters to emphasize the fact that these parameters are stored in arrays of length equal to `N_REACTIONS`. If there is only one reaction, the (1) may be omitted, but using it anyways ensures consistency. If the default combustion model is used, you can denote that the reaction produces fuel gas using the appropriate `SPEC_ID`. The `HEAT_OF_COMBUSTION` is the energy released per unit mass of fuel gas that mixes with oxygen and combusts. This has nothing to do with the pyrolysis process. It is often convenient when there are multiple materials that are pyrolyzing in a model, to represent the pyrolyzates using some average gas species. Specifying the `HEAT_OF_COMBUSTION` for each material ensures that the fuel vapors from different materials combust to produce the proper amount of energy. The mass loss rate of fuel gases is automatically adjusted so that the effective mass loss rate multiplied by the gas phase heat of combustion for the average fuel species produces the expected heat release rate. This adjustment uses the value of `HOC_COMPLETE`, see Sec. 13.1.4, on the `REAC` line. If, for example, the `HOC_COMPLETE` specified on the `REAC` line is twice that specified on the `MATL` line, the mass of contained within wall cell will be decremented by that determined by the pyrolysis model, but the mass added to gas phase would be reduced by 50 %. A different value of heat of combustion can be specified for each species, i , in each reaction, j , via the parameter `HEAT_OF_COMBUSTION(i, j)`.

Modeling Upholstered Furnishings

The example input file called `Fires/couch.fds` demonstrates a simple way to model upholstered furniture. Modeling a couch requires a simplification of its structure and materials. At the very least, we want the upholstery to be modeled as a layer of fabric covering polyurethane foam. The thermal properties of each

are needed, along with estimates of the “reference” temperatures as described above. The foam might be described as follows:

```
&MATH ID          = 'FOAM'  
    SPECIFIC_HEAT  = 1.0  
    CONDUCTIVITY   = 0.1  
    DENSITY        = 40.0  
    NU_SPEC        = 1.  
    SPEC_ID        = 'POLYURETHANE'  
    REFERENCE_TEMPERATURE = 280.  
    HEAT_OF_REACTION = 800. /
```

Note that these properties are completely made up. Both the fabric and the foam decompose into fuel gases via single-step reactions. The fuel gases from each have different composition and heats of combustion. FDS automatically adjusts the mass loss rate of each so that the “effective” fuel gas is that which is specified on the REAC line. Figure 9.4 shows the fire after 1 min, 2 min, 3 min, and 10 min. Only the flame zone of the fire is shown; the smoke is hidden so that you can see the fire progressing along the couch. Also shown in the Fig. 9.4 is the heat release rate of the burning couch. The “Expected HRR” is a rough estimate of the peak HRR of a piece of furniture of this size.

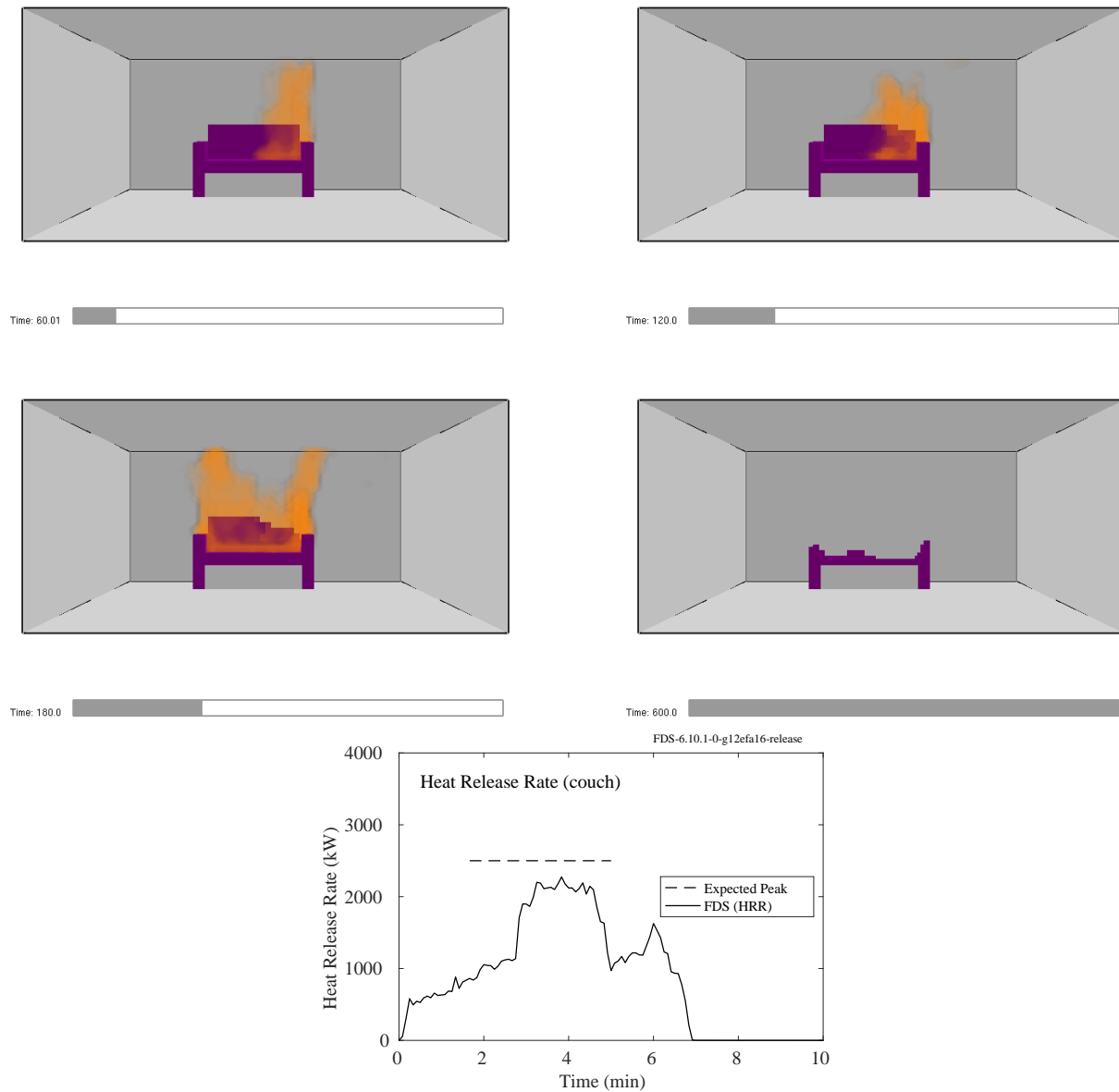


Figure 9.4: Smokeview images of the `couch` test case after 1, 2, 3, and 10 min, plus the heat release rate as a function of time.

9.2.4 Shrinking and Swelling materials

Many practical materials change in thickness during the thermal reactions. For example:

- Non-charring materials will shrink as material is removed from the condensed phase to the gas phase.
- Porous materials like foams would shrink when the material melts and forms a non-porous layer.
- Some charring materials swell, i.e., get thicker, when a porous char layer is formed.
- Intumescent fire protection materials would swell significantly, creating an insulating layer.

In FDS, the layer thickness is updated according to the ratio of the instantaneous material density and the density of the material in its pure form, i.e., the `DENSITY` on the `MATL` line. In cases involving several material components, the amount of swelling and shrinking is determined by the maximum and sum of these ratios, respectively. In mathematical terms, this means that in each time step the size of each condensed phase cell is changed according to the ratio δ

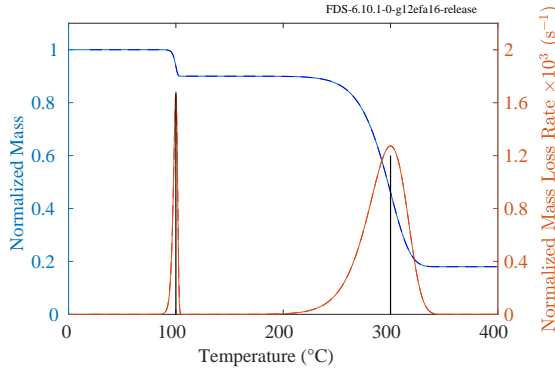
$$\delta = \begin{cases} \max_i \left(\frac{\rho_{s,i}}{\rho_i} \right) & \text{if } \max_i \left(\frac{\rho_{s,i}}{\rho_i} \right) \geq 1 \\ \sum_i \left(\frac{\rho_{s,i}}{\rho_i} \right) & \text{if } \max_i \left(\frac{\rho_{s,i}}{\rho_i} \right) < 1 \end{cases} \quad (9.6)$$

For example, if the original material with a `DENSITY` of 500 kg/m³ is completely converted into a residue with a `DENSITY` of 1000 kg/m³, the thickness of the material layer will be half of the original.

Shrinking is prevented by setting `ALLOW_SHRINKING` to `F` on the `MATL` line. Swelling is prevented by setting `ALLOW_SWELLING` to `F` on the `MATL` line. By default, these flags are true. Shrinking/swelling does not take place if any of the materials with non-zero density has the corresponding flag set to false.

9.2.5 Multiple Solid Phase Reactions

The solid phase reaction represented by Fig. 9.3 is fairly simple—a single, homogeneous material is heated and gasified completely. Figure 9.5 depicts a more complicated case. Here, a solid material that contains 10 % (by mass) water and 90 % dry solid. The water evaporates in the neighborhood of 100 °C and the dry solid then pyrolyzes in the neighborhood of 300 °C, leaving 20 % of its mass behind in the form of a solid residue. The full set of parameters for this case are listed in `pyrolysis_2.fds`.



$$\begin{aligned} \frac{dY_{s,1}}{dt} &= -A_{1,1} Y_{s,1} \exp(-E_{1,1}/RT_s) & Y_{s,1}(0) &= 0.1 \\ \frac{dY_{s,2}}{dt} &= -A_{2,1} Y_{s,2} \exp(-E_{2,1}/RT_s) & Y_{s,2}(0) &= 0.9 \\ \frac{dY_{s,3}}{dt} &= -v_{s,2,1} \frac{dY_{s,2}}{dt} & Y_{s,3}(0) &= 0.0 \end{aligned}$$

$$\begin{aligned} T_{p,1,1} &= 100 + 273 \text{ K} & T_{p,2,1} &= 300 + 273 \text{ K} \\ r_{p,1,1} &= 0.0016 \text{ s}^{-1} & r_{p,2,1} &= 0.0012 \text{ s}^{-1} \\ v_{s,1,1} &= 0 & v_{s,2,1} &= 0.2 \\ \dot{T} &= 5 \text{ K/min} \end{aligned}$$

Figure 9.5: Normalized mass ($\sum Y_{s,i}$, blue curve) and mass loss rate ($\sum dY_{s,i}/dt$, red curve) for a material that contains 10 % water (by mass) that evaporates at a temperature of 100 °C, and 90 % solid material that pyrolyzes at 300 °C, leaving a 20 % (by mass) residue behind. Note that the numbered subscripts refer to the material component and the reaction, respectively. The system of ordinary differential equations that governs the transformation of the materials is shown at right.

The plot in Fig. 9.5 contains two sets of curves. The solid curves represent the solution of the set of equations computed using a Matlab ODE solver, and the underlying dashed curves are a “fit” of the Matlab solution using FDS. The `MATL` line for the solid material contains the following three parameters that define its decomposition:

```
&MATL ID = '...'
...
REFERENCE_TEMPERATURE = 300.
PYROLYSIS_RANGE = 100.
HEATING_RATE = 5. /
```

The `REFERENCE_TEMPERATURE` is the temperature in °C where the mass loss rate is at its peak. The `HEATING_RATE` is the linear temperature rise (°C/min) used in the TGA experiment, which is represented here by the Matlab solution. The `PYROLYSIS_RANGE` is the approximate “width” of the second red hump in Fig. 9.5, fit by inspection. That is, the value of 100 °C was chosen by trial and error. This is typically how one would choose kinetic parameters in FDS to match a given TGA curve.

9.2.6 The Heat of Reaction

Equation (9.2) describes the rate of the reaction as a function of temperature. Most solid phase reactions require energy; that is, they are *endothermic*. The amount of energy consumed, per unit mass of reactant that is converted into reaction products, is specified by the `HEAT_OF_REACTION(j)`. Technically, this is the enthalpy difference between the products and the reactant. A positive value indicates that the reaction is *endothermic*; that is, the reaction takes energy out of the system. Usually the `HEAT_OF_REACTION` is accurately known only for simple phase change reactions like the vaporization of water. For other reactions, it must be determined empirically (e.g., by differential scanning calorimetry).

9.2.7 Liquid Fuels

The evaporation rate of a liquid fuel is analogous to the convective heating rate in that the evaporation rate is a function of a mass transfer coefficient, h_m , much like thermal convection is a function of the heat transfer coefficient, h , discussed in Section 8.2.2³. The FDS Technical Reference Guide [1] provides further details on how the evaporation rate is computed.

The properties of a liquid fuel are given on the `MATL` line:

```
&REAC FUEL          = 'ETHANOL'
      CO_YIELD       = 0.001
      SOOT_YIELD      = 0.008 /

&MATL ID            = 'ETHANOL LIQUID'
      EMISSIVITY      = 1.
      NU_SPEC         = 1.
      SPEC_ID         = 'ETHANOL'
      HEAT_OF_REACTION = 837
      CONDUCTIVITY     = 0.17
      SPECIFIC_HEAT    = 2.44
      DENSITY         = 794
      ABSORPTION_COEFFICIENT = 1140
      BOILING_TEMPERATURE = 78.5 /

&SURF ID            = 'ETHANOL POOL'
      COLOR           = 'YELLOW'
      MATL_ID         = 'ETHANOL LIQUID'
      THICKNESS       = 0.1 /
```

The inclusion of `BOILING_TEMPERATURE` on the `MATL` line tells FDS to use its liquid pyrolysis model. It also automatically sets `N_REACTIONS=1`, that is, the only “reaction” is the phase change from liquid to gaseous fuel. Thus, `HEAT_OF_REACTION` in this case is the latent heat of vaporization. The thermal conductivity, density and specific heat are used to compute the loss of heat into the liquid via conduction using the same one-dimensional heat transfer equation that is used for solids. Obviously, the convection of the liquid is important, but is not considered in the model. The `ABSORPTION_COEFFICIENT` denotes the absorption in

³As with the convective heat transfer coefficient, there is an option to specify a fixed `MASS_TRANSFER_COEFFICIENT` (m/s) on the `SURF` line that describes a liquid pool.

depth of thermal radiation into the liquid. Liquids do not just absorb radiation at the surface, but rather over a thin layer near the surface. Its effect on the burning rate can be significant.

In this example, 'ETHANOL' is a *known* species; that is, it is listed in Appendix A. For this reason, only its CO and soot yield need to be specified. If the species is not known, then you must furnish additional information. Sec. 12.1.3 contains details, but in brief, only the molecular weight of the gas species is of relevance for the liquid pool evaporation model. If the liquid has multiple components, like gasoline or kerosene, individual `MATL` line can be provided for each component. A single gas phase fuel species can still be used by having each `MATL` line contain the same `SPEC_ID`. If this is the case, provide the liquid component molecular weights, `MW`, with units of g/mol on the `MATL` lines. These unique molecular weights are used when computing the evaporation rates of the individual liquid components.

Evaporation of a Pure Liquid

A useful sample case found in the `Pyrolysis` folder is called `methanol_evaporation.fds`. A 1 m by 1 m pan filled with methanol at $T_\infty = 20^\circ\text{C}$ is exposed to a uniform heat flux, $\dot{q}'' = 20\text{ kW/m}^2$. The boiling temperature of methanol is $T_b = 64.65^\circ\text{C}$, its specific heat, $c = 2.48\text{ kJ/(kg}\cdot\text{K)}$, and heat of vaporization, $h_v = 1099\text{ kJ/kg}$. At steady state, the heat balance at the pool surface is

$$\dot{q}_{\text{total}}'' - \dot{q}_c'' = \dot{m}'' h_v T_s \quad (9.7)$$

where \dot{q}_c'' is the heat being conducted away from the surface. If \dot{q}_c'' is made zero, which can be done by using `BACKING='INSULATED'` and a high thermal conductivity, then the pool surface temperature, T_s , will approach the boiling temperature, T_b . In this example, since there is no gas phase combustion reaction (`REAC`), no burning occurs. The left hand plot in Fig. 9.6 displays the computed evaporation rate, \dot{m}'' , versus the ideal, $\dot{q}_{\text{total}}''/h_v T_b$. The former approaches the latter as all of the absorbed energy is used to evaporate the liquid. The right hand plot shows the computed liquid surface temperature versus the liquid boiling temperature.

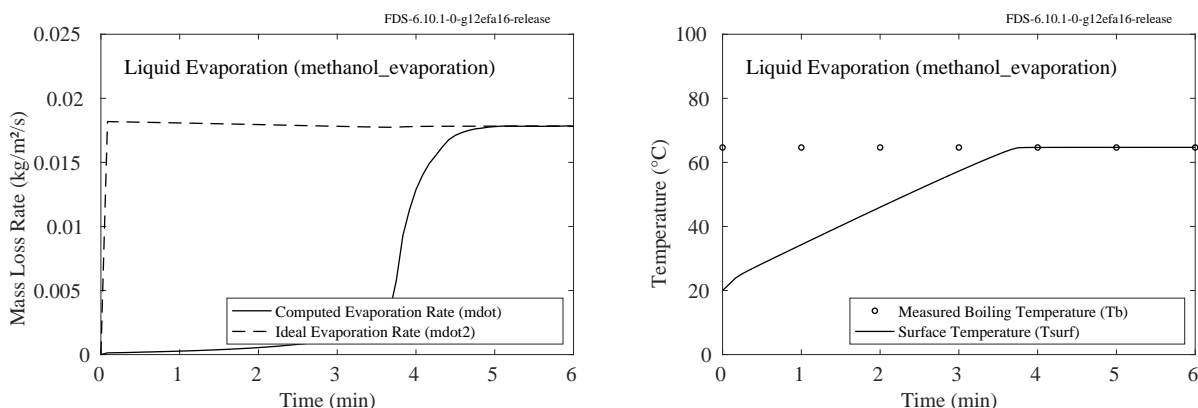


Figure 9.6: (Left) Mass loss rate of the methanol versus its expected steady state value. (Right) Computed surface temperature versus the liquid boiling temperature.

Liquid Mixtures

Most liquid fuels of interest in fire protection engineering are not pure liquids, but rather blends, like gasoline. Each component liquid has a unique boiling point, heat of vaporization, molecular weight, and specific heat that affect its evaporation rate. It is possible to specify the individual liquid components, but have them all evaporate to form the same gaseous fuel vapor. The reason for this is to save on computing cost. If

each component were to evaporate to form a unique gaseous fuel species, a separate transport equation and combustion reaction would be needed for each. Consider the following example—a mixture of hydrocarbon fuels is mixed with water forming an emulsion. The pool is specified as follows:

```
&SPEC ID='WATER VAPOR' /
&REAC FUEL='N-HEXANE' /
&SURF ID      = 'POOL'
      COLOR    = 'YELLOW'
      MATL_ID(1,1:6) = 'HEXANE','HEPTANE','OCTANE','DECANE','BENZENE','WATER'
      MATL_MASS_FRACTION(1,1:6) = 0.521,0.054,0.063,0.023,0.200,0.139
      THICKNESS = 0.002 /
```

Water is to evaporate and form 'WATER VAPOR', but the other components are all to form 'N-HEXANE'. On each MATL line for each liquid component, either 'WATER VAPOR' or 'N-HEXANE' is declared as the SPEC_ID, along with the BOILING_TEMPERATURE, MW, SPECIFIC_HEAT, and HEAT_OF_REACTION (i.e. the heat of vaporization). Additionally, a HEAT_OF_COMBUSTION may be specified on the MATL line to indicate that the liquid component has a different heat of combustion than n-hexane and FDS should adjust its evaporation rate accordingly.

An example demonstrating evaporation of a liquid mixture is found in the `Pyrolysis` folder and called `liquid_mixture.fds`. In this example, a 1 m by 1 m by 2 mm deep pool containing a mixture of hydrocarbon liquids and water is completely evaporated to form water vapor and the surrogate gaseous fuel vapor, n-hexane. Figure 9.7 displays the mass of water and fuel vapor as a function of time.

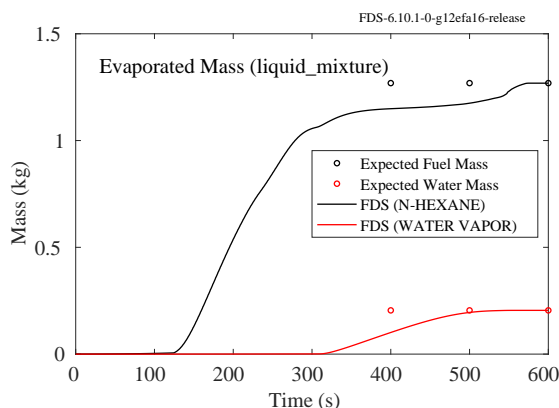


Figure 9.7: Evaporated mass of water and fuel species from an emulsified liquid fuel.

9.2.8 Fuel Burnout

The thermal properties of a solid or liquid fuel determine the length of time for which it can burn. In general, the burnout time is a function of the mass loss rate, \dot{m}'' , the density, ρ_s , and the layer thickness, δ_s :

$$t_b = \frac{\rho_s \delta_s}{\dot{m}''} \quad (9.8)$$

However, each type of pyrolysis model handles fuel burnout in a slightly different way. These differences will be highlighted in the individual sections below.

Solid Fuel Burnout When the Burning Rate is Specified

If you *specify* the burning rate using `HRRPUA` (Heat Release Rate Per Unit Area) or `MLRPUA` (Mass Loss Rate Per Unit Area), the solid surface will continue to burn at the specified rate indefinitely with no fuel burnout. You may use `RAMP_Q` to stop the burning at a desired time. You can estimate this “burnout time” for a surface using the heat of combustion, ΔH , material density, ρ_s , material thickness, δ_s , and `HRRPUA`, \dot{q}_f'' :

$$t_b = \frac{\rho_s \delta_s \Delta H}{\dot{q}_f''} \quad (9.9)$$

The burnout time is not calculated and applied automatically because there are instances where the underlying solid is not considered fuel; that is, the burning rate and duration are not determined by the composition of the solid surface.

An alternative to ramping the HRR up and down is to specify `BURN_DURATION` (s) on the `SURF` line. This is the time duration after ignition when the fire will extinguish. The default value of this parameter is a very large number, meaning that by default a fire with a specified HRR will burn indefinitely.

Solid Fuel Burnout When the Burning Rate is Not Specified

If you are using a pyrolysis model in which the `MATL` lines have `N_REACTIONS` greater than or equal to 1, the burnout time of the pyrolyzing solid fuel is calculated automatically by FDS based on the layer `THICKNESS`, component `DENSITY`, and the calculated reactions rates of the materials.

Liquid Fuel Burnout

The burnout time of a liquid fuel is calculated automatically based on the liquid layer `THICKNESS`, liquid `DENSITY`, and the calculated burning rate.

9.2.9 Solid Fuels that can Burn Away

If a combustible solid is to disappear from the simulation once it is consumed, set `BURN_AWAY=T` on the `SURF` line that is applied to the `OBST`. The solid object disappears from the calculation cell by cell as the mass contained within each solid cell is consumed either by the pyrolysis reactions or by the prescribed HRR. The following issues should be kept in mind when using `BURN_AWAY`:

- Use `BURN_AWAY` cautiously. If an object has the potential of burning away, a significant amount of extra memory has to be set aside to store additional information for newly exposed surfaces.
- The heat conduction into an `OBST` that can `BURN_AWAY` is limited to 1-D. On the `SURF` line, specify the `THICKNESS` to be approximately that of a single cell, and also set `BACKING='VOID'`. It is assumed that the solid is relatively well-insulated and that the in-depth temperature profile drops off relatively quickly at the surface. When a cell is removed, the newly exposed surface is initially at ambient temperature but heats up quickly at the surface.
- If the volume of the obstruction changes because it has to conform to the uniform mesh, FDS does *not* adjust the burning rate to account for this as it does with various quantities associated with areas, like `HRRPUA`.
- A parameter called `BULK_DENSITY` (kg/m³) can be specified on the `OBST` line to designate the *combustible* mass of the solid object. The calculation uses the user-specified object dimensions, not those of the mesh-adjusted object. This parameter over-rides all other parameters with which a combustible mass

would be calculated. Note that without a `BULK_DENSITY` specified, the total amount of mass burned will depend upon the grid resolution. The use of the `BULK_DENSITY` parameter ensures a specific fuel mass per unit volume that is independent of the grid resolution. Note that in the event that the solid phase reaction involves the production of solid residue (like char or ash), the `BULK_DENSITY` refers to the mass of the solid that is converted to gas upon reaction. You can visualize the decrease in the bulk density of a burning solid using the output quantity 'BULK DENSITY' with a slice (`SLCF`) or device (`DEVC`).

- If `BURN_AWAY` is prescribed on a `SURF` line, that single `SURF_ID` should be applied to the entire `OBST`, not to specific faces, because it is unclear how to remove solid obstructions that have different `SURF_IDS` on different faces. An exception to this rule is where a `BULK_DENSITY` is applied to the `OBST`, in which case the `OBST` will disappear when the mass of fuel designated by `BULK_DENSITY` is consumed rather than that of any particular face.
- By default, newly exposed surfaces take on the same `SURF_ID` that has been applied to the original obstruction. However, you can specify an optional `SURF_ID_INTERIOR` on the `OBST` line to define surface properties of newly exposed surfaces. For example, an upholstered cushion might have a surface layer of fabric whereas a newly exposed grid cell within the cushion will not.
- To compensate for the inaccurate obstruction area, an additional parameter `AREA_MULTIPLIER` can be added on the `SURF` line. It will multiply all mass and energy fluxes with a user-specified factor. Check that the `BURN_AWAY` works as expected if these two features are combined.
- The mass of the object is based on the densities of all material components (`MATL`), but it is only consumed by mass fluxes of the *known* species. If the sum of the gaseous yields is less than one, it will take longer to consume the mass.

Examples

Simple examples demonstrating how solid fuels can be forced to disappear from the domain are labeled `Fires/box_burn_away`. These are examples of a solid block of material that is pyrolyzed until it is completely consumed. The heat flux is generated by placing hot surfaces around the box. There is no combustion because the `AUTO_IGNITION_TEMPERATURE` has been set to a very high value. The properties of the solid material are chosen simply to assure a quick calculation. The objective is to ensure that the pyrolyzed fuel mass is consistent with the mass of the original block. The block is 0.4 m on a side, with a density of 20 kg/m³. The total mass is:

$$(0.4)^3 \text{ m}^3 \times 20 \text{ kg/m}^3 = 1.28 \text{ kg} \quad (9.10)$$

Case 1 (`box_burn_away1`) The solid material undergoes a single-step pyrolysis reaction at 200 °C that converts it all to fuel gas. Figure 9.8 displays the evolution of the fuel gas.

Case 2 (`box_burn_away2`) This case is similar to Case 1, except that the solid material vaporizes to form an inert 'GAS' with a molecular weight of 50 g/mol.

Case 3 (`box_burn_away3`) The pyrolysis rate is specified implicitly via `HRRPUA`, even though the fuel gas does not burn.

Case 4 (`box_burn_away4`) This case is similar to Case 3, but the heat of combustion for the solid material is set to a different value from that of the primary gas phase fuel, ethylene. The solid material has a specified heat of combustion of 30,000 kJ/kg while ethylene has a specified value of 40,000 kJ/kg. Figure 9.8 shows that the amount of ethylene generated is $0.75 \times 1.28 \text{ kg} = 0.96 \text{ kg}$. Because there is only one gas phase reaction, the amount of ethylene gas produced generates an equivalent amount of heat as the solid material would with its lower heat of combustion.

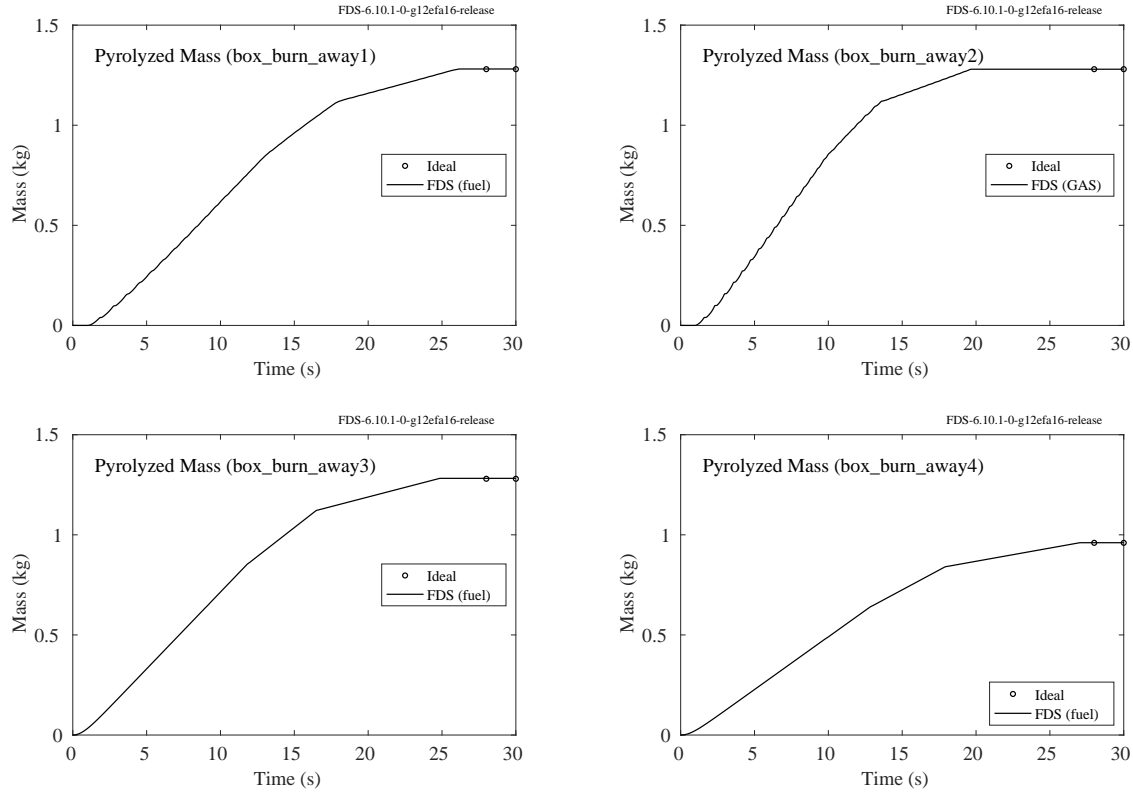


Figure 9.8: Output of `box_burn_away` test cases 1-4.

Case 5 (`box_burn_away5`) This case is a repeat of Case 1, but with `N_SIMPLE_CHEMISTRY_REACTIONS` set to 2. The results should match Case 1 because the creation of fuel gas should be based upon the total heat of combustion for 'METHANE' and not the heat of combustion for the first reaction.

Case 6 (`box_burn_away6`) The solid material generates two fuel gases with equal masses, $m_1 = m_2 = 0.64 \text{ kg}$, and both fuel gases have the same heat of combustion, $\Delta H_1 = \Delta H_2 = 50,000 \text{ kJ/kg}$. However, the gas phase reactions assign different heats of combustion, $\Delta H'_1 = 50,000 \text{ kJ/kg}$ and $\Delta H'_2 = 25,000 \text{ kJ/kg}$. The equivalent masses of gases generated, m'_1 and m'_2 , are determined via $m_i \Delta H_i = m'_i \Delta H'_i$ for $i = 1, 2$. As a result, the second fuel will have twice as much effective mass in the gas phase.

Case 7 (`box_burn_away7`) This case demonstrates that it is possible to burn and make disappear a thin, zero-cell thick obstruction. Versions of FDS prior to 6.7.2 did not allow this to happen. In the test case, a 0.4 m by 0.4 m by 0.01 m thin slab with a density of 20 kg/m³ is heated and evaporates away, creating 0.032 kg of fuel gas.

Case 8 (box_burn_away8) This case is similar to Case 4, except that the burning rate is specified using the parameter MLRPUA.

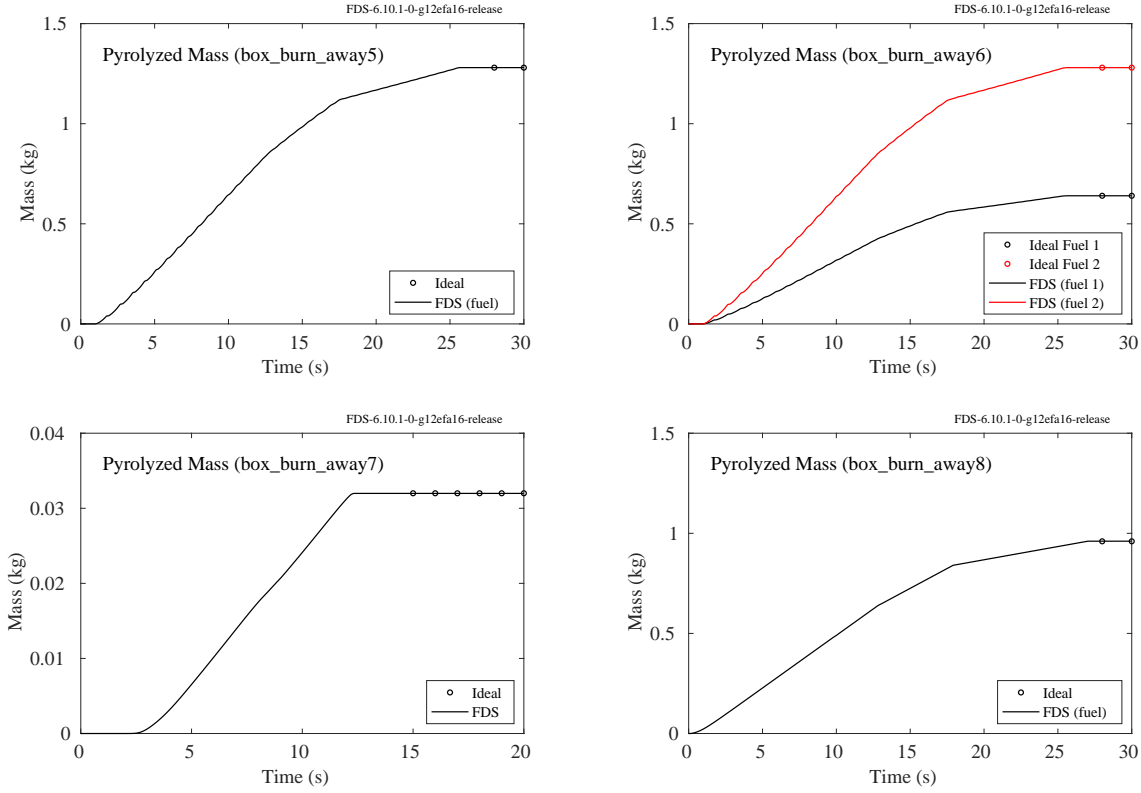


Figure 9.9: Output of box_burn_away test cases 5-8.

Case 9 (box_burn_away9) A solid cube of foam with side length $L = 0.4$ m and density $\rho_1 = 20$ kg/m³ pyrolyzes with a residue fraction of $v_1 = 0.5$. The block is covered by a $\delta = 0.002$ m fabric layer with density $\rho_2 = 50$ kg/m³ and residue fraction $v_2 = 0.2$. The 3-D heat transfer algorithm is invoked. The expected mass of gaseous fuel is:

$$m = L^2 \left((L - 2\delta)(1 - v_1)\rho_1 + 2\delta(1 - v_2)\rho_2 \right) = 0.6592 \text{ m} \quad (9.11)$$

Note that the 3-D heat conduction equation is solved via 1-D segments in all three coordinate directions. Each segment accounts for the layers that are perpendicular to its direction, but not those that are parallel. As a result, the pyrolyzed mass in this example is less than expected if one were to consider the actual 3-D object.

Case 10 (box_burn_away10) This case is the same as box_burn_away1 with the addition of a 1 cm thick plate attached to the side of the solid made of the same material. The plate is 0.6 m by 0.6 m by 0.01 m thick adding 0.072 kg of additional fuel mass.

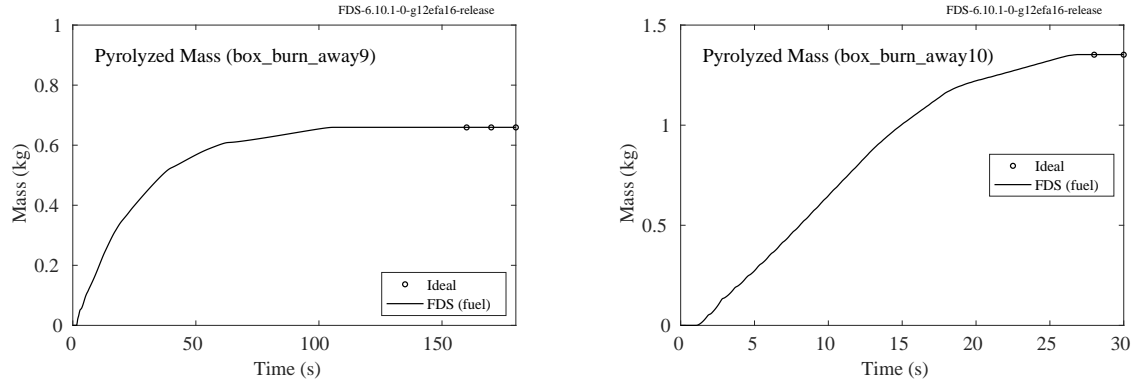


Figure 9.10: Output of `box_burn_away9` and `box_burn_away10` test cases.

Case 11 (`box_burn_away11`) The “box” in this case consists of 0.4 m by 0.4 m by 1 cm plates forming a collection of 10 cm cubical compartments. There are 15 plates in all, each with a density of 20 kg/m^3 . The total mass is 0.48 kg.

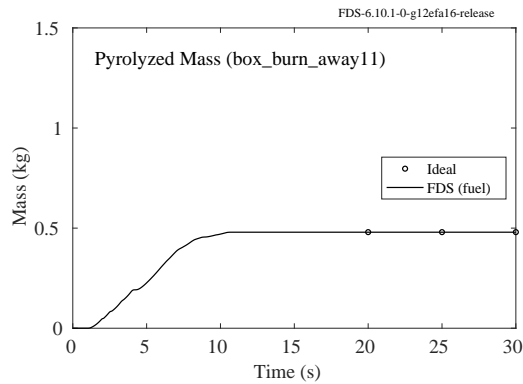


Figure 9.11: Output of `box_burn_away11` test case.

Case 1, 2D (`box_burn_away_2D` and `box_burn_away_2D_residue`) Case 1 is repeated in two dimensions. In the latter case, only half of the mass is converted to fuel, leaving behind a residue that is 50 % of the original mass. The box is forced to burn away by setting the `BULK_DENSITY` to 10 kg/m^3 . This is the *combustible* mass. These two cases exhibit a fictitious increase in solid mass when new unburned surfaces are exposed as entire mesh cells disappear. The increased mass is just an artifact of reporting the residual solid mass as the product of surface density and surface area. Both the final solid mass and the gaseous degradation products should match the expected values at the end of the simulation. The results are shown in Fig. 9.12.

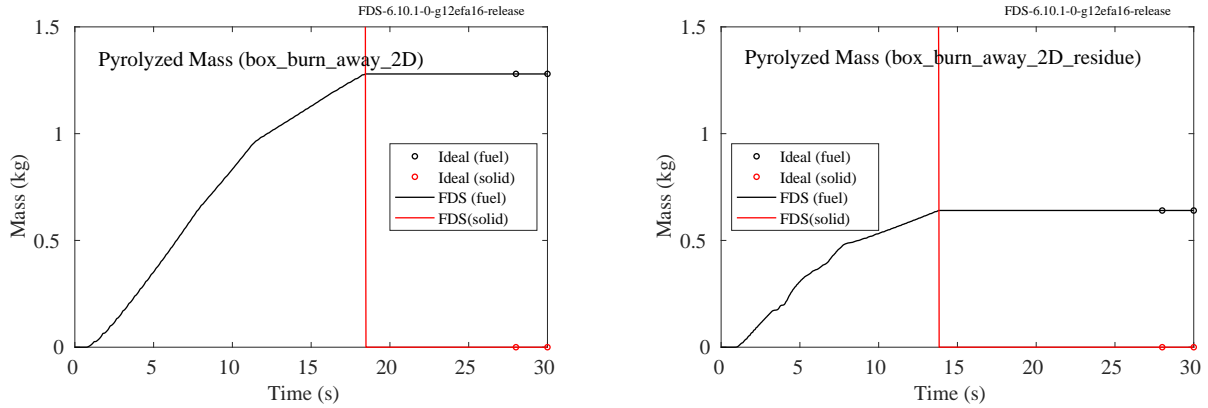


Figure 9.12: Output of box_burn_away_2D test cases.

9.3 Testing a Pyrolysis Model

The `SURF` and `MATL` lines describing the pyrolysis of real materials consist of a combination of empirical and fundamental properties, often originating from different sources. How do you know that the various property values and the associated thermo-physical model in FDS constitute an appropriate description of the solid? For a full-scale simulation, it is hard to untangle the uncertainties associated with the gas and solid phase routines. However, you can perform a simple check of any set of solid phase model by essentially turning off the gas phase. In the following sections, guidance is provided on how to perform a quick simulation of the cone calorimeter and bench-scale measurements like thermal gravimetric analysis (TGA), differential scanning calorimetry (DSC), and micro-combustion calorimetry (MCC).

9.3.1 Simulating the Cone Calorimeter

This section describes how to set up a simple model of the cone calorimeter or similar apparatus. This is not a full 3-D simulation of the apparatus, but rather a 1-D simulation of the solid phase degradation under an imposed external heat flux. While a full 3-D simulation of the cone heater and sample holder can be created in FDS, such a simulation would take some time to complete and would not disentangle issues with the solid phase model from uncertainties in the gas phase. It is worthwhile to perform a quick simulation like the one described here to test the solid phase model only.

1. Create a trivially small mesh, just enough to allow FDS to run. Since the gas phase calculation is essentially being shut off, you just need nine cells total (`IJK=3, 1, 3`) for the pressure solver to function properly.
2. On the `MISC` line, set `SOLID_PHASE_ONLY=T` to turn off the gas phase fluid flow computation. This reduces the CPU time. Note that convective and radiative heat transfer to/from the surface is still applied using the ambient temperature as the assumed gas and far-field temperature. The `HEAT_TRANSFER_COEFFICIENT` is either specified or computed from the gas phase thermal conductivity divided by the wall-normal grid spacing. It is best to specify it explicitly.
3. Create `SPEC` lines to list any gas species created in the pyrolysis process. A `REAC` line is not needed, as there is no gas phase combustion allowed.
4. On the `TIME` line, set `WALL_INCREMENT=1` to force FDS to update the solid phase every time step (normally it does this every other time step), and set `DT` to a value that is appropriate for the solid phase

calculation. Since there is no gas phase calculation that will limit the time step, it is best to control this yourself.

5. Generate `MATL` lines, plus a single `SURF` line, as you normally would, except add `EXTERNAL_FLUX` to the `SURF` line. This is simply a “virtual” source that heats the solid. Think of this as a perfect radiant panel or conical heating unit. You can control the `EXTERNAL_FLUX` using either `TAU_EF` or `RAMP_EF`. This is useful if you want to ramp up the heat flux following ignition to account for the additional radiation from the flame. See Sec. 11.1 for more details about ramps.
6. On the `SURF` line, set a gas temperature, `TMP_GAS_FRONT` ($^{\circ}\text{C}$), and optionally a `HEAT_TRANSFER_COEFFICIENT` ($\text{W}/(\text{m}^2\cdot\text{K})$), allowing you to control the convective heat flux from gas to surface and vice versa.
7. Assign the `SURF_ID` to a `VENT` that spans the bottom of the computational domain. Create `OPEN` vents on all other faces of the computational domain.
8. Add solid phase output devices to the solid surface, like `'WALL TEMPERATURE'`, `'TOTAL HEAT FLUX'`, `'GAUGE HEAT FLUX'`, and `'WALL THICKNESS'`. Use these to track the condition of the solid as a function of time. The generation rate of the various gases is output via the quantity `'MASS FLUX'` along with the appropriate `SPEC_ID`. Do not specify the quantity `'BURNING RATE'` because FDS assumes that this is specific for fuel gas, and in this exercise there is no fuel gas.

Compare your results to measurements made in a bench-scale device, like the cone calorimeter. Keep in mind, however, that the calculation and the experiment are not necessarily perfectly matched. The calculation is designed to eliminate uncertainties related to convection, combustion, and apparatus-specific phenomena. Below is an FDS input file that demonstrates how you can test a candidate pyrolysis model by running very short calculations. The simulation only involves the solid phase model. Essentially, the gas phase calculation is shut off except for the imposition of a $50 \text{ kW}/\text{m}^2$ “external” heat flux. The solid in this example is a 8.5 mm thick slab of PMMA. For more details, see the FDS Validation Guide under the heading “FAA Polymers.”

```
&HEAD CHID='pmma_example', TITLE='Black PMMA at 50 kW/m2' /
&MESH IJK=3,3,3, XB=-0.15,0.15,-0.15,0.15,0.0,0.3 /
&TIME T_END=600., WALL_INCREMENT=1, DT=0.05 /
&MISC SOLID_PHASE_ONLY=T /
&SPEC ID='METHANE' /
&MATL ID='BLACKPMMA'
    ABSORPTION_COEFFICIENT=2700.
    N_REACTIONS=1
    A(1) = 8.5E12
    E(1) = 188000
    EMISSIVITY=0.85
    DENSITY=1100.
    SPEC_ID='METHANE'
    NU_SPEC=1.
    HEAT_OF_REACTION=870.
    CONDUCTIVITY = 0.20
    SPECIFIC_HEAT = 2.2
&SURF ID='PMMA SLAB'
    COLOR='BLACK'
    BACKING='INSULATED'
    MATL_ID='BLACKPMMA'
    THICKNESS=0.0085
    EXTERNAL_FLUX=50 /
```

```

&VENT XB=-0.05,0.05,-0.05,0.05,0.0,0.0, SURF_ID = 'PMMA SLAB' /
&DUMP DT_DEVC=5. /
&DEVC XYZ=0.0,0.0,0.0, IOR=3, QUANTITY='WALL TEMPERATURE', ID='temp' /
&DEVC XYZ=0.0,0.0,0.0, IOR=3, QUANTITY='MASS FLUX', SPEC_ID='METHANE', ID='MF' /
&DEVC XYZ=0.0,0.0,0.0, IOR=3, QUANTITY='WALL THICKNESS', ID='thick' /
&TAIL /

```

9.3.2 Simulating Bench-scale Measurements like the TGA, DSC, and MCC

There are a number of techniques to measure the thermo-physical properties of a solid material. Most of these involve heating a very small sample at a relatively slow, linear rate. In this way, thermal conduction is minimized and the sample can be considered thermally-thin. FDS has a special feature that mimics thermogravimetric analysis (TGA), differential scanning calorimetry (DSC), and micro-combustion calorimetry (MCC) measurements. To use it, set up your input file as you normally would. Then, add the flag `TGA_ANALYSIS=T` to the `SURF` line you want to analyze. Only one `SURF` line can be analyzed at a time. Doing multiple `SURF` lines will require multiple runs. The linear heating rate (K/min) and final sample temperature (°C) are respectively specified using `TGA_HEATING_RATE` and `TGA_FINAL_TEMPERATURE`. The default values are 5 K/min and 800 °C. The initial temperature is `TMPA`. Note that this feature is only appropriate for a `SURF` line that describes a thermally-thick sample consisting of a single layer with multiple reacting components. For example, the following `SURF` line describes a material that consists of three components:

```

&SURF ID = 'Cable Insulation'
      TGA_ANALYSIS = T
      TGA_HEATING_RATE = 60.
      THICKNESS = 0.005
      MATL_ID(1,1) = 'Component A',
      MATL_ID(1,2) = 'Component B',
      MATL_ID(1,3) = 'Component C',
      MATL_MASS_FRACTION(1,1:3) = 0.26,0.33,0.41 /

```

The two `TGA` entries will force FDS to perform a numerical version of the TGA, DSC and MCC measurements. The `THICKNESS` and other boundary conditions on the `SURF` line will be ignored. After running the analysis, which only takes a second or two, FDS will then shut down without running the actual simulation. To run the simulation, either remove the `TGA` entries or set `TGA_ANALYSIS` to `F`.

Note: If applying `TGA_ANALYSIS` to a case where the `SURF` is associated with a `PART` class yet to be inserted into the calculation, FDS may not find the `SURF` and hence throw an error. In this case, create a simple case with a single particle that is inserted at the start.

The result of the `TGA_ANALYSIS` is a single comma-delimited file called `CHID_tga.csv`. The first and second columns of the file consist of the time and sample temperature. The third column is the normalized sample mass; that is, the sample mass divided by its initial mass. The following columns list the mass fractions of the individual material components. The next column is the total mass loss rate, in units of s^{-1} , followed by the mass loss rates of the individual material components. The next column is the heat release rate per unit mass of the sample in units of W/g, typical of an MCC measurement. The final two columns are the rate of heat absorbed by the sample normalized by its *original* mass and by its *instantaneous* mass, respectively. The former is labeled `DSC` and the latter `STA`, both in units of W/g. `STA` means Simultaneous Thermal Analysis, where a TGA and DSC measurement are made simultaneously, in which case the instantaneous mass of the sample is available.

The timestep used in the TGA analysis can be controlled with `TGA_DT` on the `SURF` line. The output spacing for temperature can be set with `TGA_DUMP` on the `SURF` line.

The TGA, MCC or DSC/STA output can be multiplied by `X_CONVERSION_FACTOR`, where `X` is TGA, MCC, or DSC, respectively. For example, the DSC output assumes that an endothermic reaction is represented by a positive peak, and an exothermic reaction by a negative peak. To flip this around, set `DSC_CONVERSION_FACTOR=-1`.

Details of the output quantities are discussed in Sec. 22.10.17. Further details on these measurement techniques and how to interpret them are found in the FDS Verification Guide [4].

Example

Results for an analysis (`Pyrolysis/tga_analysis.fds`) of a simplified version of wood are shown in Fig. 9.13. The sample, referred to as “wet wood”, contains 10 % “water” by mass. It is heated at a rate of $\beta = 5/60$ K/s and undergoes three reactions: the evaporation of “water”, the conversion of “dry wood” to “char”, and the conversion of “char” to “ash”.

The upper two plots in Fig. 9.13 show the results of the TGA analysis; that is, the decrease in mass as a function of sample temperature. The left plot shows the total sample mass and the right shows the individual components. Below these are two plots that show the mass loss rates of the total sample and its components. These plots are simply the (negative) values of the first derivative of the upper plots.

The lower left plot in Fig. 9.3.2 shows the results of the MCC analysis; that is, the gas phase combustion heat release rate per unit mass of the original sample. Note that this MCC plot does not include evidence of the evaporation of water because water vapor does not combust. The integral (with respect to time) under the MCC curve yields $Q = 12680$ J/g. The heat of combustion of the “cellulose,” which is the assumed gas phase fuel resulting from the pyrolysis of dry wood and char, is $h_c = 14988$ J/g. The dry wood and char constitute $Y = 0.846$ of the original sample, thus $h_c Y \approx Q$.

The lower right plot Fig. 9.3.2 shows the results of the DSC/STA analysis. The black curve (DSC) represents the rate of heat absorbed per unit mass of the original sample. The red curve represents the rate of heat absorbed per unit mass of the pyrolyzing sample. The three peaks represent the endothermic reactions where heat from the hot gas is used to evaporate water or pyrolyze the wood and char. The plateaus between the reaction peaks represent the heating of the solid only. For example, at a temperature of 500 °C, only ash remains and the value of the DSC curve is $\dot{q}_{DSC}(500) \approx 0.0045$ W/g from which the value of the specific heat can be computed:

$$c_p = \frac{\dot{q}_{DSC}(500)}{Y(500)\beta} \approx 1 \text{ kJ/(kg K)} \quad (9.12)$$

The term $Y(500) \approx 0.054$ in the denominator is the ratio of mass of the ash at 500 °C to the mass of the original wet wood. For the STA curve, the specific heat is obtained simply by dividing the STA value $\dot{q}_{STA}(500) \approx 0.0834$ W/g by the heating rate, β .

$$c_p = \frac{\dot{q}_{STA}(500)}{\beta} \approx 1 \text{ kJ/(kg K)} \quad (9.13)$$

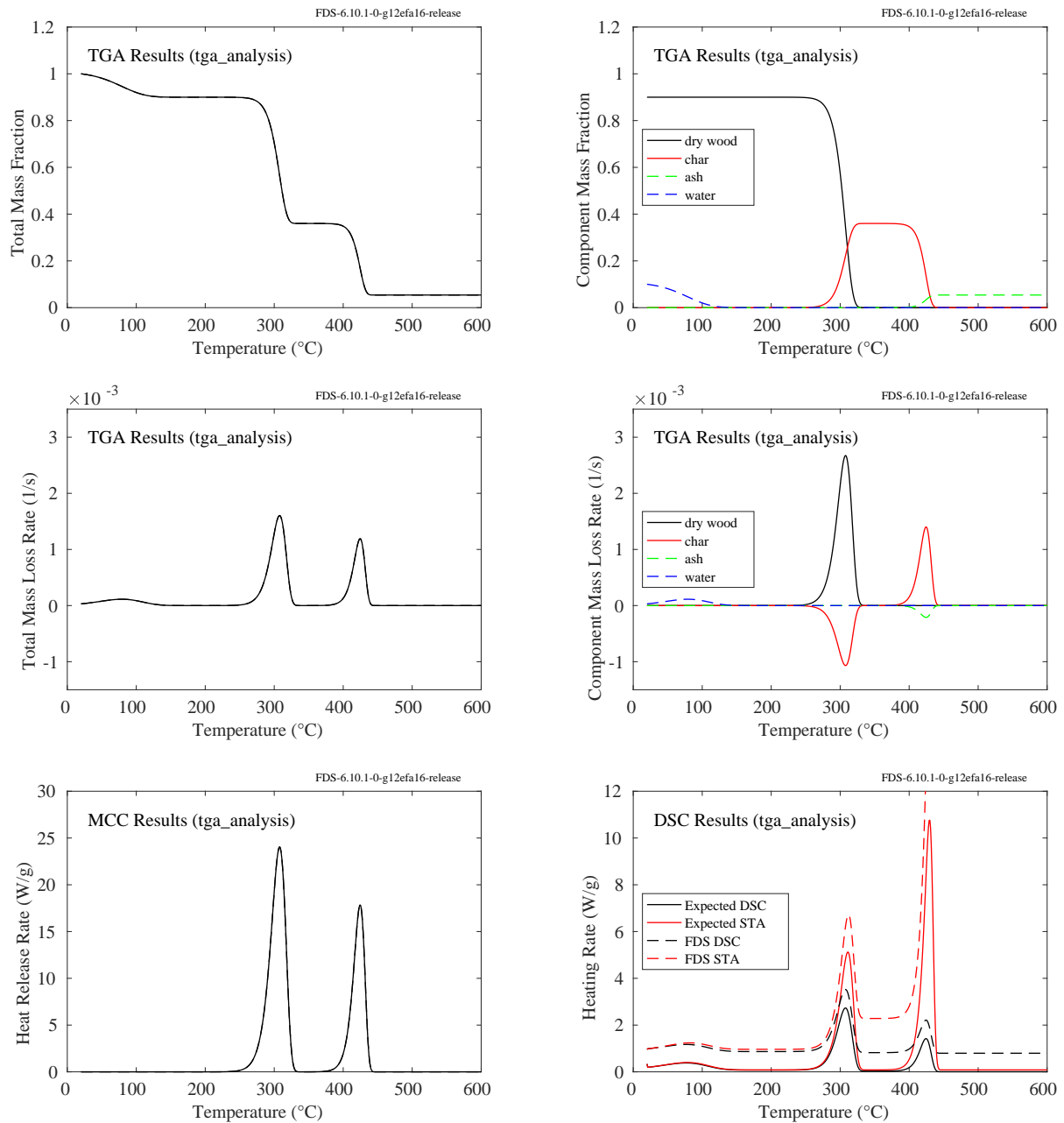


Figure 9.13: Sample results of a tga_analysis.

9.4 Pyrolysis and Energy Conservation

The `HEAT_OF_REACTION` is the energy gained or lost in converting the material to residue(s) and/or gas(es). In other words the `HEAT_OF_REACTION` is the enthalpy of the products at the current solid temperature minus the enthalpy of the initial material at the current solid temperature. For total internal energy to be conserved, this must hold true. A challenge is that while the heat of reaction may be known for a solid material, detailed knowledge of the material enthalpy or the enthalpy of any residue or gaseous products is often not known. In cases where a reference enthalpy for a material is known, you can input on the `MATL` line the `REFERENCE_ENTHALPY` in kJ/kg and the `REFERENCE_ENTHALPY_TEMPERATURE` in °C.

FDS will attempt to adjust all material enthalpies so that internal energy is conserved. For each material reaction a linear equation is defined:

$$H_x(T_{\text{ref}}) + H_{x,\text{adj}} + H_{x,\text{r}}(T_{\text{ref}}) = \sum_{i=1}^{N_{\text{residues}}} v_i (H_i(T_{\text{ref}}) + H_{i,\text{adj}}) + \sum_{j=1}^{N_{\text{species}}} v_j H_j(T_{\text{ref}}) \quad (9.14)$$

where $H(T)$ is the enthalpy of a material or gas, H_r is the `HEAT_OF_REACTION`, T_{ref} is the temperature at which the `HEAT_OF_REACTION` was specified for, v is a yield, and H_{adj} is a constant that forces the correct enthalpy to balance the reactions. During initialization, $H(T)$ for a material is initialized using the defined `SPECIFIC_HEAT` or `ramp`. $H(T)$ for a gas is defined per the discussion in Sec. 12.1.2. No adjustment parameter is included for gases since typically pyrolysis is either producing a predefined species such as carbon dioxide or water vapor or it is producing a fuel species use on `REAC`. For a predefined species, the enthalpy is known. When a non-predefined species is used on `REAC`, FDS adjusts the species enthalpy so that the correct `HEAT_OF_COMBUSTION` is obtained. This means it cannot be adjusted again to balance a material reaction.

The result is a system of linear equations where the H_{adj} values are the unknowns. This results in a matrix whose row count is the total number of reactions over all materials, and whose column count is the total number of materials that are either a reactant or a residue. The number of rows and columns may not be equal to one another. If they are equal, provided the matrix is not singular, then a standard matrix solution can be done. If there are more equations than materials, then the system is overdetermined, and there may not be a solution that conserves energy. In this case, a least-squares solution of the system is performed to find the best possible values for the H_{adj} values. A warning message will be written in this case. If there are more materials than reactions, then the system is under-determined, and there will be an infinite number of solutions. In this case, the matrix is solved to yield a minimal solution for H_{adj} values. That is it attempts to find values for H_{adj} where the vector of values has the smallest magnitude.

In the case of a singular matrix, a warning is written that no adjustment is done. A singular matrix happens when a row in the matrix is a linear combination of one or more other rows. If this happens, there is likely an error in one of the `MATL` definitions.

The value of T_{ref} used for a reaction is the value specified by the `REFERENCE_TEMPERATURE` for that reaction. If no value is given (when the reaction is defined using `A` and `E`), then FDS will do a virtual TGA using just the single reaction for the single material. The temperature where the peak reaction rate occurs is used for T_{ref} . Since in most cases the specific heat of a material and the specific heats of its residues and product gases are not the same, the heat of reaction is temperature dependent. FDS will create a temperature dependent array for the heat of reaction that accounts for this where the value at T_{ref} is fixed to the value specified with `HEAT_OF_REACTION`.

This process of adjusting enthalpies can be skipped by setting `ADJUST_H=F` on a `MATL` line. This should be done when material reactions do not represent actual chemical reactions. If no `HEAT_OF_REACTION` is specified for a `MATL` with a reaction then `ADJUST_H=F` will be set. Note that this means if a reaction has `HEAT_OF_REACTION` that is actually zero, and enthalpy adjustment is desired, then set `HEAT_OF_REACTION` to 0 in the input file for that reaction.

If the sum of the yields is less than 1, then for the purpose of solving for the H_{adj} values, FDS will assume that the missing mass is a material with the same specific heat as the original material.

If a material has a reaction where `N_O2` is set and the adjustment process needs to determine the `REFERENCE_TEMPERATURE`, then a value for the oxygen concentration is needed. This value will default to the ambient oxygen concentration for the simulation; however, it can be overridden on a material basis by specifying `X_O2_PYRO` for that material. This may be needed if, for example, the kinetics and heat of reaction(s) for a material were developed using a specific oxygen concentration.

Chapter 10

Ventilation

This chapter explains how to model a ventilation system. There are two ways to do this. First, is to explicitly specify air flow rates into and out of compartments, see Sec. 10.1 for a description of simple velocity boundary conditions. Second, is to model airflow rates using an HVAC system, see Sec. 10.2 for a description of modeling HVAC systems.

10.1 Simple Vents, Fans and Heaters

The ventilation system of individual compartments within a building is described using velocity *boundary conditions*. For example, fresh air can be blown into, and smoke can be drawn from, a compartment by specifying a velocity in the *normal* direction to a solid surface. However, there are various other facets of velocity boundary conditions that are described below.

10.1.1 Simple Supply and Exhaust Vents

The easiest way to describe a supply or exhaust fan is to specify a `VENT` on a solid surface, and designate a `SURF_ID` with some form of specified velocity or volume flow rate. The normal component of velocity is usually specified directly via the parameter `VEL`. If `VEL` is negative, the flow is directed *into* the computational domain, i.e., a supply vent. If `VEL` is positive, the flow is drawn *out of* the domain, i.e., an exhaust vent. For example, the lines

```
&SURF ID='SUPPLY', VEL=-1.2, COLOR='BLUE' /  
&VENT XB=5.0,5.0,1.0,1.4,2.0,2.4, SURF_ID='SUPPLY' /
```

create a `VENT` that *supplies* air at a velocity of 1.2 m/s through an area of nominally 0.16 m², depending on the realignment of the `VENT` onto the FDS mesh. Regardless of the orientation of the plane $x = 5$, the flow will be directed *into* the room because of the sign of `VEL`. In this example the `VENT` may not be exactly 0.16 m² in area because it may not align exactly with the computational mesh. If this is the case then `VOLUME_FLOW` can be prescribed instead of `VEL`. The units are m³/s. If the flow is entering the computational domain, `VOLUME_FLOW` should be a negative number, the same convention as for `VEL`. Note that a `SURF` with a `VOLUME_FLOW` prescribed can be invoked by either a `VENT` or an `OBST`, but be aware that in the latter case, the resulting velocity on the face or faces of the obstruction will be given by the specified `VOLUME_FLOW` divided by the area of that particular face. For example:

```
&SURF ID='SUPPLY', VOLUME_FLOW=-5.0, COLOR='GREEN' /  
&OBST XB=..., SURF_ID6='BRICK','SUPPLY','BRICK','BRICK','BRICK','BRICK' /
```

dictates that the forward x -facing surface of the obstruction is to have a velocity equal to $5 \text{ m}^3/\text{s}$ divided by the area of the face (as approximated within FDS) flowing into the computational domain.

Note that `VEL` and `VOLUME_FLOW` should not be specified on the same `SURF` line. The choice depends on whether an exact velocity is desired at a given vent, or whether the given volume flow is desired.

Note also that if the `VENT` or `OBST` crosses mesh boundaries, the specified `VOLUME_FLOW` will be recomputed in each mesh so that the desired volume flow is achieved. This was not the case in FDS version 6.3 and earlier. The sample cases called `volume_flow_1.fds` and `volume_flow_2.fds` in the `Flowfields` folder demonstrate that a `VENT` or an `OBST` can be divided among several meshes. In both cases, air is drawn from a 1 m by 1 m by 1 m box at a rate of $0.01 \text{ m}^3/\text{s}$. These cases also ensure that the `VENT` or `OBST` need not be aligned with the mesh to yield the desired flow rate. Figure 10.1 displays the volume flow drawn through an `OPEN` boundary on an opposite face of the box with either a `VENT` (left) or `OBST` (right) with a specified volume flow.

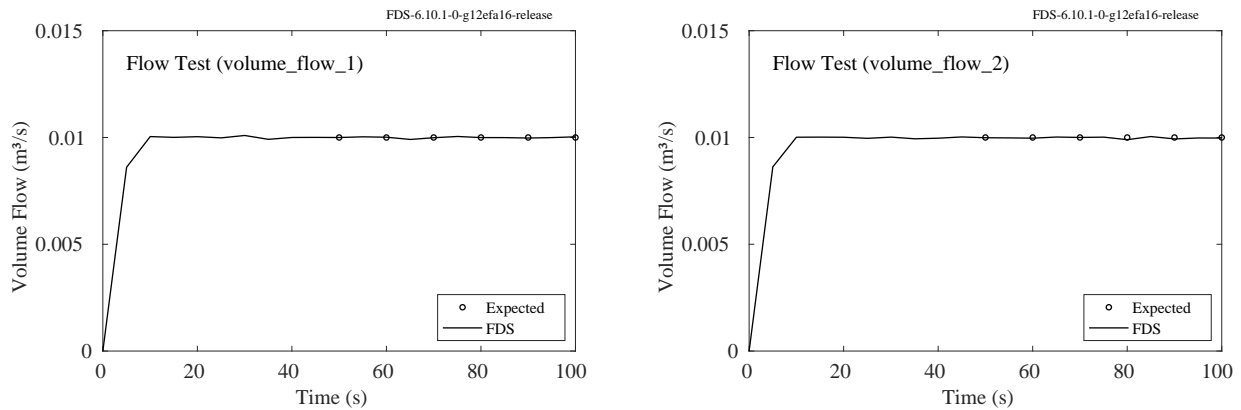


Figure 10.1: Flow rate of air drawn through a unit cube via a `VENT` (left) and `OBST` (right) with specified `VOLUME_FLOW`.

10.1.2 Total Mass Flux

Most often, a simple supply or exhaust vent is specified by setting either a normal velocity or volume flux at a solid surface. However, it is also possible to specify the total mass flow rate per unit area ($\text{kg}/(\text{m}^2 \text{s})$) via the parameter `MASS_FLUX_TOTAL`. This parameter uses the same sign convention as `VEL` above. In fact, the value entered for `MASS_FLUX_TOTAL` is converted internally into a velocity boundary condition whose value for an outflow is adjusted based on the local density. Note that `MASS_FLUX_TOTAL` should only be used for an outflow boundary condition; for inflow use `MASS_FLUX` which is discussed in Sec. 10.1.6.

10.1.3 Heaters

You can create a simple heating vent by changing the temperature of the incoming air

```
&SURF ID='BLOWER', VEL=-1.2, TMP_FRONT=50. /
```

The `VENT` with `SURF_ID='BLOWER'` would blow 50°C air at 1.2 m/s into the flow domain. Making `VEL` positive would suck air out, in which case `TMP_FRONT` would not be necessary.

Note that if `HRRPUA` or solid phase reaction parameters are specified, no velocity should be prescribed. The combustible gases are ejected at a velocity computed by FDS.

10.1.4 Louvered Vents

Most real supply vents are covered with some sort of grill or louvers which act to redirect, or *diffuse*, the incoming air stream. It is possible to mimic this effect, to some extent, by prescribing both a normal and the tangential components of the flow. The normal component is specified with `VEL` as described above. The tangential is prescribed via a pair of real numbers `VEL_T` representing the desired tangential velocity components in the other two coordinate directions (x or y should precede y or z). For example, the line in the example case `Flowfields/tangential_velocity.fds`

```
&SURF ID='LOUVER', VEL=-2.0, VEL_T=3.0,0.0, TAU_V=5., COLOR='GREEN' /
```

is a boundary condition for a louvered vent that pushes air into the space with a normal velocity of 2 m/s and a tangential velocity of 3 m/s in the first of the two tangential directions. Note that the negative sign of the normal component of velocity indicates that the fluid is injected into the computational domain. The tangential velocity of 3 m/s indicates that the flow is in the positive y direction. Both the normal and tangential velocity components are ramped up with either `TAU_V` or `RAMP_V`, as shown in Fig. 10.2.

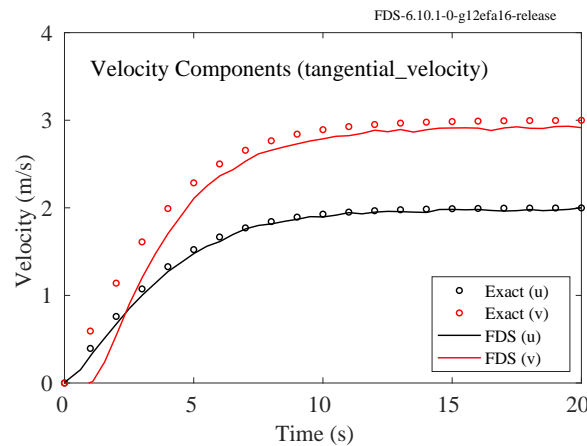


Figure 10.2: Normal and tangential velocity components at a louvered vent, compared to the ideal curve.

In cases of limited mesh resolution, there may not be enough mesh cells spanning the opening for `VEL_T` to operate correctly. In these cases, you might consider simply specifying a flat plate obstruction in front of the `VENT` with an offset of one mesh cell. The plate will simply redirect the air flow in all lateral directions.

If the louvered vent is part of an HVAC system, see 10.2.7 for details on how to specify the louver.

10.1.5 Specified Normal Velocity Gradient

It is sometimes desirable to specify a Neumann boundary condition (specified gradient) for the velocity in the direction normal to the boundary. For example, the following allows inflow and outflow along the top of the domain, but $\partial w / \partial z = 0$. Note that `FREE_SLIP=T` only sets $\partial u / \partial z = 0$ and $\partial v / \partial z = 0$.

```
&SURF ID = 'sky', COLOR = 'INVISIBLE', VEL_GRAD=0., FREE_SLIP=T /  
&VENT MB='ZMAX', SURF_ID='sky' /
```

10.1.6 Species and Species Mass Flux Boundary Conditions

There are two types of species boundary conditions (see Chapter 12 for a general discussion of gas species). By default, gas species do not penetrate solid surfaces and you need not specify anything if that is all that is needed. If the mass fraction of the species is to be some value at a forced flow boundary where `VEL`, `VOLUME_FLOW`, or `MASS_FLUX_TOTAL` is specified, set `MASS_FRACTION(:)` equal to the desired species mass fractions on the appropriate `SURF` line. If the mass flux of the species is desired, set `MASS_FLUX(:)` instead of `MASS_FRACTION(:)`. If `MASS_FLUX(:)` is set, do not specify `VEL`, `VOLUME_FLOW`, or `MASS_FLUX_TOTAL`. These are automatically calculated based on the specified mass flux. The inputs `MASS_FLUX(:)` and typically `MASS_FRACTION(:)` should only be used for inflow boundary conditions. `MASS_FLUX(:)` should be positive with units of $\text{kg}/(\text{m}^2\text{s})$. Also note that the background species cannot be specified when using `MASS_FRACTION`. The mass fraction of the background species will be set to account for any mass fraction not specified with other species.

Here is an example of how to specify a surface that generates methane at a rate of $0.025 \text{ kg}/(\text{m}^2\text{s})$:

```
&SPEC ID='METHANE' /
&SURF ID='METHANE BURNER', SPEC_ID(1)='METHANE', MASS_FLUX(1)=0.025 /
&VENT XB=..., SURF_ID='METHANE BURNER' /
```

Here is example of how to specify a surface that blows methane with a velocity of 0.1 m/s :

```
&SPEC ID='METHANE' /
&SURF ID='METHANE BLOWER', MASS_FRACTION(1)=1.0, SPEC_ID(1)='METHANE', VEL=-0.1 /
&VENT XB=..., SURF_ID='METHANE BLOWER' /
```

Note that specifying a combination of `VEL` and `MASS_FRACTION` can lead to inaccurate results if the specified velocity is small because diffusion will dominate the mass transport. To obtain an accurate species mass flux at a boundary, use `MASS_FLUX`.

Alternatively, add `CONVERT_VOLUME_TO_MASS=T` for velocity boundaries (`VEL`, `VOLUME_FLOW`, or `MASS_FLUX_TOTAL`), which converts volume flow to a mass flux based on the specified boundary composition (`MASS_FRACTION`) and temperature (`TMP_FRONT`):

$$\dot{m}''_{\alpha} = \rho Z_{\alpha} u = \frac{p_{\infty} \bar{W}}{RT} Z_{\alpha} \frac{\dot{Q}}{A} \quad (10.1)$$

where ρ is the density, Z_{α} is the mass fraction of α , u is the velocity normal to the surface, p_{∞} is the ambient pressure, \bar{W} is the mixture molecular weight, R is the ideal gas constant, T is the surface temperature, \dot{Q} is the volume flow rate, and A is the area of the vent. Note that using `CONVERT_VOLUME_TO_MASS=T` converts the `SURF` into a mass flux boundary condition and so velocity profiles may not be applied.

10.1.7 Tangential Velocity Boundary Conditions at Solid Surfaces

The no-slip condition implies that the continuum tangential gas velocity at a surface is zero. In turbulent flow the velocity increases rapidly through a boundary layer that is only a few millimeters thick to its “free-stream” value. In most practical simulations, it is not possible to resolve this boundary layer directly; thus, an empirical model is used to represent its effect on the overall flow field. For a DNS (Direct Numerical Simulation), the velocity gradient at the wall is computed directly from the resolved velocity near the wall (`NO_SLIP=T` by default). For an LES (Large Eddy Simulation), a “log law” wall model is applied. The “sand grain” surface roughness¹ (in meters) is set by `ROUGHNESS` on `SURF`. See the FDS Technical Reference

¹Note the *sand grain* roughness, s , is different than the *aerodynamic* roughness, z_0 , used in atmospheric flows (see Sec. 16.3.4).

Guide [3] for wall model details. To force a solid boundary to have a free-slip condition, set `FREE_SLIP=T` on the `SURF` line. In LES, to override the wall model and force a no-slip boundary condition, set `NO_SLIP=T` on the `SURF` line.

10.1.8 Synthetic Turbulence Inflow Boundary Conditions

Real flows of low-viscosity fluids like air are rarely perfectly stationary in time or uniform in space—they are turbulent (to some degree). Of course, the turbulence characteristics of the flow may have a significant impact on mixing and other behaviors, so the specification of nominally constant and uniform boundary conditions may be insufficient. To address this issue, FDS employs a synthetic eddy method (SEM)². Refer to Jarrin [12] for a detailed description. In brief, “eddies” are injected into the flow at random positions on the boundary and advect with the mean flow over a short distance near the boundary equivalent to the maximum eddy length scale. Once the eddy passes through this region it is recycled at the inlet of the boundary with a new random position and length scale. The eddies are idealized as velocity perturbations over a spherical region in space with a diameter (eddy length scale) selected from a uniform random distribution. The selection procedures guarantee that prescribed first and second-order statistics (including Reynolds stresses) are satisfied.

Synthetic turbulence is invoked by setting the number of eddies, `N_EDDY`, the characteristic eddy length scale, `L_EDDY`, and either the root mean square (RMS) velocity fluctuation, `VEL_RMS`, or the Reynolds stress tensor components, `REYNOLDS_STRESS(3,3)` on the `VENT` line. In Fig. 10.3 we show examples using SEM for flat, parabolic, atmospheric, and ramp profiles with 10 % turbulence intensity (see the `sem_*` test series in the Turbulence verification subdirectory). The input lines for the atmospheric case are (see Section 16.5 for further discussion of profile parameters).

```
&SURF ID='inlet', VEL=-1, PROFILE='ATMOSPHERIC', Z0=0.5, PLE=0.3 /
&VENT MB='XMIN', SURF_ID='inlet', N_EDDY=100, L_EDDY=0.2, VEL_RMS=.1 /
```

Note that the Reynolds stress is symmetric and only the lower triangular part needs to be specified. The RMS velocity fluctuation is isotropic (equivalent for each component). Thus, $VEL_RMS \equiv \sqrt{2k/3}$, where $k \equiv \langle \frac{1}{2} u'_i u'_i \rangle$ is the turbulent kinetic energy per unit mass. Below is an example illustrating the equivalence between the RMS velocity fluctuation and the diagonal components of the Reynolds stress. Note that if `VEL_RMS` is specified, this is equivalent to

```
REYNOLDS_STRESS(1,1) = VEL_RMS**2
REYNOLDS_STRESS(2,2) = VEL_RMS**2
REYNOLDS_STRESS(3,3) = VEL_RMS**2
```

and all other components of `REYNOLDS_STRESS` are zero. If the fluctuations are not isotropic, then the Reynolds stresses must be specified component-wise.

In Chapter 7 of Jarrin’s thesis [12], he introduces the Modified Synthetic Eddy Method in which the eddy length scales are anisotropic. This allows more realistic characterization of stream-wise vortices in a turbulent boundary layer. To specify the length scales corresponding to the σ_{ij} values in Jarrin’s Eq. (7.1) use `L_EDDY_IJ(3,3)`. Here is an example with random values for the eddy length scales and Reynolds stress components:

```
&VENT XB=... , SURF_ID='WIND', N_EDDY=500,
  L_EDDY_IJ(1,1)=21., L_EDDY_IJ(1,2)=6.22, L_EDDY_IJ(1,3)=4.23
  L_EDDY_IJ(2,1)=2.35, L_EDDY_IJ(2,2)=5.66, L_EDDY_IJ(2,3)=2.50
```

²SEM may not be used for HVAC vents, which are strict mass flux boundaries.

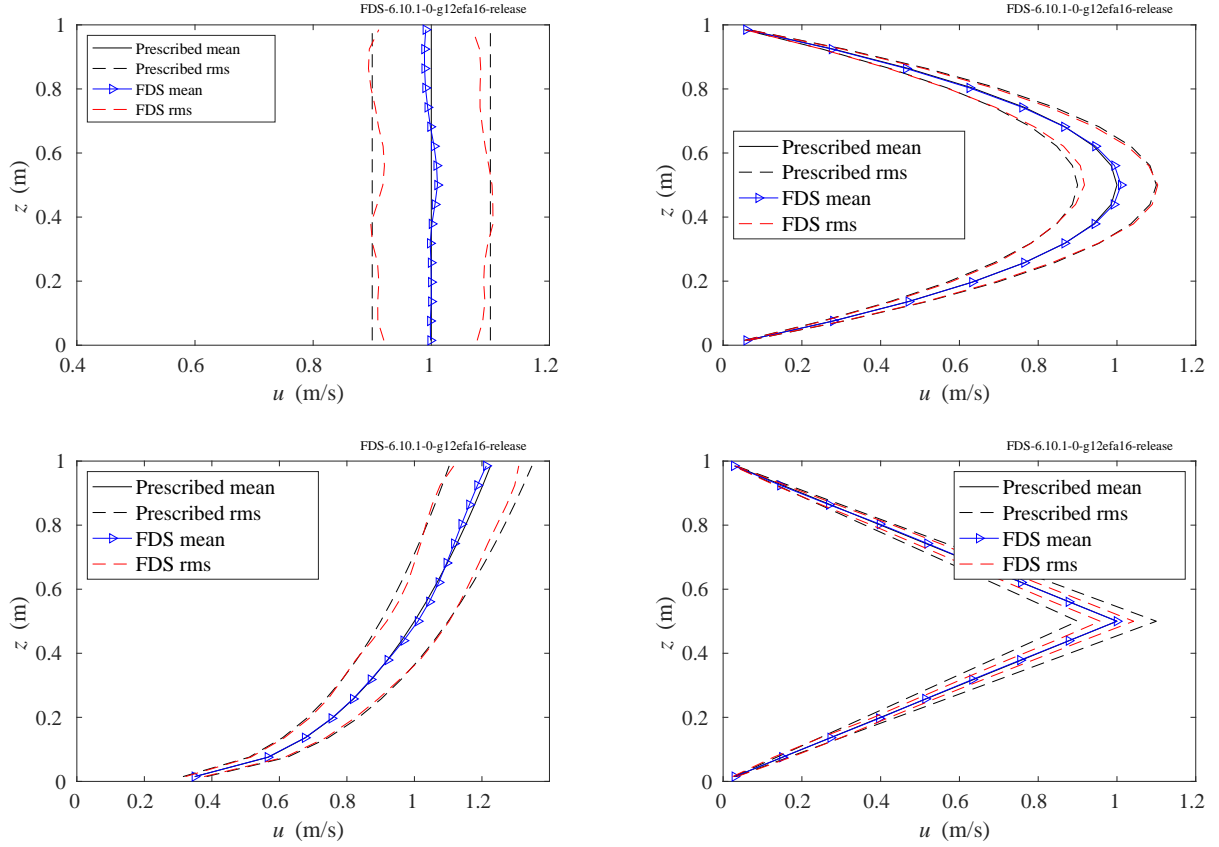


Figure 10.3: Synthetic Eddy Method vent profiles: flat (upper left), parabolic (upper right), atmospheric (lower left), and linear ramp (lower right).

```
L_EDDY_IJ(3,1)=5.42, L_EDDY_IJ(3,2)=0.78, L_EDDY_IJ(3,3)=1.01
REYNOLDS_STRESS(1,1)=2.16, REYNOLDS_STRESS(1,2)=0., REYNOLDS_STRESS(1,3)=-0.47
REYNOLDS_STRESS(2,1)=0., REYNOLDS_STRESS(2,2)=1.53, REYNOLDS_STRESS(2,3)=0.
REYNOLDS_STRESS(3,1)=-0.47, REYNOLDS_STRESS(3,2)=0., REYNOLDS_STRESS(3,3)=4.259 /
```

Synthetic Turbulence at OPEN Boundaries

For wind simulations (see Sec. 16.2) it may be necessary to add a degree of inflow turbulence in order to match atmospheric conditions without drastically increasing the size of the computational domain. The Synthetic Eddy Method of Jarrin [12] may be invoked on the VENT line with an 'OPEN' boundary. This should be used in conjunction with a WIND line to specify the mean wind field. For example,

```
&WIND SPEED=10, DIRECTION=270, ... /
&VENT DB='XMIN', SURF_ID='OPEN', N_EDDY=500, L_EDDY=3, VEL_RMS=1 /
```

Figure 10.4 shows results from a demonstration case (sem_open_wind.fds). In this case the mean wind speed is 10 m/s at 10 m elevation with a Monin-Obukhov length of -667 m (unstably stratified) and an aerodynamic roughness of $z_0 = 0.022$ m. The ground temperature is set to 20°C and the ambient temperature is set to 19.18°C based on the Monin-Obukhov profile. The rms velocity fluctuation at the inlet is set to 1 m/s. The lateral boundaries are 'PERIODIC'. The inflow, outflow, and top boundary conditions

are set to 'OPEN'. Note that a special OPEN boundary condition for WIND has been developed, which is discussed in the FDS Tech Guide [3].

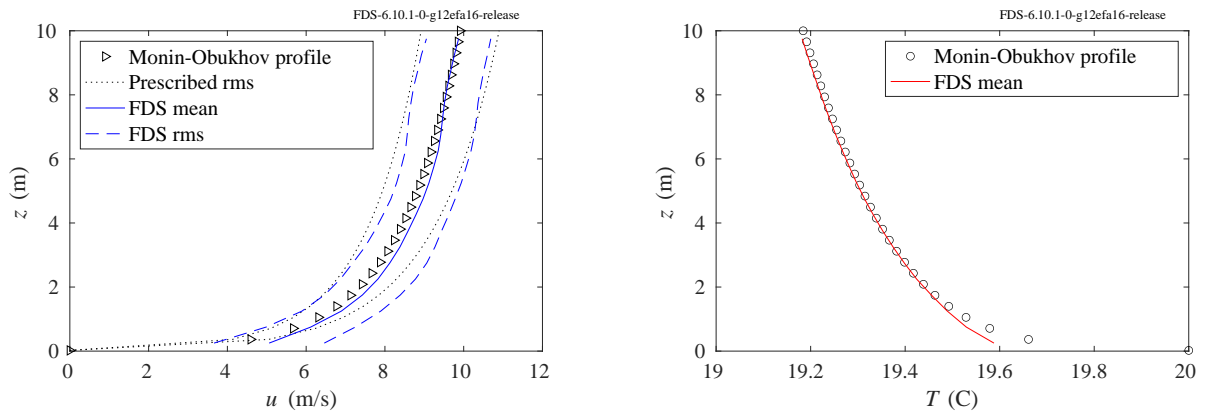


Figure 10.4: Synthetic Eddy Method at OPEN boundary with prescribed WIND field: (left) Monin-Obukhov velocity profile, (right) Monin-Obukhov temperature profile.

10.1.9 Random Mass Flux Variation

The current implementation of the Synthetic Eddy Method does not allow variation in mass flux defined on a SURF line. For example, the parameters HRRPUA and MASS_FLUX follow a user-specified time history which does not include random fluctuations. However, you may specify a mass flux variation using the parameter MASS_FLUX_VAR. For example, if you want a 10 % variation in a heat release rate per unit area of 100 kW/m² then use

```
&SURF ID='burner', HRRPUA=100., MASS_FLUX_VAR=0.1 /
```

Using the boundary file output quantity 'MASS FLUX' might help to visualize the variation. For example,

```
&BNDF QUANTITY='MASS FLUX', CELL_CENTERED=T /
```

10.2 HVAC Systems

There are occasions where simply defining fixed flow and fixed species boundary conditions is not sufficient to model the behavior of an HVAC (Heating, Ventilation, and Air Conditioning) system. If the ability to transport heat and combustion products through a duct network or the ability to fully account for the pressurization of a compartment due to a fire on the flows in a duct network is important, you can make use of a coupled HVAC network solver. The solver computes the flows through a duct network described as a mapping of duct segments and nodes where a node is either the joining of two or more ducts (a tee for example) or where a duct segment connects to the FDS computational domain. For more information on coupled hybrid modeling in fire safety engineering, interested readers may want to refer to Ralph et al.'s literature review of the topic [13].

By default the HVAC solver does not allow for mass storage in the duct network (i.e., what goes in during a time step, goes out during a time step). However, if you set `N_CELLS` for an HVAC duct, then the HVAC solver will account for mass storage and will compute transient species and energy transport through the ducts.

HVAC components such as fans and binary dampers (fully open or fully closed) can be included in the HVAC network and are coupled to the FDS control function capability. Three fan models are available for use.

The HVAC solver is invoked if there is an `HVAC` namelist group present in the input file. An HVAC network is defined by providing inputs for the ducts; duct nodes; and any fans, dampers, filters, or heating and coiling coils present in the system. Additionally you must define the locations where the HVAC network joins the computational domain. The basic syntax for an HVAC component is:

```
&HVAC TYPE_ID='componenttype', ID='componentname', ... /
```

where:

- `TYPE_ID` is a character string that indicates the type of component that the namelist group is defining. `TYPE_ID` can be `DUCT`, `NODE`, `FAN`, `FILTER`, `AIRCOIL`, or `LEAK` (see Sec. 10.3.2).
- `ID` is a character string giving a name to the component. The name must be unique amongst all other components of that type; however, the same name can be given to components of different types (i.e., a duct and a node can have the same name but two ducts cannot).

A number of examples of simple HVAC systems are given in the HVAC folder of the sample cases and are discussed in the FDS Verification Guide.

As described in the Technical Reference Manual, the HVAC pressure solution is not directly coupled to the FDS pressure solution. Rather there is implicit coupling using the wall boundary condition for HVAC vents. At times this can result in stability problems. There are two methods of attempting to resolve this.

The first method is the keyword `HVAC_PRES_RELAX` on the `MISC` line with a default value of 1.0. At each time step FDS evaluates the average pressure in the FDS gas cells adjacent to nodes connecting the FDS domain to the HVAC network, P_{FDS} . The node pressure at time step $n+1$, P_{node}^{n+1} , is taken as:

$$P_{node}^{n+1} = P_{node}^n (1 - \text{HVAC_PRES_RELAX}) + P_{FDS}^{n+1} \text{HVAC_PRES_RELAX} \quad (10.2)$$

Setting this parameter closer to 0 reduces the sensitivity of the HVAC solution to short, transient pressure changes in the FDS domain; however, doing so will also result in the HVAC solution lagging for longer duration pressure changes.

The second method is setting the keyword `HVAC_LOCAL_PRESSURE` on the `MISC` line. When set, this will cause FDS to use the `ZONE` pressure at a vent plus the stagnation pressure of any flow normal to the vent to

determine the pressure boundary condition for a node connected to the FDS domain rather than the `ZONE` pressure plus the the pressure derived from the local value of H .

10.2.1 HVAC Duct Parameters

A typical input line specifying a duct is as follows:

```
&HVAC TYPE_ID='DUCT', ID='ductname', NODE_ID='node 1','node 2', AREA=3.14,  
      LOSS=1.,1., LENGTH=2., ROUGHNESS=0.001, FAN_ID='fan 1', DEVC_ID='device 1' /
```

Or, if you are using HVAC mass transport, as follows:

```
&HVAC TYPE_ID='DUCT', ID='ductname', NODE_ID='node 1','node 2', AREA=3.14,  
      LOSS=1.,1., LENGTH=2., ROUGHNESS=0.001, FAN_ID='fan 1', DEVC_ID='device 1',  
      N_CELLS=200 /
```

where:

- `AIRCOIL_ID` is the ID of an aircoil located in the duct. The operation of the aircoil can be controlled by either a device or a control function.
- `AREA` is the cross sectional area of the duct in m^2 .
- `DAMPER` is a logical parameter indicating the presence of a damper in the duct. The state of the damper is controlled by either a device or a control function (see Sec. 10.2.2).
- `DIAMETER` is the diameter of the duct in m.
- `DEVC_ID` is the ID of a `DEVC` for a damper, fan, or aircoil in the duct. An alternative is `CTRL_ID`.
- `FAN_ID` is the ID of a fan located in the duct. Instead of specifying a `FAN_ID`, you could specify the `VOLUME_FLOW` rate (m^3/s) through the duct. The operation of the fan can be controlled by either a device or a control function.
- `LENGTH` is the `LENGTH` of the duct in m. If this is not specified, it will be computed automatically as the difference between the `XYZ` of the duct's endpoints or as the distance along the path between endpoints if `WAYPOINTS` are defined.
- `LOSS` is a pair of real numbers giving the forward and reverse dimensionless "minor losses" loss coefficient (K_{minor}) in the duct. Minor losses are pressure losses through components such as tees, valves and bends. However, you can use `LOSS` to represent wall friction losses if you want - in this case ensure you leave `ROUGHNESS` unset so that the HVAC solver does not compute a friction factor. The forward direction is defined as flow from the first node listed in `NODE_ID` to the second node listed in `NODE_ID`.
- `MASS_FLOW` is a fixed mass flow rate (kg/s) through the duct. Do not specify both `MASS_FLOW` and `VOLUME_FLOW` for the same duct. Its value in time may be changed by either using the characteristic time, `TAU_VF`, to define a tanh (`TAU_VF`>0) or t^2 ramp (`TAU_VF`<0); or `RAMP_ID` to specify the flow over time. `MASS_FLOW` should only be specified for conditions where the upstream node density will not change during the solution process.
- `N_CELLS` Enables HVAC mass transport. It defines the number of cells used in a discretized duct.

- **NETWORK_ID** Used for Smokeview visualization. All ducts and nodes with a common **NETWORK_ID** can be selected or deselected as a group in Smokeview. If no value is given, the duct will be placed in the Unassigned network.
- **NODE_ID** gives the IDs of the nodes on either end of the duct segment. Positive velocity in a duct is defined as flow from the first node to second node.
- **PERIMETER** is used along with **AREA** to specify a duct with non-circular cross-section. The **DIAMETER** will be computed as the hydraulic diameter.
- **RAMP_LOSS** If specified this **RAMP** is a multiplier for the **LOSS**.
- **REVERSE** is a logical parameter that when **T** indicates that the specified **FAN_ID** blows from the second node to the first. **REVERSE** has no effect on **VOLUME_FLOW** or **MASS_FLOW** as a duct input. If **VOLUME_FLOW** or **MASS_FLOW** is specified for a duct and the reverse flow direction is needed, change the sign of the input to negative.
- **ROUGHNESS** is the absolute roughness in m of the duct that is used to compute the friction factor for the duct. If **ROUGHNESS** is not set, the HVAC solver will not compute the friction factor and the wall friction will be zero - if this is the case you may want to account for wall friction losses in **LOSS**. "Perfectly smooth" ducts and pipes still have wall losses and therefore setting **ROUGHNESS** to zero will tell the HVAC solver to compute the minimum friction factor (which is non-zero) - this is not the same as leaving **ROUGHNESS** unset.
- **ROUND** flag indicating a round duct.
- **SQUARE** flag indicating a square duct.
- **VOLUME_FLOW** is a fixed flow rate (m^3/s) through the duct. Only specify one of **MASS_FLOW** or **VOLUME_FLOW**. If you specify **VOLUME_FLOW**, its value in time either using the characteristic time, **TAU_VF**, to define a tanh (**TAU_VF**>0) or t^2 ramp (**TAU_VF**<0); or **RAMP_ID** to specify the flow over time. **VOLUME_FLOW** cannot be controlled by a device or control function. This can, however, be done using a constant volume flow **FAN**.
- **WAYPOINTS** is an array of coordinates in m that define the path a duct takes. Currently this only used for Smokeview visualization of the HVAC network. Up to 30 **WAYPOINTS** can be defined. For example: **WAYPOINTS (1,1:3)=1,1,1, WAYPOINTS (2,1:3)=2,2,2** as part of a duct input would mean the duct starts at the first node, then goes to the point (1,1,1), then to (2,2,2), and finally to the second node.

Note that only one of **AIRCOIL_ID**, **DAMPER**, or **FAN_ID** should be specified for a duct. Also note that if one of these is specified, but no device or control function is provided, then the item will be assumed to be on or open as appropriate.

If **ROUND** or **SQUARE** are set, then only one of **AREA**, **DIAMETER**, or **PERIMETER** should be set. Otherwise two of the three are required.

To reduce the computational cost of the HVAC solver, a duct should be considered as any length of duct that connects two items that must be defined as nodes (i.e., a connection to the FDS domain, a filter, or a location where more than two ducts join). That is, a duct should be considered as any portion of the HVAC system where flow can only be in one direction at given point in time (flow can reverse direction over time). For example the top of Figure 10.5 shows a segment of an HVAC system where flow from a tee goes through an expansion fitting, two elbows, an expansion fitting, and a straight length of duct before it terminates as a connection to the FDS domain.

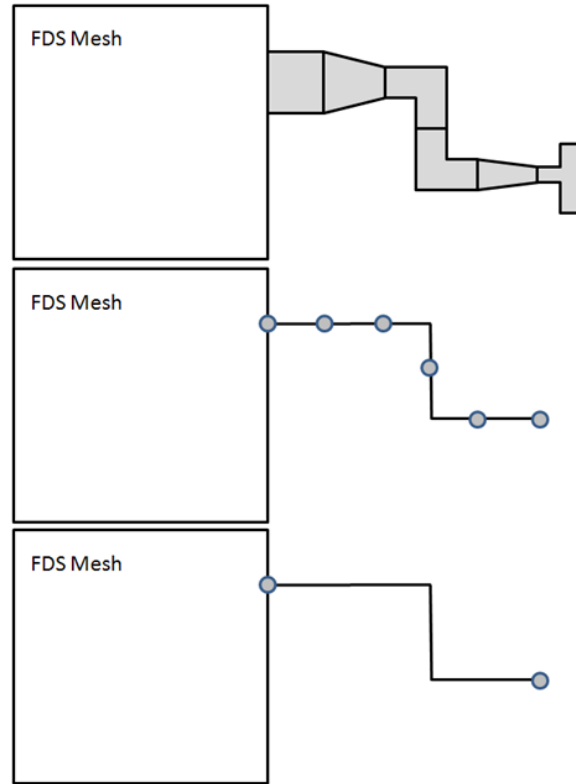


Figure 10.5: An example of simplifying a complex duct.

This could be input as each individual fitting or duct with its associated area and loss as shown in the middle of the figure; however, this would result in five duct segments (one for each component) with six node connections resulting in eleven parameters (five velocities and six pressures) which must be solved for. This is not needed since whatever the flow rate is in any one segment of the duct, that same flow rate exists in all other segments; thus, the velocities in any segment can be found by taking the area ratios, $v_1/v_2 = A_2/A_1$. Since flow losses are proportional to the square of the velocity, an equivalent duct can be constructed using the total length of the duct, and a representative area (A_{eff}) or diameter. The pressure losses associated with all the segments of the duct can be collapsed to a single effective loss (K_{eff}) by summing all of the fitting losses (K_{minor}) through the duct as follows:

$$K_{\text{eff}} = \sum_i (K_{\text{minor}})_i \left(\frac{A_{\text{eff}}}{A_i} \right)^2 \quad (10.3)$$

where i is a fitting and A_i is the area associated with the fitting loss.

10.2.2 HVAC Dampers

Dampers can be modeled in one of two ways. The first method is a simple binary (flow or no flow) damper. This can be placed in a duct by adding the keyword `DAMPER` along with either a `CTRL_ID` or `DEVC_ID`. When the control or device is `T` the damper will be open, and when `F` the damper will be closed and block 100 % of the duct area. The example below shows a duct with a damper that is linked to a `DEVC` that closes the damper at 10 s.

```
&HVAC TYPE_ID='DUCT', ID='EXHAUST 2', NODE_ID='TEE', 'EXHAUST 2', AREA=0.01,
      LENGTH=1.0, LOSS=0, 0, DAMPER=T, DEVC_ID='TIMER' /
&DEVC QUANTITY='TIME', ID='TIMER', SETPOINT=10, INITIAL_STATE=T, XYZ=0, 0, 0 /
```

To start with the damper in the closed position and then open at a later time, the `INITIAL_STATE` of the `DEVC` should not be set to `T`. The following example shows a duct with a damper which starts closed and then opens at 10 s. For further details see the `HVAC_damper` example case, which is documented in the Verification Guide.

```
&HVAC TYPE_ID='DUCT', ID='EXHAUST 2', NODE_ID='TEE', 'EXHAUST 2', AREA=0.01,
      LENGTH=1.0, LOSS=0, 0, DAMPER=T, DEVC_ID='TIMER' /
&DEVC QUANTITY='TIME', ID='TIMER', SETPOINT=10, XYZ=0, 0, 0 /
```

The second method is to specify a `RAMP_LOSS` for the duct. This approach multiplies the `LOSS` array for the duct by the output of the `RAMP`. This allows for dampers that have leakage and/or dampers that have a variable position other than fully open or fully closed. An example is shown below where the `LOSS` in the duct changes from 1,1 at 10 s to 2000,2000 at 11 s.

```
&HVAC TYPE_ID='DUCT', ID='EXHAUST 2', NODE_ID='TEE', 'EXHAUST 2', AREA=0.01,
      LENGTH=1.0, LOSS=1, 1, RAMP_LOSS='LOSS RAMP' /
&RAMP ID='LOSS RAMP', T=10, F=1 /
&RAMP ID='LOSS RAMP', T=11, F=2000 /
```

10.2.3 HVAC Node Parameters

Below are three example duct node inputs representing a typical tee-type connection (multiple ducts being joined), a connection to the FDS domain, and a connection to the ambient outside the FDS domain.

```
&HVAC TYPE_ID='NODE', ID='tee', DUCT_ID='duct 1', 'duct 2', .. 'duct n',
      LOSS=lossarray, XYZ=x, y, z /
&HVAC TYPE_ID='NODE', ID='FDS connection', DUCT_ID='duct 1', VENT_ID='vent',
      LOSS=enter, exit /
&HVAC TYPE_ID='NODE', ID='ambient', DUCT_ID='duct 1', LOSS=enter, exit,
      XYZ=x, y, z, AMBIENT=T /
```

where:

- `AMBIENT` is a logical value. If `T`, then the node is connected to the ambient (i.e., it is equivalent to the `OPEN` boundary condition on a `SURF` line).
- `DUCT_ID` gives the IDs of the ducts connected to the node. Only one duct is allowed if the node is `AMBIENT` or has a `VENT_ID`. Up to 10 ducts can be connected to a node.
- `FILTER_ID` gives the ID a filter located at the node. A node with a filter must have two connected ducts. This means a filter cannot be located at an ambient node, a node that is attached to a `VENT`, or node with three or more ducts.
- `LOSS` is an n by n array of real numbers giving the dimensionless loss coefficients for the node. `LOSS(I, J)` is the loss coefficient for flow from duct `I` to duct `J` expressed in terms of the downstream duct area (see discussion in 10.2.1 on how to adjust losses for area changes). For a terminal node (e.g., a node connected to the ambient or to a `VENT`) the `LOSS` is entered as a pair of numbers representing loss coefficient for flow entering the HVAC system and for flow exiting the HVAC system.

- `NETWORK_ID` Used for Smokeview visualization. All ducts and nodes with a common `NETWORK_ID` can be selected or deselected as a group in Smokeview. If no value is given, the node will be placed in the Unassigned network.
- `VENT_ID` is the name of the `VENT` where the node connects to the FDS computational domain. No two `VENTs` should be defined with the same `VENT_ID`.
- `XYZ` is a triplet of real numbers giving the coordinates of the node. If the node is connected to the FDS domain, then do not specify `XYZ` as FDS will compute it as the centroid of the `VENT`.

A duct node must either have two or more ducts attached to it or it must have either `AMBIENT=T` or a specified `VENT_ID`. When defining a `VENT` as a component of an HVAC system you must set `SURF_ID` to 'HVAC' and you must set the `VENT_ID`.

It is permissible to have individual `VENT` lines for an HVAC system span multiple meshes; however, the entire `VENT` must lie within a single pressure zone.

Note that a `VENT` being used for an HVAC system must be present throughout the simulation. The `VENT` should not have a `CTRL_ID` or `DEVC_ID`. This also includes the solid surface it is attached; i.e., any `OBST` the `VENT` is attached to also needs to be present throughout the simulation. Turning on or off a `VENT` connected to an HVAC system, requires a damper in the duct connected to `VENT`.

10.2.4 HVAC Fan Parameters

Below are sample inputs for the three types of fans supported by FDS.

```
&HVAC TYPE_ID='FAN', ID='constant volume', VOLUME_FLOW=1.0, LOSS=2./
&HVAC TYPE_ID='FAN', ID='quadratic', MAX_FLOW=1., MAX_PRESSURE=1000., LOSS=2. /
&HVAC TYPE_ID='FAN', ID='user fan curve', RAMP_ID='fan curve', LOSS=2. /
```

where:

- `LOSS` is the loss coefficient for flow through the fan when it is not operational. FDS assumes a default value of 1.
- `MAX_FLOW` is the maximum volumetric flow of the fan in m^3/s . This input activates a quadratic fan model. Value must be > 0 .
- `MAX_PRESSURE` is the stall pressure of the fan in units of Pa. This input activates a quadratic fan model. Value must be > 0 .
- `RAMP_ID` identifies the `RAMP` that contains a table of pressure drop across the fan (Pa) versus the volumetric flow rates (m^3/s) for a user-defined fan curve.
- `TAU_FAN` defines a tanh ($\text{TAU_FAN} > 0$) or t^2 ramp ($\text{TAU_FAN} < 0$) for the fan. This is applied to the flow rate computed by any of the three types (constant flow, quadratic, or user-defined ramp) of fans.
- `VOLUME_FLOW` is the fixed volumetric flow of the fan (m^3/s). If you wish to have a time dependent flow use the `VOLUME_FLOW` input for a duct rather than for a fan.

Note that only one set of fan model inputs (`VOLUME_FLOW`, `RAMP_ID`, or `MAX_FLOW` + `MAX_PRESSURE`) should be specified. Also note that `FAN` defines a class of fans rather than one specific fan. Therefore, more than one duct can reference a single `FAN`.

Fan Curves

In Sec. 10.1 there is a discussion of velocity boundary conditions, in which a fan is modeled simply as a solid boundary that blows or sucks air, regardless of the surrounding pressure field. In the HVAC model, this approach to modeling a fan occurs when the fan is specified with a `VOLUME_FLOW`. In reality, fans operate based on the pressure drop across the duct or manifold in which they are installed. A very simple “fan curve” is given by:

$$\dot{V}_{\text{fan}} = \dot{V}_{\text{max}} \text{sign}(\Delta p_{\text{max}} - \Delta p) \sqrt{\frac{|\Delta p - \Delta p_{\text{max}}|}{\Delta p_{\text{max}}}} \quad (10.4)$$

This simple “fan curve” is the “quadratic” fan model as the pressure is proportional to the square of the volume flow rate.

The volume flow in the absence of a pressure difference, `MAX_FLOW`, is given by \dot{V}_{max} . The pressure difference, $\Delta p = p_1 - p_2$, indicates the difference in pressure between the downstream compartment, or “zone,” and the upstream. The subscript 1 indicates downstream and 2 indicates upstream. The term, Δp_{max} , is the maximum pressure difference, `MAX_PRESSURE`, the fan can operate upon, and it is assumed to be a positive number. The flow through a fan will decrease from \dot{V}_{max} at zero pressure difference to 0 m³/s at Δp_{max} . If the pressure difference increases beyond this, air will be forced backward through the fan. If the downstream pressure becomes negative, then the volume flow through the fan will increase beyond `MAX_FLOW`. More complicated fan curves can be specified by defining a `RAMP`. In the example inputs below, one fan of each type is specified. A constant volume flow fan with a `VOLUME_FLOW` of 10 m³/s, a quadratic fan with `MAX_FLOW`=10 and `MAX_PRESSURE`=500, and a user-defined fan with the `RAMP` set to the values of the quadratic fan in 200 Pa increments,

```
&HVAC TYPE_ID='FAN', ID='constant volume', VOLUME_FLOW=10.0/
&HVAC TYPE_ID='FAN', ID='quadratic', MAX_FLOW=10., MAX_PRESSURE=500./
&HVAC TYPE_ID='FAN', ID='user fan curve', RAMP_ID='fan curve'/
&RAMP ID='fan curve', T=-10.00, F= 1000/
&RAMP ID='fan curve', T= -7.75, F=  800/
&RAMP ID='fan curve', T= -4.47, F=  600/
&RAMP ID='fan curve', T=  4.47, F=  400/
&RAMP ID='fan curve', T=  7.75, F=  200/
&RAMP ID='fan curve', T= 10.00, F=   0/
&RAMP ID='fan curve', T= 11.83, F= -200/
&RAMP ID='fan curve', T= 13.42, F= -400/
&RAMP ID='fan curve', T= 14.83, F= -600/
&RAMP ID='fan curve', T= 16.12, F= -800/
&RAMP ID='fan curve', T= 17.32, F=-1000/
```

Figure 10.6 displays the fan curves for the inputs shown above. Additional examples can be found in the `ashrae7` and `fan_test` example cases, which are documented in the Verification Guide.

Jet Fans

Fans do not have to be mounted on a solid wall, like a supply or an exhaust fan. If you just want to blow gases in a particular direction, create an obstruction `OBST`, at least one cell thick, and apply to it `VENT` lines that are associated with a simple HVAC system. This allows hot, smokey gases to pass through the obstruction, much like a free-standing fan. See the example case `jet_fan.fds` which places a louvered fan (blowing diagonally down) near a fire (see Fig. 10.7).

You may also want to construct a *shroud* around the fan using four flat plates arranged to form a short passageway that draws gases in one side and expels them out the other. The obstruction representing the fan

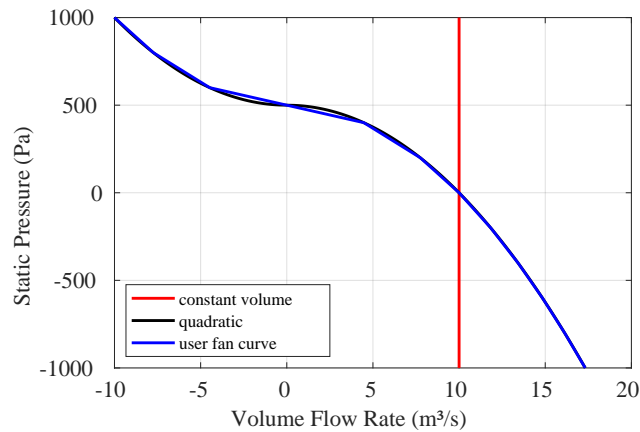


Figure 10.6: Fan curves corresponding to a constant fan with `VOLUME_FLOW=10`, a quadratic fan with `MAX_FLOW=10` and `MAX_PRESSURE=500`, and a user-defined RAMP equivalent to the quadratic fan.

can be positioned about halfway along the passage (if a louvered fan is being used, place the fan at the end of the passage).

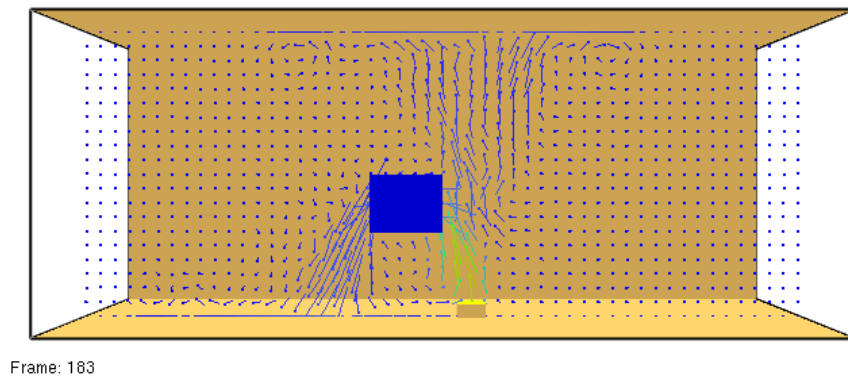


Figure 10.7: Jet fan with a louvered output `UVW=-1, 0, -1`.

System Curves

An HVAC system can be characterized with two curves. One, already discussed, is the fan curve which describes the flow through the fan as a function of the pressure head across the fan. This is a curve with zero flow at maximum pressure and maximum flow at zero pressure. The second is the system curve. This describes how the flow in an HVAC system changes as the pressure imposed by a fan changes. The system curve accounts for all the losses in the system plus any pressure differences seen at the inlets and outlets to the system. For an HVAC system whose inlets and outlets are all at the same pressure, the system curve will have zero flow at zero pressure and increasing flow as the fan pressure increases. The operation point for a specific combination of an HVAC system and a fan is given by the intersection of these two curves.

For relatively simple HVAC networks with only a single fan for in any given duct run and no large

variation in pressures over the network inlet and outlets, FDS uses a simple approach for selecting the fan pressure that essentially considers the system curve to be the inverse of the fan curve. For more complex networks, FDS can generate a system curve. This is done at each time step, as the system curve depends on the specific configuration of the HVAC system (status of fans and dampers) and the pressures at the inlets and outlets to the system. The operation point of a fan is then set by finding the intersection of that fan's curve with that fan's system curve (i.e., a curve built by finding the flow rate in the HVAC system for various pressures added at the location of the fan). This approach significantly increases the cost of the HVAC solver as it must now solve a system curve for multiple fan pressures, determine the operation point, and then perform the HVAC solution used for wall boundary conditions. However, since the HVAC solver is generally a very small portion of the overall cost of a simulation, this increased cost should not significantly impact the overall run time.

This method is turned on by setting `HVAC_QFAN=T` on the `MISC` line.

10.2.5 HVAC Filter Parameters

A filter must be located at a node with two connected ducts. A filter cannot be located at an ambient node, a node that is attached to a `VENT`, or node with three or more ducts. A sample input for a filter is given by:

```
&HVAC TYPE_ID='FILTER', ID='filter 1', LOADING=0., SPEC_ID='SOOT',  
      EFFICIENCY=0.99, LOADING_MULTIPLIER=1, CLEAN_LOSS=2., LOSS=100./
```

where:

- `AREA` is the area in m^2 associated with the flow loss measurement. Typically this is the area of the filter. If not provided, the average of the two attached duct areas will be used.
- `CLEAN_LOSS` is the dimensionless loss coefficient for flow through the filter when it is clean (zero loading).
- `EFFICIENCY` is an array of the species removal efficiency from 0 to 1 where 0 is no removal of that species and 1 is complete filtration of the species. The species are identified using `SPEC_ID`.
- `LOADING` is an array of the initial loading (kg) of the filter for each species being filtered.
- `LOADING_MULTIPLIER` is an array of the species multiplier, M_i , used in computing the total filter loading when computing the loss coefficient of the filter.
- `LOSS` invokes a linear loss coefficient model where the dimensionless loss coefficient, K , is given as a linear function of the total loading, $K_{\text{FILTER}} = K_{\text{CLEAN_LOSS}} + K_{\text{LOSS}} \sum (L_i M_i)$, where L_i is the species loading and M_i is a multiplier. Only one of `LOSS` or `RAMP_ID` should be specified.
- `RAMP_ID` identifies the `RAMP` that contains a table of pressure drop across the filter as a function of total loading (the summation term given in the definition of `LOSS` above). Only one of `LOSS` or `RAMP_ID` should be specified.
- `SPEC_ID` identifies the tracked species for the inputs of `LOADING_MULTIPLIER` and `LOADING`.

A sample set of filter inputs is shown below. These lines define a filter that removes the species `PARTICULATE` with 100 % efficiency. The filter has an initial loss coefficient of 1 and that loss increases by a factor of 7332 for each kg of `PARTICULATE` captured by the filter. For further details see the sample case `HVAC_filter`, which is documented in the Verification Guide.

```
&SPEC ID='PARTICULATE',MW=28.,MASS_FRACTION_0=0.001,SPECIFIC_HEAT=1./
&HVAC TYPE_ID='NODE',ID='FILTER',DUCT_ID='DUCT1','DUCT2',XYZ(3)=0.55,
FILTER_ID='FILTER'/
&HVAC TYPE_ID='FILTER',ID='FILTER',CLEAN_LOSS=1.,SPEC_ID='PARTICULATE',EFFICIENCY=1.,
LOSS=7732.446,LOADING_MULTIPLIER=1./
```

Note that a filter input refers to a class of filters and that multiple ducts can reference the same filter definition.

10.2.6 HVAC Aircoil Parameters

An aircoil refers to a device that either adds or removes heat from air flowing through a duct. In a typical HVAC system this is done by blowing the air over a heat exchanger (hence the term aircoil) containing a working fluid such as chilled water or a refrigerant. A sample input line is as follows:

```
&HVAC TYPE_ID='AIRCOIL', ID='aircoil 1', EFFICIENCY=0.5,
COOLANT_SPECIFIC_HEAT=4.186, COOLANT_TEMPERATURE=10., COOLANT_MASS_FLOW=1./
```

where:

- `COOLANT_MASS_FLOW` is the flow rate of the working fluid (kg/s).
- `COOLANT_SPECIFIC_HEAT` is the specific heat (kJ/(kg K)) of the working fluid.
- `COOLANT_TEMPERATURE` is the inlet temperature of the working fluid (°C).
- `EFFICIENCY` is the heat exchanger efficiency, η , from 0 to 1. A value of 1 indicates the exit temperatures on both sides of the heat exchanger will be equal.
- `FIXED_Q` is the constant heat exchange rate. A negative value indicates heat removal from the duct. The heat exchange rate can be controlled by either `RAMP_ID` or by `TAU_AC`.
- `TAU_AC` defines a tanh ($TAU_AC > 0$) or t^2 ramp ($TAU_AC < 0$) for the aircoil. This is applied to the `FIXED_Q` of the aircoil. Alternatively, a `RAMP_ID` can be given.

Note that either `FIXED_Q` or the set `COOLANT_SPECIFIC_HEAT`, `COOLANT_MASS_FLOW`, `COOLANT_TEMPERATURE`, and `EFFICIENCY` should be specified. In the latter case, the heat exchange is computed as a two step process. First, the outlet temperature is determined assuming 100 % efficient (i.e., both fluids exit at the same temperature):

$$T_{\text{fluid,out}} = \frac{c_{p,\text{gas}} u_{\text{duct}} A_{\text{duct}} \rho_{\text{duct}} T_{\text{duct,in}} + c_{p,\text{fluid}} \dot{m}_{\text{fluid}} T_{\text{fluid,in}}}{c_{p,\text{gas}} u_{\text{duct}} A_{\text{duct}} \rho_{\text{duct}} + c_{p,\text{fluid}} \dot{m}_{\text{fluid}}} \quad (10.5)$$

Second, the actual heat exchanged is computed using the `EFFICIENCY`.

$$\dot{q}_{\text{coil}} = \eta c_{p,\text{fluid}} \dot{m}_{\text{fluid}} (T_{\text{fluid,in}} - T_{\text{fluid,out}}) \quad (10.6)$$

Note that an aircoil input refers to a class of aircoils and that multiple ducts can reference the same aircoil definition.

The sample input file `HVAC_aircoil.fds` demonstrates the use of the aircoil inputs. A constant flow duct removes air (defined as 28 g/mol with a specific heat of 1 kJ/(kg K)) from the floor and injects in through the ceiling at a volume flow rate of 1 m³/s. An aircoil is defined with a working fluid flowing at 10 kg/s and 100 °C with a specific heat of 4 kJ/(kg K). The aircoil has an efficiency of 50 %. Using the above equations the aircoil will add 45.2 kW of heat to the gas flowing through the duct resulting in a duct exit temperature of 332 K. These results are shown in Fig. 10.8.

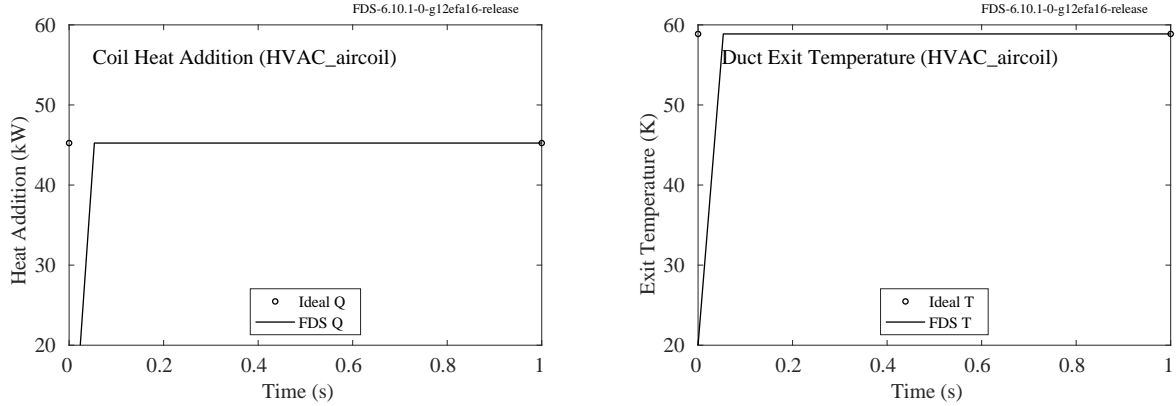


Figure 10.8: (Left) Heat addition and (Right) duct exit temperature for the HVAC_aircoil case.

10.2.7 Louvered HVAC Vents

The HVAC system being modeled may either have louvers that redirect the flow leaving a vent or the orientation of the real vent may not lie along one of the axes in FDS. To define the flow direction for an HVAC outlet, you can use the keyword `UVW` on `VENT`. `UVW` is the vector indicating the direction of flow from the `VENT`. For example:

```
&OBST XB=1.0,2.0,0.0,1.0,0.0,1.0 /
&VENT XB=1.0,1.0,0.0,1.0,0.0,1.0, SURF_ID='HVAC', ID='HVAC OUTLET', UVW=-1,0,1 /
```

The above input defines a vent lying in the y - z plane facing in the $-x$ direction. The flow vector indicates that the flow from this vent is in the $-x$ direction with a 45 degree up angle (the x and z components are equal in size). FDS will set the tangential velocity of the vent to obtain the specified direction indicated by `UVW`. This will only be done if the vent is inputting gas into the domain. Note that the values given for `UVW` are normalized to a unit vector.

10.2.8 HVAC Mass Transport

FDS can account for the time delay of species and enthalpy transport through HVAC networks by discretizing individual HVAC ducts. This can be done using either `N_CELLS` on an HVAC duct input or by setting `HVAC_MASS_TRANSPORT_CELL_L` on `MISC`. The input `N_CELLS` divides the length on the duct into `N_CELLS` uniform segments. The parameter `HVAC_MASS_TRANSPORT_CELL_L` defines a cell length that is applied to all ducts in an input file unless overridden by `N_CELLS`. When specified, the number of cells for a duct is the duct `LENGTH` divided by `HVAC_MASS_TRANSPORT_CELL_L` rounded to the nearest integer (minimum of 1 cell). The current HVAC solver does not account for heat loss in the HVAC network.

If a duct is discretized, then all other ducts in that ductrun must also be discretized. There is no requirement that the same cell size be used for all ducts in a ductrun. A ductrun is any collection of interconnected HVAC components where a path exists between two components via traversing ducts in the ductrun.

If you are using `DEVCS` to output spatially-integrated statistics as per Sec. 22.2.3 (such as `VOLUME MEAN`, `VOLUME INTEGRAL` or `MASS INTEGRAL`) then be aware that, even if the integration volume bound by `XB` encapsulates the spatial location of a duct, the quantity in the duct will not be recorded by the `DEVC`. For example, if there is 1 m^3 of species 1 initialized in an upstream FDS compartment which is transported to a downstream FDS compartment via an HVAC network with a total volume greater than 1 m^3 , the value of total mass of species 1 output by a `DEVC` recording the `VOLUME INTEGRAL` of `DENSITY` will reduce to zero

during the time for which it is in the HVAC network domain.

By default, cells in a mass transport duct are initialized to ambient values. A duct can be initialized a duct to other values by specifying `NODE_ID` on `INIT` plus `TEMPERATURE` and/or `SPEC_ID` plus `VOLUME_FRACTION` or `MASS_FRACTION`. The values in the duct will then be set by linear interpolation of the values at the duct nodes.

10.2.9 Specified Flow vs. Unspecified Flow

There are two basic approaches for defining HVAC networks in FDS.

In the first approach, one can specify enough `VOLUME_FLOW` and `MASS_FLOW` inputs such that all flows everywhere in the network are known. For example, if a tee has three ducts and two of the ducts have specified flow, then the third duct flow is known based on conservation of mass at that node. If all flows are specified everywhere in the HVAC network, then `LOSS` inputs for ducts and nodes are not needed. Do not over specify flows. For example, if a node has three ducts, do not specify `VOLUME_FLOW` or `MASS_FLOW` for more than two of the ducts. This ensures that any numerical round-off error does not result in conservation errors.

In the second approach, one or more ducts have flows that are not specified, in this case FDS must solve for the pressures at either end of the duct to determine the flow through the duct. As one example, if a tee has three ducts and only one of the ducts has a specified flow, then FDS will use the relative pressure drops along the two other ducts to determine the flow. If no `LOSS` inputs are given, then FDS may not correctly solve for the flow. As another example, losses in the HVAC network limit how quickly flow in the ducts can change over time. If there is a single duct connecting two rooms with no `LOSS` inputs given, then small pressure changes can lead to large changes in duct velocity and increase the risk of a numerical instability. If you specify an HVAC network where flow is being solved for by FDS, then you must provide `LOSS` inputs for each possible flow path. FDS will perform a check at startup and return an error message if it finds insufficient losses have been specified; however, this check may not discover all cases.

10.3 Sealed Compartments, Leakage, and Void Spaces

All buildings have spaces that are relatively sealed off from the larger interior, like plenum spaces, closed rooms, HVAC ducts, and so on. In reality, these spaces are connected via seams and gaps such that none are perfectly sealed. However in FDS, these spaces are assumed to be perfectly sealed unless you say otherwise. This section introduces the concept of a “pressure zone” and how to work with them, in particular how to handle leakage.

These spaces are important because FDS assumes pressure to be composed of a “background” component, $\bar{p}(z, t)$, plus a perturbation pressure, $\tilde{p}(\mathbf{x}, t)$. Most often, \bar{p} is just the hydrostatic pressure, and \tilde{p} is the flow-induced spatially-resolved perturbation. If the computational domain has no sealed void spaces and there is at least one `OPEN` boundary, FDS can assume that the background pressure is simply atmospheric pressure and that it is the same throughout the entire domain. However, if void spaces exist, their background pressures will be different. For example, if a fire starts in a sealed compartment, the background pressure must rise above atmospheric.

FDS has an algorithm that identifies regions within the computational domain that are sealed; that is, are not connected to an `OPEN` boundary. Thus, it is not necessary for you to explicitly declare these regions, or “pressure zones.” However, if you desire that a pressure zone be connected to another via a leak path, you need to explicitly declare it. The next section discusses how to do that.

10.3.1 Specifying Pressure Zones

A pressure zone can be any region within the computational domain that is separated from the rest of the domain, or the exterior, by solid obstructions. There is an algorithm within FDS to identify these zones based solely on your specified obstructions. Consequently, it is not necessary that you identify these zones explicitly in the input file. However, you might want to have a particular zone leak to another zone or to the exterior of the domain. In that case, the basic syntax for a pressure `ZONE` is:

```
&ZONE XYZ=1.2,3.4,5.6, LEAK_AREA(0)=0.001 /
```

The parameter `XYZ` specifies a single point (1.2,3.4,5.6) that is within a sealed compartment, and it is not embedded within a solid obstruction. There can be multiple `ZONES` declared. The indices of the zones, which are required for the specification of leaks and fans, are determined simply by the order in which they are specified in the input file. By default, the exterior of the computational domain is Zone 0. If there are no `OPEN` boundaries, the entire computational domain will be assumed to be Zone 1.

There are several restrictions to assigning pressure zones. First, the declared pressure zones must be completely within a region of the domain that is bordered by solid obstructions. “Breaking” pressure zones by removing obstructions between them is possible. An example of how to break pressure zones is given below. Second, pressure zones can span multiple meshes, but checking the pressures in each mesh to ensure consistency is recommended. If the `ZONE` does span multiple meshes, you do *not* need to add multiple `XYZ` points for each mesh. A search algorithm will determine all other grid cells belonging to that pressure zone.

Note that removing a solid obstruction (like a door or window) separating two pressure zones will result in FDS merging the two zones into one such that the background pressure within each remains nearly the same.

Example Case: Pressure Rise in a Compartment

This example tests several basic features of FDS. A narrow channel, 3 m long, 0.002 m wide, and 1 m tall, has air injected at a rate of 0.1 kg/m²/s over an area of 0.2 m by 0.002 m for 60 s, with a linear ramp-up and ramp-down over 1 s. The total mass of air in the channel at the start is 0.00718 kg. The total

mass of air injected is 0.00244 kg. The domain is assumed two-dimensional, the walls are adiabatic, and STRATIFICATION is set to F simply to remove the slight change in pressure and density with height. The domain is divided into three meshes, each 1 m long and each with identical gridding. Pressure, temperature, and density are expected to rise during the 60 s injection period. Afterwards, the temperature, density, and pressure should remain constant, according to the equation of state. Figure 10.9 shows the results of this calculation. The density matches exactly showing that FDS is injecting the appropriate amount of mass. The steady state values of the pressure, density and temperature are consistent with the ideal values obtained from the first law of thermodynamics.

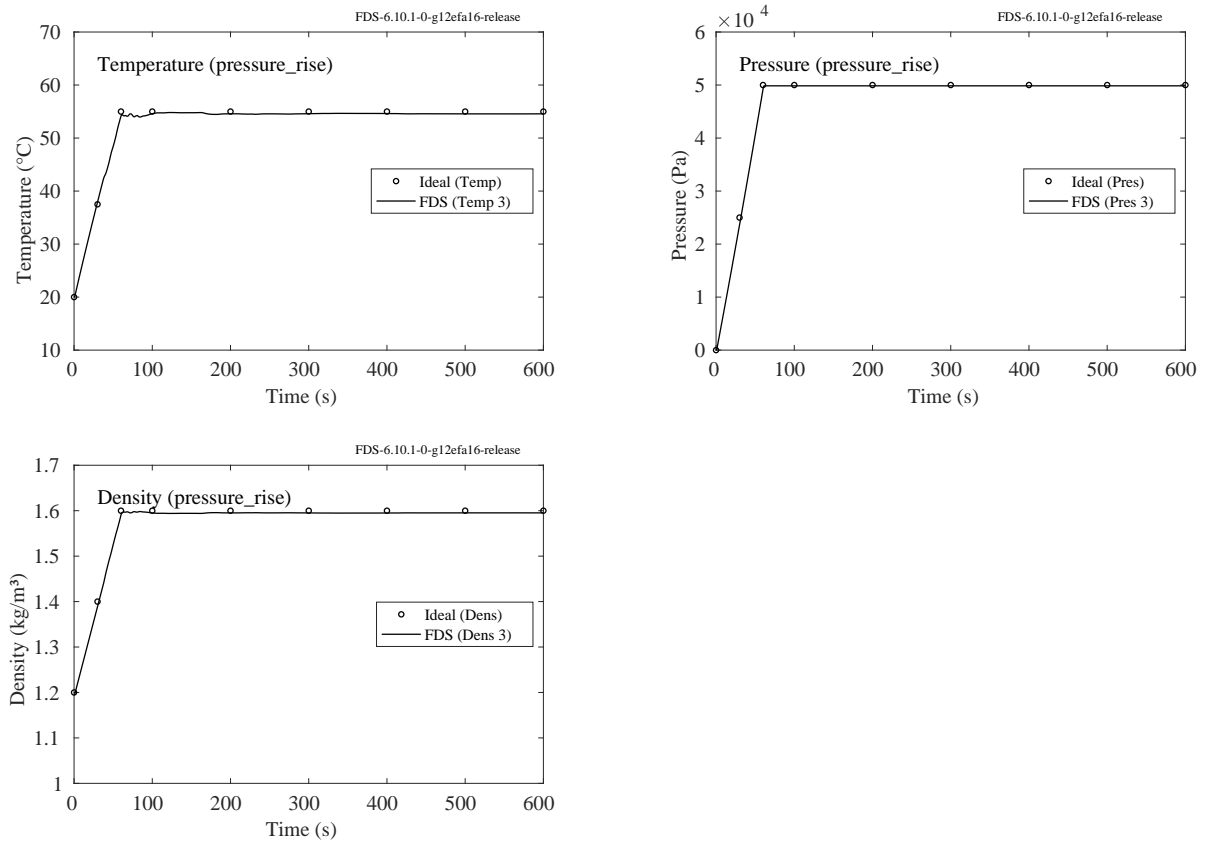


Figure 10.9: Output of pressure_rise test case.

Example Case: Breaking Pressure Zones

In this example, three simple compartments are initially isolated from each other and from the ambient environment outside. Each compartment is a separate pressure zone. Air is blown into Zone 1 at a constant rate of 0.1 kg/s, increasing its pressure approximately 2000 Pa by 10 s, at which time Zone 1 is opened to Zone 2, decreasing the overall pressure in the two zones to roughly one-third the original value because the volume of the combined pressure zone has been roughly tripled. At 15 s, the pressure is further decreased by opening a door to Zone 3, and, finally, at 20 s the pressure returns to ambient following the opening of a door to the outside. Figure 10.10 displays the pressure within each compartment. Notice that the pressures do not come to equilibrium instantaneously. Rather, the PRESSURE_RELAX_TIME (on the PRES line) is applied to bring the zones into equilibrium over a specified period of time. This is done for several reasons. First, in reality doors and windows do not magically disappear as they do in FDS. It takes a finite amount of time

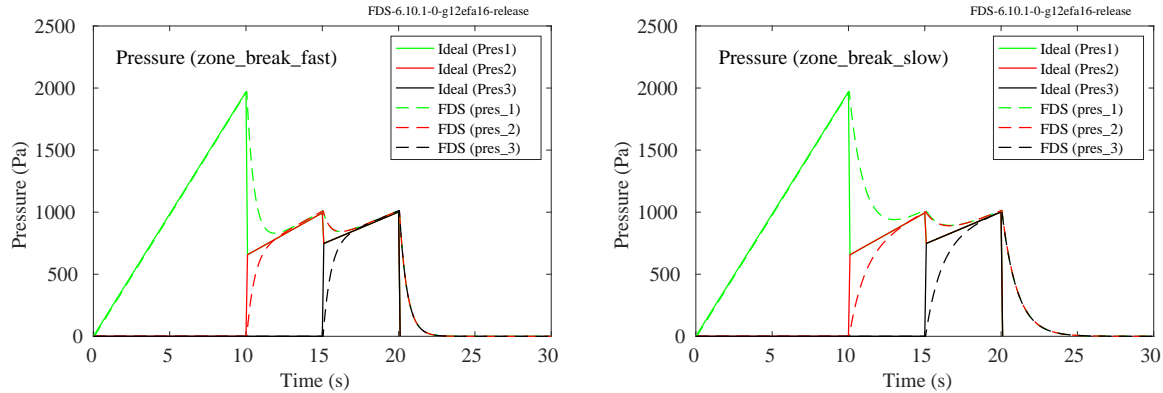


Figure 10.10: Output of `zone_break` test cases. The figure on the left results from using a pressure relaxation time of 0.5 s. The figure on the right uses 1 s, the default.

to fully open them, and the slowing of the pressure increase/decrease is a simple way to simulate the effect. Second, relatively large pressure differences between zones wreak havoc with flow solvers, especially ones like FDS that use a low Mach number approximation. To maintain numerical stability, FDS gradually brings the pressures into equilibrium. This second point ought to be seen as a warning.

Do not use FDS to study the sudden rupture of pressure vessels! Its low Mach number formulation does not allow for high speed, compressible effects that are very important in such analyses. The zone breaking functionality described in this example is only intended to be used for relatively small pressure differences (<0.1 atm) between compartments. Real buildings cannot withstand substantially larger pressures anyway.

Example Case: Irregularly Shaped Zone

This example is similar to the ones in the previous section, except in this case, the pressure zone is L-shaped and split across two meshes. The objective is simply to ensure that the specification of the pressure zone is properly accounted for in the model. Figure 10.11 compares the predicted pressure in the compartment compared to an exact solution. Air is injected into the compartment for 5 s, after which the compartment is opened to a smaller compartment. At 15 s, the smaller compartment is opened to the outside.

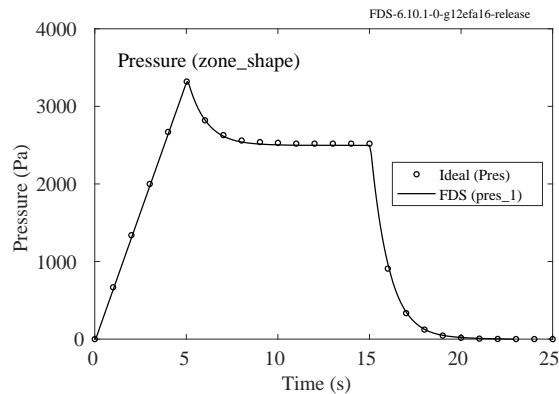


Figure 10.11: Output of `zone_shape` test case. Shown is the pressure in an L-shaped compartment that is opened to another compartment at 5 s, and the outdoors at 15 s.

10.3.2 Leaks

With a few notable exceptions, like containment buildings for nuclear power plants, real world construction is not air tight. Small gaps occur along windows and doors and where walls abut each other and the floors and ceilings. As a compartment is pressurized by a fire, air will escape through these small gaps. This is referred to as leakage.

Leakage is inherently a subgrid scale phenomenon because the leakage area is usually very small. In other words, it is not possible to define a leak directly on the numerical mesh. It is sometimes possible to “lump” the leaks into a single mesh-resolvable hole, but this is problematic for two reasons. First, the leakage area rarely corresponds neatly to the area of a single mesh cell-sized hole. Second, the flow speeds through the hole can be large and cause numerical instabilities.

A better way to handle leakage is by exploiting the HVAC model. The compartment surface that is leaking can be thought of as a large HVAC vent that connects via a very small duct to the outside. This allows the leakage to be removed over a large area in the domain (just as it would be in reality) while correctly capturing the actual area of the leakage path. There are two approaches to this. The first approach is by exploiting only pressure zones. A pressure zone is a user-specified volume within the computational domain that is entirely surrounded by solid obstructions. For example, the interior of a closed room can be, and should be, declared a pressure zone. In this approach surfaces within a pressure zone are denoted as leaking, and those surfaces can be considered an HVAC vent that connects to the outside via a tiny duct whose area is the leakage area. This leakage approach will prevent a compartment from seeing large pressure changes as fires grow and decay, but it cannot account for effects like exterior wind or the stack effect. The second approach is intended for leaks with well defined locations (a cracked open door where the crack size is subgrid) or for leaks where the stack effect is important. It uses the local pressure (which includes the zone pressure), which allows for leakage to vary in magnitude.

Pressure Zone Leakage

The pressure zone leakage approach is intended to capture the bulk leakage that occurs through walls. This approach assumes that the amount of leaking gas is very small, and that it will exchange sufficient heat as it moves through a wall to be at the same temperature as the wall surface. With this approach the pressure between the source and destination zones is used to compute a leakage flow via the HVAC model. That flow is then uniformly imposed over all surfaces designated as part of the leakage path. The first step is to define pressure zones, leakage areas, and a description of the surfaces through which leaking occurs:

```
&ZONE XYZ=..., LEAK_AREA(0)=0.0001 /
&ZONE XYZ=..., LEAK_AREA(1)=0.0002, LEAK_AREA(0)=0.0003 /
&SURF ID='LEAKY EXTERIOR WALL',..., LEAK_PATH=1,0 /
&SURF ID='LEAKY INTERIOR WALL',..., LEAK_PATH=1,2 /
```

The first line designates a region of the computational domain to be Pressure Zone 1. Note that the order of the `ZONE` lines is important; that is, the order implicitly defines Zone 1, Zone 2, etc. Zone 0 is by default the ambient pressure exterior. In this example, a leak exists between Zone 1 and the exterior Zone 0, and the area of the leak is 0.0001 m^2 (1 cm by 1 cm hole, for example). Zone 2 leaks to Zone 1 (and vice versa) with a leak area of 0.0002 m^2 . Zone 2 also leaks to the outside with an area of 0.0003 m^2 . Note that zones need not be physically connected for a leak to occur, but in each zone, except for the exterior, there must be some surface with a designated `LEAK_PATH`. Here, in Zones 1 and 2 there are surfaces defined by both 'LEAKY EXTERIOR WALL' and 'LEAKY INTERIOR WALL' so as to provide a surface over which to apply the leakage. Leakage is uniformly distributed over all of the solid surfaces assigned the `LEAK_PATH`. The order of the two pressure zones designated by `LEAK_PATH` is unimportant, and the solid obstructions where the

leakage is applied need not form a boundary between the two zones. Note that `LEAK_PATH_ID` can be used instead of `LEAK_PATH`. To use `LEAK_PATH_ID` provide the `ID` for the two ZONES where specifying 'AMBIENT' indicates Zone 0.

The volume flow, \dot{V} , through a leak of area A_L is given by

$$\dot{V}_{\text{leak}} = C_d A_L \text{sign}(\Delta p) \sqrt{2 \frac{|\Delta p|}{\rho_\infty}} \quad (10.7)$$

where C_d is a discharge coefficient, Δp is the pressure difference (Pa) between the adjacent compartments and ρ_∞ is the ambient density (kg/m^3). The discharge coefficient normally seen in this type of formula is sometimes assumed to be 1, but you may change it.

As the interior pressure rises in a typical building, the leakage area grows as small gaps, cracks, and other leakage paths open up. Leakage tests performed according to test standards such as ASTM E779 provide two additional data points to quantify this behavior. These are the `LEAK_PRESSURE_EXPONENT` and the `LEAK_REFERENCE_PRESSURE`. The use of these additional inputs are shown in the equation below as n and Δp_{ref} respectively where $A_{L,\text{ref}}$ is given by `LEAK_AREA`.

$$A_L = A_{L,\text{ref}} \left(\frac{|\Delta p|}{\Delta p_{\text{ref}}} \right)^{n-0.5} \quad (10.8)$$

By default, $n = 0.5$ and $\Delta p_{\text{ref}} = 4$ Pa, meaning that the leak area will not change with pressure. The `DISCHARGE_COEFFICIENT` is 1 by default.

The HVAC output quantities can be used to determine the leakage flows. FDS names the duct connecting Zone A with Zone B 'LEAK A B' and the duct nodes 'LEAK A B' for the Zone A side of the leak and 'LEAK B A' for the Zone B side. Note that for the duct names, FDS will use the lower numbered zone as Zone A.

FDS is limited by default to a maximum of 200 ZONE inputs. This can be increased if needed by the parameter `MAX_LEAK_PATHS` on the `MISC` line.

Localized Leakage

The localized leakage approach addresses situations where the leakage is not necessarily from one sealed pressure zone to another. For example, the gap created by a partially opened door might be too narrow to explicitly resolve, but the hot gases escaping at the top and cold gases entering at the bottom may still be represented in the model. The localized leakage approach uses the local pressure rather than the mean zone pressure, allowing one to define multiple leakage paths for windows and doors that might span a significant height, like in a stairwell where stack effect might be important. To use this approach two `VENT` lines are linked via an `HVAC` line with `TYPE_ID='LEAK'`. For example, the input lines

```
&VENT XB=..., SURF_ID='SURF 1', ID='VENT 1' /
&VENT XB=..., SURF_ID='SURF 2', ID='VENT 2' /
&HVAC ID='LEAK1', TYPE_ID='LEAK', VENT_ID='VENT 1', VENT2_ID='VENT 2', AREA=0.001 /
```

specify a 0.001 m^2 leak between the `VENTS` named 'VENT 1' and 'VENT 2'. If the leakage path is to outside of the domain, set `VENT2_ID='AMBIENT'`, in which case the second node will be assigned the name of the first with 'AMB' added, e.g. 'VENT 1 AMB'. Each `VENT` must be given an explicit `SURF_ID`. Wall heat transfer will be computed based upon the specified `SURF_ID`. Following are some rules and practices regarding localized leakage:

- The `SURF_ID` for a `VENT` with localized leakage is *not* 'HVAC'. Each `VENT` must be given an explicit `SURF_ID`.

- Each VENT must lie in one pressure zone; however, it may span more than one MESH.
- You may add the parameters LEAK_PRESSURE_EXPONENT, LEAK_REFERENCE_PRESSURE, and DISCHARGE_COEFFICIENT to the HVAC line, according to Eqs. (10.7) and (10.8).
- All localized leakage HVAC components are given the NETWORK_ID='LEAK'.
- A VENT used for localized leakage should not be given a SURF_ID with a LEAK_PATH input.
- The localized leakage area should not also be part of the LEAK_AREA defined on a ZONE line.
- A VENT used for localized leakage should not have any other input defining a mass flow.
- Particles can be transported through a localized leakage path by setting TRANSPORT_PARTICLES=T on the HVAC line. This requires that the two VENTS have identical area and orientation and offset by no more than one grid cell.
- A localized leakage VENT must be present for the entire simulation. If you wish to have, for example, a door that leaks when closed but is opened during the simulation, then define the VENTS for the leakage to be on the floor, above the door, or to the side of the door. The leakage path will still be present with the door opened, but the pressure across the leak path will be much lower and the leak flow will be insignificant compared to the doorway flow.
- Unlike the zone leakage approach, localized leakage has the option to preserve the energy of the gas flowing through the leak. For example, for a door crack using pressure zone leakage, the outflowing gas at the top of the door takes on the same temperature as the outside. To maintain hot gas flowing out of the leak, add LEAK_ENTHALPY=T to the HVAC input. For each outflowing wall cell, this will compute the enthalpy difference between the temperature of the leak flow and the temperature of the surface and add it as a source of heat to the adjacent gas cell. The default value is LEAK_ENTHALPY=F.
- Localized leakage has the option of changing the flow loss by specifying LOSS on the HVAC input. The default is LOSS=1, the same as for pressure zone leakage.

Example Case: door_crack

This example involves a small compartment that contains a fan in one wall and a closed door with leakage at its bottom in the opposite wall. A small (160 kW) fire is added to the compartment. Initially, the pressure rises due to the heat from the fire and the fan blowing air into the compartment. Eventually the pressure rise inside the compartment exceeds the maximum pressure of the fan, at which point the compartment begins to exhaust from both the fan and the leakage. Pressure will continue to rise due to the fire until the pressure relief due to leakage and back flow through the fan equals the pressure increase from the fire.

10.3.3 Breaking Pressure Zones

There are two parameters on the PRES line that control iterative procedures related to the coupling of velocity and pressure. One is called RELAXATION_FACTOR and its default value is 1. When there is an error in the normal component of velocity at a solid boundary, this parameter dictates that the correction be applied in 1 time step. If its value were 0.5, the correction would be applied in 2 time steps.

A similar parameter is the PRESSURE_RELAX_TIME. It controls the rate at which the pressures in adjacent compartments are brought into equilibrium following a breach. Its default value is 1 s, meaning that equilibrium is achieved in roughly a second.

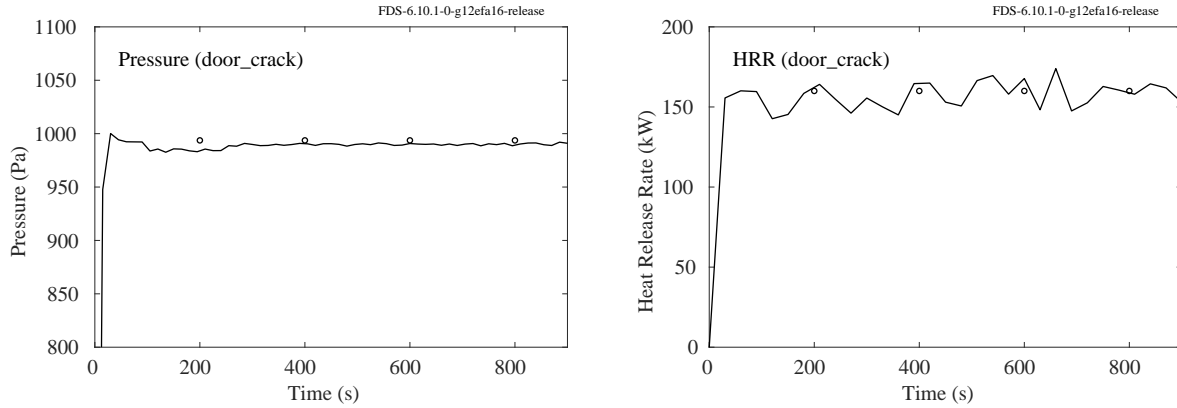


Figure 10.12: Output of `door_crack` test case. Symbols are expected values.

10.3.4 Filling Pressure Zones

Sometimes a complicated geometry might have relatively small void spaces that are not connected to other parts of the domain or `OPEN` boundaries. Sometimes these spaces are desired; sometimes not. That is, sometimes they are artifacts of the geometry generation software. If you do not want these void spaces within the domain, you can set a parameter called `MINIMUM_ZONE_VOLUME` on the `MISC` line. This parameter, which has units of m^3 , tells FDS to fill in all void spaces less than this minimum volume. By default, its value is 0 m^3 , meaning that no voids are filled. If you make its value huge, all voids will be filled.

On rare occasions completely suppressing pressure `ZONES` completely may be required. Although not recommended, this approach might be useful for debugging or diagnostic purposes. To remove all implicit `ZONES`, set `NO_PRESSURE_ZONES` on the `MISC` line. This is not the same as setting a high value for `MINIMUM_ZONE_VOLUME`. Rather, it tells FDS to simply ignore the void spaces and use the same background pressure throughout the domain.

10.4 Pressure Boundary Conditions

In some situations, it is more convenient to specify a pressure, rather than a velocity, at a boundary. Suppose, for example, that you are modeling the interior of a tunnel, and a wind is blowing at one of the portals that affects the overall flow within the tunnel. If (and only if) the portal is defined using an `OPEN` vent, then the *dynamic pressure* at the boundary can be specified like this:

```
&VENT XB=..., SURF_ID='OPEN', DYNAMIC_PRESSURE=2.4, PRESSURE_RAMP='wind' /
&RAMP ID='wind', T= 0.0, F=1.0 /
&RAMP ID='wind', T=30.0, F=0.5 /
.
.
```

The use of a *dynamic pressure* boundary affects the FDS algorithm as follows. At `OPEN` boundaries, the hydrodynamic pressure (head) H is specified as

$$\begin{aligned} H &= \text{DYNAMIC_PRESSURE} / \rho_{\infty} + |\mathbf{u}|^2 / 2 \quad (\text{outgoing}) \\ H &= \text{DYNAMIC_PRESSURE} / \rho_{\infty} \quad (\text{incoming}) \end{aligned} \quad (10.9)$$

where ρ_{∞} is the ambient density and \mathbf{u} is the most recent value of the velocity on the boundary. The `PRESSURE_RAMP` allows for alteration of the pressure as a function of time. Note that you do not need to ramp the pressure up or down starting at zero, like you do for various other ramps. The net effect of a positive dynamic pressure at an otherwise quiescent boundary is to drive a flow into the domain. However, a fire-driven flow of sufficient strength can push back against this incoming flow.

The following lines, taken from the sample case, `pressure_boundary`, demonstrates how to specify a time-dependent pressure boundary at the end of a tunnel. The tunnel is 10 m long, 1 m wide, 1 m tall with a fire in the middle and a pressure boundary imposed on the right side. The left side (`XMIN`) is just an `OPEN` boundary with no pressure specified. It is assumed to be at ambient pressure.

```
&VENT MB = 'XMIN' SURF_ID = 'OPEN' /
&VENT MB = 'XMAX' SURF_ID = 'OPEN', DYNAMIC_PRESSURE=2.4, PRESSURE_RAMP='wind_ramp' /
&RAMP ID='wind_ramp', T= 0., F= 1. /
&RAMP ID='wind_ramp', T=15., F= 1. /
&RAMP ID='wind_ramp', T=16., F=-1. /
```

Figure 10.13 shows two snapshots from Smokeview taken before and after the time when the positive pressure is imposed at the right portal of a tunnel. The fire leans to the left because of the preferential flow in that direction. It leans back to the right when the positive pressure is directed to become negative.



Figure 10.13: Snapshots from the sample case `pressure_boundary` showing a fire in a tunnel leaning left, then right, due to a positive, then negative, pressure imposed at the right portal.

10.5 Special Flow Profiles

By default, the air injected at a vent has a uniform or “top hat” velocity profile, but the parameter `PROFILE` on the `SURF` line can yield other profiles.

Parabolic

`PROFILE='PARABOLIC'` produces a parabolic profile with `VEL` (m/s) being the maximum velocity or `VOLUME_FLOW` (m³/s) being the desired volume flow. As an example, the test case in the `Flowfields` examples folder called `parabolic_profile.fds` demonstrates how you can create a circular or rectangular vent, each with a parabolic inlet profile. The two `VENT` lines below create circular and rectangular inlets, respectively, each of which inject air (or the background gas) at a rate of 0.5 m³/s *into* the compartment.

```
&SURF ID='BLOW', VOLUME_FLOW=-0.5, PROFILE='PARABOLIC' /
&VENT SURF_ID='BLOW', XB=-3,1,-3,1,0,0, RADIUS=2., XYZ=-1,-1,0 /
&VENT SURF_ID='BLOW', XB= 3,5,-2,1,0,0 /
```

The purpose of the test case is to ensure that the proper amount of gas (in this case nitrogen) is forced into the compartment, as confirmed by the pressure rise. Figure 10.14 displays a comparison of the calculated versus the exact pressure rise in a large compartment with these two parabolic vents. The pressure should rise according to the equation and analytical solution:

$$\frac{dp}{dt} = \frac{\gamma \dot{V}}{V} p \implies p(t) - p_0 = p_0 \left(e^{\frac{\gamma \dot{V}}{V} t} - 1 \right) \quad (10.10)$$

where the ratio of specific heats, $\gamma = 1.4$, volume flow rate, $\dot{V} = 1$ m³/s, volume, $V = 4000$ m³, and ambient pressure, $p_0 = 101325$ Pa. Note that to obtain this simple result, FDS was run with the option `CONSTANT_SPECIFIC_HEAT_RATIO` set to true.

Boundary Layer (Circular Vent)

`PROFILE='BOUNDARY LAYER'` may be used for circular vents created using `RADIUS`. By adding `VEL_BULK` on the `SURF` line together with `VEL`, FDS will produce a plug flow core profile, with max velocity given by `VEL`, and a quadratic profile in the boundary layer. The form of the profile is illustrated in Fig. 10.15 for a circular

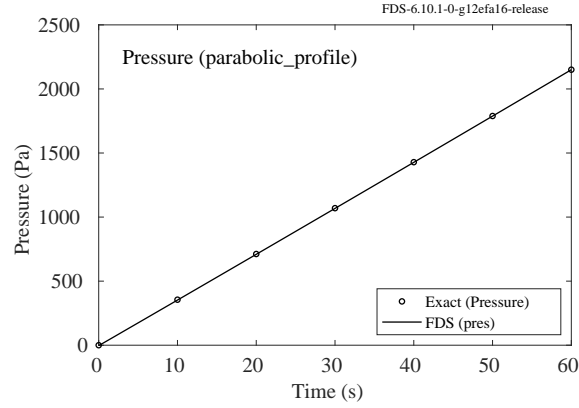


Figure 10.14: Results of the `parabolic_profile` test case

vent. The functional form of the velocity profile (here taken a vertical profile) is

$$w(r) = \begin{cases} w_{\max} & \text{if } r \leq R - \delta \\ w_{\max} \left(1 - \left(\frac{r - (R - \delta)}{\delta} \right)^2 \right) & \text{if } R - \delta < r \leq R \end{cases} \quad (10.11)$$

The bulk velocity is the volumetric flow rate divided by the circular flow area. `VEL_BULK` is negative pointing into the domain. The boundary layer thickness is δ . This feature is handy for dealing with the case where the maximum velocity is higher than the bulk velocity. Here is an example input file line:

```
&SURF ID='JET', VEL=-69, VEL_BULK=-53, PROFILE='BOUNDARY LAYER', ... /
```

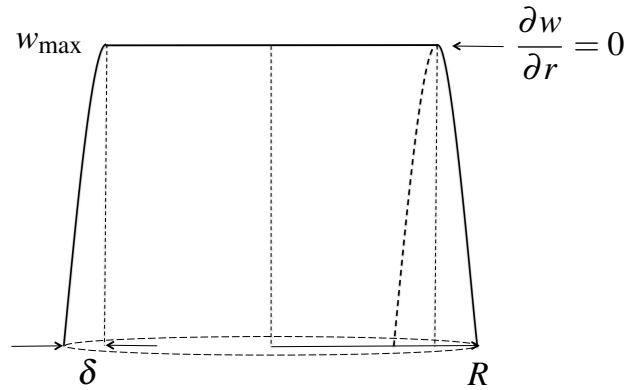


Figure 10.15: Boundary layer profile.

Chapter 11

User-Specified Functions

Many of the parameters specified in the FDS input file are fixed constants. However, there are several parameters that may vary in time, temperature, or space. The namelist groups, `RAMP` and `TABL`, allow you to control the behavior of these parameters. `RAMP` allows you to specify a function with one independent variable (such as time) and one dependent variable (such as velocity). `TABL` allows you to specify a function of multiple independent variables (such as a solid angle) and multiple dependent variables (such as a sprinkler flow rate and droplet speed).

FDS limits the number of `RAMP`s that can be specified in the input file. This limit is controlled by the parameter `MAX_RAMP`s on `MISC`. The default value is 100.

11.1 Time-Dependent Functions

At the start of any calculation, the temperature is ambient everywhere, the flow velocity is zero everywhere, nothing is burning, and the mass fractions of all species are uniform. You can control the rate at which things turn on, or turn off, by specifying time histories either with pre-defined functions or with user-defined functions. The parameters `TAU_Q`, `TAU_T`, and `TAU_V` indicate that the heat release rate (`HRRPUA`); surface temperature (`TMP_FRONT`); and/or normal velocity (`VEL`, `VOLUME_FLOW`), or `MASS_FLUX_TOTAL` are to ramp up to their prescribed values as follows, using the heat release rate, $\dot{Q}(t)$, as an example:

$$\dot{Q}(t) = \begin{cases} \dot{Q}_0 (t/\tau)^2 & \text{if } \text{TAU_Q is negative} \\ \dot{Q}_0 \tanh(t/\tau) & \text{if } \text{TAU_Q is positive} \end{cases} \quad (11.1)$$

\dot{Q}_0 is the user-specified heat release rate. If the fire ramps up following a t -squared curve, then it remains constant after `TAU_Q` seconds. These rules apply to `TAU_T` and `TAU_V` as well. The default value for all `TAU`s is 1 s¹. If something other than a tanh or t -squared ramp up is desired, then a user-defined function can be input. To do this, set `RAMP_Q`, `RAMP_T` or `RAMP_V` equal to a character string designating the ramp function to use for that particular surface type, then somewhere in the input file generate lines of the form:

```
&RAMP ID='rampname1', T= 0.0, F=0.0 /  
&RAMP ID='rampname1', T= 5.0, F=0.5 /  
&RAMP ID='rampname1', T=10.0, F=0.7 /
```

Here, `T` is the time, and `F` indicates the fraction of the heat release rate, wall temperature, velocity, mass

¹You can change the default ramp-up time by setting `TAU_DEFAULT` on the `MISC` line.

fraction, etc., to apply. Linear interpolation² is used to fill in intermediate time points. Note that each set of RAMP lines must have a unique ID and that the lines must be listed with monotonically increasing T. Note also that the TAUS and the RAMPs are mutually exclusive. For a given surface quantity, both cannot be prescribed. As an example, a simple blowing vent can be controlled via the lines:

```
&SURF ID='BLOWER', VEL=-1.2, TMP_FRONT=50., RAMP_V='BLOWER RAMP', RAMP_T='HEATER
RAMP' /
&RAMP ID='BLOWER RAMP', T= 0.0, F=0.0 /
&RAMP ID='BLOWER RAMP', T=10.0, F=1.0 /
&RAMP ID='BLOWER RAMP', T=80.0, F=1.0 /
&RAMP ID='BLOWER RAMP', T=90.0, F=0.0 /
&RAMP ID='HEATER RAMP', T= 0.0, F=0.0 /
&RAMP ID='HEATER RAMP', T=20.0, F=1.0 /
&RAMP ID='HEATER RAMP', T=30.0, F=1.0 /
&RAMP ID='HEATER RAMP', T=40.0, F=0.0 /
```

Use TAU_T or RAMP_T to control the ramp-ups for surface temperature. The surface temperature at time t , $T_w(t)$, is

$$T_w(t) = T_0 + f(t)(TMP_FRONT - T_0) \quad (11.2)$$

where $f(t)$ is the result of evaluating the RAMP_T at time t , T_0 is the ambient temperature, and TMP_FRONT is specified on the same SURF line as RAMP_T. Use TAU_MF(N) or RAMP_MF(N) to control the ramp-ups for either the mass fraction or mass flux of species N. For example:

```
&SURF ID='...', MASS_FLUX(1:2)=0.1,0.3, SPEC_ID(1:2)='ARGON','NITROGEN',
TAU_MF(1:2)=5.,10. /
```

indicates that argon and nitrogen are to be injected at rates of 0.1 kg/(m²s) and 0.3 kg/(m²s) over time periods of approximately 5 s and 10 s, respectively.

When a time-base RAMP is evaluated, the time used for the RAMP is the time since the item (e.g., VENT, OBST, DEVC) using the RAMP became active. This means RAMPs use the actual time if the activation time of the item using the RAMP is the same as T_BEGIN (see 6.2.1). Otherwise, they are evaluated using the time from when the RAMP activates. Therefore, if you are setting T_BEGIN in order to test a time-based CTRL or DEVC that is ultimately linked to a RAMP, then you should set T_BEGIN to be slightly less than the time the RAMP will activate. For example, if you are testing a VENT that is to open at 10 s whose SURF_ID uses a RAMP, then T_BEGIN should be set slightly less than 10 s.

Table 11.1 lists the various quantities that can be controlled by RAMPs.

²By default, FDS uses a linear interpolation routine to find time or temperature-dependent values between user-specified points. The default number of interpolation points is 5000, more than enough for most applications. However, you can change this value by specifying NUMBER_INTERPOLATION_POINTS on any RAMP line.

Table 11.1: Parameters for controlling the time-dependence of given quantities.

Quantity	Group	Input Parameter(s)	TAU	RAMP ID
Volume Flow	HVAC	VOLUME_FLOW	TAU_FAN	RAMP_ID
Heating Rate	HVAC	FIXED_Q	TAU_AC	RAMP_ID
Heat Release Rate	SURF	HRRPUA	TAU_Q	RAMP_Q
Heat Flux	SURF	NET_HEAT_FLUX etc.	TAU_Q	RAMP_Q
Temperature	SURF	TMP_FRONT	TAU_T	RAMP_T
Velocity	SURF	VEL	TAU_V	RAMP_V
Volume Flux	SURF	VOLUME_FLOW	TAU_V	RAMP_V
Mass Flux	SURF	MASS_FLUX_TOTAL	TAU_V	RAMP_V
Mass Fraction	SURF	MASS_FRACTION (N)	TAU_MF (N)	RAMP_MF (N)
Mass Flux	SURF	MASS_FLUX (N)	TAU_MF (N)	RAMP_MF (N)
Particle Mass Flux	SURF	PARTICLE_MASS_FLUX	TAU_PART	RAMP_PART
External Heat Flux	SURF	EXTERNAL_FLUX	TAU_EF	RAMP_EF
Pressure	VENT	DYNAMIC_PRESSURE		PRESSURE_RAMP
Flow	PROP	FLOW_RATE	FLOW_TAU	FLOW_RAMP
Gravity	MISC	GVEC (1)		RAMP_GX
Gravity	MISC	GVEC (2)		RAMP_GY
Gravity	MISC	GVEC (3)		RAMP_GZ

11.2 Temperature-Dependent Functions

Thermal properties like conductivity and specific heat can vary significantly with temperature. In such cases, use the `RAMP` function like this:

```
&MATL ID          = 'STEEL'
    FYI            = 'A242 Steel'
    SPECIFIC_HEAT_RAMP = 'c_steel'
    CONDUCTIVITY_RAMP  = 'k_steel'
    DENSITY          = 7850. /

&RAMP ID='c_steel', T= 20., F=0.45 /
&RAMP ID='c_steel', T=377., F=0.60 /
&RAMP ID='c_steel', T=677., F=0.85 /

&RAMP ID='k_steel', T= 20., F=48. /
&RAMP ID='k_steel', T=677., F=30. /
```

Note that for temperature ramps, as opposed to time ramps, the parameter `F` is the actual physical quantity, not just a fraction of some other quantity. Thus, if `CONDUCTIVITY_RAMP` is used, there should be no value of `CONDUCTIVITY` given. Note also that for values of temperature, `T`, below and above the given range, FDS will assume a constant value equal to the first or last `F` specified. Note also that the `DENSITY` of a material cannot be controlled with a `RAMP` function.

11.3 Spatially-Dependent Velocity Profiles

Similar to using `PROFILE='ATMOSPHERIC'` on `SURF`, it is possible to specify `PROFILE='RAMP'` to generate 1D or 2D profiles of the normal component of velocity on a surface. The following code generates a $u(z)$ profile on the `'XMIN'` boundary.

```
&SURF ID='inlet', VEL=-7.72, PROFILE='RAMP', RAMP_V_Z='u_prof' /
&VENT MB='XMIN', SURF_ID='inlet' /

&RAMP ID='u_prof', T=0., F=0. /
&RAMP ID='u_prof', T=0.0098, F=0. /
&RAMP ID='u_prof', T=0.01005, F=0.2474 /
&RAMP ID='u_prof', T=0.01029, F=0.4521 /
&RAMP ID='u_prof', T=0.01077, F=0.6256 /
&RAMP ID='u_prof', T=0.01174, F=0.7267 /
&RAMP ID='u_prof', T=0.01368, F=0.8238 /
&RAMP ID='u_prof', T=0.01562, F=0.8795 /
&RAMP ID='u_prof', T=0.01756, F=0.9378 /
&RAMP ID='u_prof', T=0.0195, F=0.9663 /
&RAMP ID='u_prof', T=0.02144, F=0.9922 /
&RAMP ID='u_prof', T=0.02338, F=0.9987 /
&RAMP ID='u_prof', T=0.02532, F=1 /
&RAMP ID='u_prof', T=0.0588, F=1 /
```

Note that `v` indicates the velocity component normal to the surface. You can also specify `RAMP_V_X` and `RAMP_V_Y` and add these to the `SURF`. Note that only profiles in the planar directions will affect a given surface. That is, if the surface is oriented in the $y-z$ plane, only `RAMP_V_Y` and `RAMP_V_Z` apply. In the `RAMP` definition `T` is the independent variable and `F` is the dependent variable. In this example, `T` is the z coordinate in meters and `F` is the factor multiplying `VEL` on the `SURF` line. Two ramps may be applied to a surface.

`T` does not need to be directly related to the FDS mesh. The velocity points will be interpolated linearly by the ramp function. In fact, this functionality is convenient for taking experimental data directly as a boundary condition to FDS. You basically just need to list `T` and `F` from the data (you can set `VEL=-1` if you want). The results for this particular case are included under the heading “Backward Facing Step” in the FDS Validation Guide [5]. In this problem, the step height is $h = 0.0098$ m, and the specification of the inlet profile is critical to correctly matching the reattachment point downstream of the step.

11.4 Scaling, Rotation and Translation

It is possible to scale, rotate and/or translate various objects within an FDS simulation using the namelist group called `MOVE`. For example, the line

```
&MOVE ID='spin', ROTATION_ANGLE=45., X0=2., Y0=3., Z0=4., AXIS=1,0,0 /
```

causes an object to be rotated 45° about the direction vector (`AXIS`) (1,0,0) emanating from the point (x_0, y_0, z_0) . You can scale the object in its original unrotated axes respect to (x_0, y_0, z_0) , by either defining the factor `SCALE > 0.` for scaling in every direction, or factors `SCALEX`, `SCALEY`, `SCALEZ` for different scaling by direction. You may also translate the object in the three coordinate directions using the parameters `DX`, `DY`, `DZ`, each in meters. It is also possible to rotate, scale and translate an object using a 3×4 transformation matrix. The equation that describes the transformation for a point p with coordinates x, y, z

is:

$$\begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} = \begin{bmatrix} i_1 & j_1 & k_1 & a_1 \\ i_2 & j_2 & k_2 & a_2 \\ i_3 & j_3 & k_3 & a_3 \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix} \quad (11.3)$$

where the primes refer to the new location of point p . In that case the previous `MOVE` line is defined as:

```
&MOVE ID='My Move', T34 = i1,i2,i3, j1,j2,j3, k1,k2,k3, a1,a2,a3 /
```

where (i_1, i_2, i_3) , (j_1, j_2, j_3) and (k_1, k_2, k_3) are the orthogonal basis vectors for the rotation and scaling operations, and (a_1, a_2, a_3) is the translation vector. Note that if `T34` is defined, it takes precedence over all other transformation parameters.

Currently, the move feature only applies to point and line devices, and in future shall be applied to non-rectangular immersed objects.

Chapter 12

Chemical Species

FDS was designed primarily to study fire phenomena, and much of the basic chemistry of combustion is handled with a minimum of user inputs. However, there are many applications in which you might want to simulate the movement of gases in the absence of fire, or additional chemical species might be added to a simulation that involves fire. Gas species are defined with the input group `SPEC`. This input group is used to define both *primitive* gas species and *lumped* species (mixtures of one or more primitive `SPEC`).

There are different roles that a gas species might play in a simulation. A gas species might be explicitly tracked. In other words, a transport equation is solved for it. A gas species might be one component of a mixture of gases that are transported together. For example, FDS exploits the idea that the products of combustion from a fire mix and travel together; you only need to solve one transport equation for this “lumped species.” Or, a gas species might do both such as H_2O which could be both part of a lumped species for combustion products as well as an explicitly tracked species tracking H_2O from sprinkler droplet evaporation.

The default combustion model in FDS assumes that the reaction is mixing-controlled, and transport equations are solved for three species: the two lumped species of Products and Air (the default background) and the species Fuel (as defined on `REAC`). There is no reason to solve individual (and costly) transport equations for the major reactants and products of combustion—Fuel, O_2 , CO_2 , H_2O , N_2 , CO and soot—because they are all pre-tabulated functions of the three species. More detail on combustion is given in Chapter 13. For the moment, just realize that you need not, *and should not*, explicitly list the reactants and products of combustion using `SPEC` lines if all you want is to model a fire involving a hydrocarbon fuel.

12.1 Specifying Primitive Species

The `SPEC` namelist group is used to define a gas species in FDS. Once defined the species can be tracked as a single species (i.e., a primitive species) and/or the species can be used as part of one or more lumped species, also defined with `SPEC`. It is possible for a species to be both part of a lumped species and tracked separately. Often an extra gas introduced into a calculation is the same as a product of combustion, like water vapor from a sprinkler or carbon dioxide from an extinguisher. These gases are tracked separately. Thus, water vapor generated by the combustion is tracked via the `PRODUCTS` lumped species variable and water vapor generated by evaporating sprinkler droplets is tracked via its own transport equation.

If a species is only to be used as part of one or more lumped species, `LUMPED_COMPONENT_ONLY=T` must be added to the `SPEC` line. This tells FDS not to allocate space for the species in the array of tracked gases. If a species is to be used as the background species, the parameter `BACKGROUND=T` should be set on the `SPEC` line. A species with `LUMPED_COMPONENT_ONLY=T` cannot be used as an individual species, and it cannot be used as the background species. However, a primitive species with `BACKGROUND=T` can also be used as part

of a lumped species definition. Note that the default background species is `AIR` which is defined as a lumped species consisting of N_2 , O_2 , CO_2 , and H_2O . If no background species is defined in the input file, then FDS will create the background species of `AIR` by internally creating the input lines shown in Example 2 in Sec. 12.2.

Note that while you can define as many species using `SPEC` as you wish in an input file, any namelist input tied to a list of species, such as any `SPEC_ID` input or `MASS_FRACTION` input, is limited to using no more than 20 species.

12.1.1 Basics

Each `SPEC` line should include at the very least the name of the species via a character string, `ID`. Once the extra species has been declared, you introduce it at surfaces via the parameters `MASS_FRACTION(:)` or `MASS_FLUX(:)` along with the character array `SPEC_ID(:)`. A very simple example of how a gas can be introduced into the simulation is given by the simple input file called `gas_filling.fds`. The relevant lines are as follows:

```
&SPEC ID='HYDROGEN' /
&SURF ID='LEAK', SPEC_ID(1)='HYDROGEN', MASS_FLUX(1)=0.01667, RAMP_MF(1)='leak_ramp' /
&RAMP ID='leak_ramp', T= 0., F=0.0 /
&RAMP ID='leak_ramp', T= 1., F=1.0 /
&RAMP ID='leak_ramp', T=180., F=1.0 /
&RAMP ID='leak_ramp', T=181., F=0.0 /
&VENT XB=-0.6,0.4,-0.6,0.4,0.0,0.0, SURF_ID='LEAK', COLOR='RED' /
&DUMP MASS_FILE=T /
```

The hydrogen is injected through a 1 m by 1 m vent at a rate of $0.01667 \text{ kg}/(\text{m}^2 \text{ s})$ and shut off after 3 min. The total mass of hydrogen at that point ought to be 3 kg (see Fig. 12.1). Notice that no properties were needed for the `HYDROGEN` because it is a species whose properties are included in Appendix A. The background species in this case is assumed to be air. The mass flow rate of the hydrogen is controlled via the ramping parameter `RAMP_MF(1)`. The parameter `MASS_FILE=T` instructs FDS to produce an output file that contains a time history of the hydrogen mass.

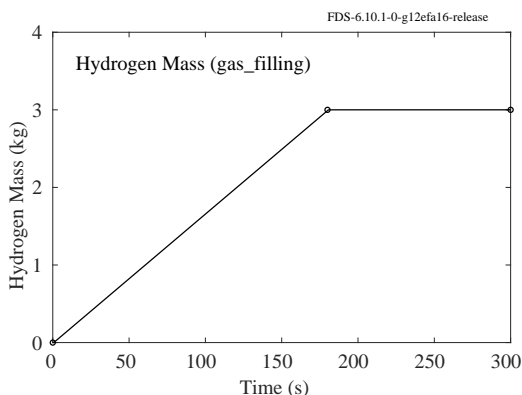


Figure 12.1: Hydrogen mass vs. time for `gas_filling` test case.

A species can be given an alternate ID using `ALT_ID`. When looking to see if species properties are predefined, FDS will only use `ID`, but when looking for a species specified in the input file with `SPEC_ID` for any input other than `SPEC`, FDS will use both `ID` and `ALT_ID`. One use of this is being able to use predefined

properties for species while also being able to use an abbreviated name or formula for complex chemistry reaction inputs.

Initial Conditions

If the initial mass fraction of the gas is something other than zero, then the parameter `MASS_FRACTION_0` is used to specify it. For example, if you want the initial concentration in the domain to be 90% background (air) diluted with 10% argon, use

```
&SPEC ID='ARGON', MASS_FRACTION_0=0.1 /
```

An alternative means of specifying the initial mass fraction is to use the `BACK` namelist with inputs of `SPEC_ID` and `MASS_FRACTION`. One use of this would be to modify or add initial species to a pre-existing set of detailed chemistry inputs referenced via `CATF`.

Specifying Humidity

If you are using the default background lumped species `AIR`, then you can specify `HUMIDITY` on `MISC` to set the ambient mass fraction of water vapor. `HUMIDITY` is the relative humidity of water vapor in units of %. It is 40 % by default.

If you are defining the primitive species of `WATER VAPOR`, then `MASS_FRACTION_0` is independent of `HUMIDITY`. That is setting `MASS_FRACTION_0` for the species `WATER VAPOR` will not change the ambient humidity, it will add additional water vapor.

12.1.2 Pre-Defined Gas and Liquid Properties

Gases and liquids whose properties are tabulated within FDS are listed in Appendix A. The physical properties of these species are known and do not need to be specified. When using one of these species you need only specify the correct `ID` and provide, if needed, the initial mass fraction. FDS will then use pre-compiled data to compute the various thermo-physical properties.

12.1.3 User-Defined Gas and Liquid Properties

If the gas species is not included in Appendix A, then you must specify its thermo-physical properties. By using the inputs discussed below, you can also override the default properties for a pre-defined gas species. For a gas species not included in Appendix A, its molecular weight, `MW`, should be specified on the `SPEC` line in units of g/mol, otherwise the molecular weight of nitrogen will be used. The molecular weight will be computed by FDS if a chemical formula is specified. If the species is participating in a reaction, and it is not the fuel species, then a reference enthalpy also be specified, see Sec. 12.1.3. The remaining thermo-physical properties of conductivity, diffusivity, enthalpy, viscosity, absorptivity (thermal radiation), and liquid properties are discussed below. Most properties can be defined as a constant value or as a temperature dependent look-up table using a `RAMP`. For the latter, if the temperature in a gas cell is above or below the endpoints of the `RAMP`, the endpoint value will be used. FDS will not extrapolate beyond the ends of the `RAMP`.

If conductivity, viscosity, or diffusivity are not specified they will be computed assuming the Lennard-Jones parameters for nitrogen and the specified molecular weight of the species. See Sec. 12.1.3 for the assumptions made if no specific heat or reference enthalpy are specified.

Note that when using LES, the molecular values of conductivity, viscosity, or diffusivity are not used in the transport calculation (see the FDS Technical Reference Guide [1] chapter titled Momentum Transport and Pressure). They are, however, used in correlations related to calculating heat and mass transfer coefficients.

Specifying a Chemical Formula

If you want FDS to compute the molecular weight of the gas species, you can input a `FORMULA` rather than the molecular weight, `MW`. This will also be used as the label for the gas species by Smokeview. `FORMULA` is a character string consisting of elements followed by their atom count. Subgroups bracketed by parentheses can also be given. The element name is given by its standard, case-sensitive, IUPAC¹ abbreviation (e.g., C for carbon, He for helium). The following are all equivalent:

```
&SPEC ID='ETHYLENE GLYCOL', FORMULA='C2H6O2' /
&SPEC ID='ETHYLENE GLYCOL', FORMULA='OHC2H4OH' /
&SPEC ID='ETHYLENE GLYCOL', FORMULA='C2H4(OH)2' /
```

Fuels compatible with the simple chemistry approach to defining reactions can use the inputs `C`, `H`, `O`, and `N`. Which specify the number of atoms of those elements in the fuel molecule. If these inputs are used, then `FORMULA` should not be specified and vice versa. The following would be equivalent to the prior examples:

```
&SPEC ID='ETHYLENE GLYCOL', C=2, H=6, O=2 /
```

Conductivity

Gas phase thermal conductivity, k , can be specified in one of four ways. First, it can be defined as a constant, `CONDUCTIVITY` ($W/(mK)$), on the `SPEC` line. Second, it can be defined as a function of temperature via the ramp, `RAMP_K`, on the `SPEC` line. Third, it can be computed by FDS using the parameters `MW` and `PR_GAS` given on the `SPEC` line (the default value `PR_GAS` is `PR` given on the `MISC` line). Fourth, it can be computed using the Lennard-Jones potential parameters σ (`SIGMALJ`) and ϵ/k (`EPSILONKLJ`) given on the `SPEC` line. If no inputs are specified, FDS will compute the conductivity using the `MW` and the Lennard-Jones parameters for nitrogen.

These methods of specifying the thermal conductivity are less important in LES simulations where the effective value is the sum of the molecular value, k , and a turbulent component, k_t :

$$k_{LES} = k + k_t \quad ; \quad k_t = \frac{c_p \mu_t}{Pr_t} \quad (12.1)$$

The turbulent viscosity, μ_t , is a function of the local flow field and grid size. The turbulent Prandtl number, Pr_t , is a specified constant. The specific heat, c_p , is a function of the gas temperature and composition for `SIMULATION_MODE='LES'`, and it is the constant value of the background gas species at ambient temperature for `'VLES'` and `'SVLES'`.

The gas phase output quantity `'CONDUCTIVITY'` denotes that which is actually used in the simulation, k_{LES} , while `'MOLECULAR CONDUCTIVITY'` is the molecular value, k , only; that is, the actual thermal conductivity of the gas with no turbulent component added.

¹International Union of Pure and Applied Chemistry

Diffusivity

Diffusivity is assumed to be the binary diffusion coefficient between the given species and the background species. Diffusivity can be specified in one of three ways: it can be defined as a constant using `DIFFUSIVITY` (m^2/s), it can be defined as a temperature vs. diffusivity ramp using `RAMP_D`, or it can be computed by FDS using `MW` and the Lennard-Jones potential parameters σ (`SIGMALJ`) and ϵ/k (`EPSILONKLJ`). If no inputs are specified, FDS will compute the diffusivity using the `MW` and the Lennard-Jones parameters for nitrogen.

Note that the turbulent diffusivity, $D_t = \mu_t/\text{Sc}_t$, is controlled by the turbulent Schmidt number, Sc_t , which is 0.5 by default and presumed the same for all tracked species. In some special cases, usually when the species is an aerosol such as soot, it may be valid to alter the turbulent Schmidt number independently from other species. To accommodate this, you may set the turbulent Schmidt number on the `SPEC` line for a primitive species (one with no subspecies). For example, the following lines set up tracking for primitive species with ethylene as the primary fuel allowing differential turbulent transport of soot (note that soot particles transport very slowly compared to gas species [14, 15]).

```
&SPEC ID='NITROGEN',          MASS_FRACTION_0=0.762470, BACKGROUND=T /
&SPEC ID='OXYGEN',           MASS_FRACTION_0=0.230997 /
&SPEC ID='CARBON DIOXIDE',    MASS_FRACTION_0=0.000591 /
&SPEC ID='WATER VAPOR',       MASS_FRACTION_0=0.005941 /
&SPEC ID='CARBON MONOXIDE',   MASS_FRACTION_0=0.000000 /
&SPEC ID='SOOT',              MASS_FRACTION_0=0.000000, TURBULENT_SCHMIDT_NUMBER=60. /
&SPEC ID='ETHYLENE',          MASS_FRACTION_0=0.000000 /
```

Enthalpy

The enthalpy of the gas mixture is given by the following formula:

$$h(T) = h(T_{\text{ref}}) + \int_{T_{\text{ref}}}^T c_p(T') dT' \quad (12.2)$$

where c_p is the `SPECIFIC_HEAT` ($\text{kJ}/(\text{kg K})$) with optional temperature dependence using `RAMP_CP`. The value of $h(T_{\text{ref}})$ can be specified in two ways. The first is to specify `REFERENCE_TEMPERATURE`, T_{ref} ($^{\circ}\text{C}$) and the value of the enthalpy at that temperature using `REFERENCE_ENTHALPY`, $h(T_{\text{ref}})$ (kJ/kg). The second is to specify the `ENTHALPY_OF_FORMATION` in kJ/mol . For this input T_{ref} is taken as the `H_F_REFERENCE_TEMPERATURE` on the `MISC` line. The default value of either reference temperature is 25°C .

If no input for $h(T_{\text{ref}})$ is provided, then FDS will assume that the enthalpy at 0 K is 0 kJ/kg , i.e. if a constant `SPECIFIC_HEAT` is given, then `REFERENCE_ENTHALPY` will be set to $h(T_{\text{ref}}) = c_p T_{\text{ref}}$. For `RAMP_CP`, the `RAMP` will be integrated from 0 K to T_{ref} .

If no input for specific specific heat is provided, then the specific heat of the gas will be calculated from its molecular weight using the relation:

$$c_{p,\alpha} = \frac{\gamma}{\gamma - 1} \frac{R}{W_{\alpha}} \quad (12.3)$$

The ratio of specific heats, `GAMMA`, is 1.4 by default and can be changed on the `MISC` line. If you want all the gas specific heats to follow this relation, set `CONSTANT_SPECIFIC_HEAT_RATIO=T` on the `MISC` line. For high molecular weight species, use of the default gamma will result in very low values of the specific heat which can cause issues with the default extinction model. In this case it is recommended that the specific heat be defined. If the FDS is determining the specific heat using this relation, then it is recommended that you check the `CHID.out` and verify that reasonable specific heat values have been created.

Note that species used in chemical reactions must either be predefined or have an explicitly defined $h(T_{\text{ref}})$. The exception is the species defined as `FUEL` on the `REAC` input as long as a `HEAT_OF_COMBUSTION` is

defined or the reaction is a simple chemistry reaction where EPUM02 applies. Specifying either SPECIFIC_HEAT or RAMP_CP is considered to have explicitly defined $h(T_{\text{ref}})$. If you still wish to use the default EPUM02 or 13,100 kJ/kg, then you will need to explicitly define it on the REAC input.

The thermodynamic properties (specific heat, enthalpy, entropy, and other derived properties) of a species can also be specified using NASA polynomials, as shown below. Two sets of POLYNOMIAL_COEFF are specified: the first set is valid for the temperature range of 200 K to 1000 K, and the second set is valid for the temperature range of 1000 K to 5000 K, as indicated by the POLYNOMIAL_TEMP parameter. In general, detailed mechanisms (see Section 13.3.2) specify species properties using NASA polynomials.

```
&SPEC ID='OH',
      PR_GAS= 0.763 ,
      FORMULA='H1.0O1.0',
      SIGMALJ= 2.75 ,
      EPSILONKLJ= 80.0 ,
      POLYNOMIAL='NASA7',
      POLYNOMIAL_TEMP= 200.0,1000.0,5000.0,
      POLYNOMIAL_COEFF(1:7,1)= 3.99,-0.002,4.61e-06,-3.88e-09,1.36e-12,3615.08,-0.104,
      POLYNOMIAL_COEFF(1:7,2)= 3.09,0.0005,1.26e-07,-8.79e-11,1.17e-14,3858.65,4.47,
      ENTHALPY_OF_FORMATION= 39.34692 /

&SPEC ID='NAOH',
      PR_GAS= 0.79 ,
      FORMULA='H1.0Na1.0O1.0',
      SIGMALJ= 3.804 ,
      EPSILONKLJ= 1962.0 ,
      POLYNOMIAL='NASA9',
      POLYNOMIAL_TEMP= 200.0,1000.0,5000.0,
      POLYNOMIAL_COEFF(1:9,1)=
        34420.36,-792.32,8.99,-0.004,3.06e-06,-5.11e-10,-1.54e-13,-20869.51,-25.10,
      POLYNOMIAL_COEFF(1:9,2)=
        875378.77,-2342.51,7.97,0.0001,-6.26e-08,1.02e-11,-5.71e-16,-9509.90,-22.02,
      ENTHALPY_OF_FORMATION= -190.99912 /
```

For NASA7 polynomials, the molar specific heat (J/mol/K), molar specific enthalpy (J/mol), and molar specific entropy (J/mol/K) are calculated as follows:

$$\begin{aligned}\frac{C_{p,\alpha}(T)}{R} &= a_1 + a_2T + a_3T^2 + a_4T^3 + a_5T^4 \\ \frac{H_{p,\alpha}(T)}{R} &= a_1T + \frac{a_2}{2}T^2 + \frac{a_3}{3}T^3 + \frac{a_4}{4}T^4 + \frac{a_5}{5}T^5 + a_6 \\ \frac{S_{p,\alpha}(T)}{R} &= a_1 \ln(T) + a_2T + \frac{a_3}{2}T^2 + \frac{a_4}{3}T^3 + \frac{a_5}{4}T^4 + a_7\end{aligned}\quad (12.4)$$

where $a_1, a_2, a_3, a_4, a_5, a_6, a_7$ are the polynomial coefficients specified by POLYNOMIAL_COEFF.

For NASA9 polynomials, the same values are calculated as follows:

$$\begin{aligned}\frac{C_{p,\alpha}(T)}{R} &= a_1T^{-2} + a_2T^{-1} + a_3 + a_4T + a_5T^2 + a_6T^3 + a_7T^4 \\ \frac{H_{p,\alpha}(T)}{R} &= -\frac{a_1}{T} + a_2 \ln(T) + a_3T + \frac{a_4}{2}T^2 + \frac{a_5}{3}T^3 + \frac{a_6}{4}T^4 + \frac{a_7}{5}T^5 + a_8 \\ \frac{S_{p,\alpha}(T)}{R} &= -\frac{a_1}{2T^2} - \frac{a_2}{T} + a_3 \ln(T) + a_4T + \frac{a_5}{2}T^2 + \frac{a_6}{3}T^3 + \frac{a_7}{4}T^4 + a_9\end{aligned}\quad (12.5)$$

where $a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8$, and a_9 are the polynomial coefficients specified by POLYNOMIAL_COEFF.

Also note that if a predefined species name is used and any enthalpy related input in this section is specified for that species, FDS will not use the predefined enthalpy of formation.

Viscosity

The dynamic viscosity of the gas species can be specified in one of three ways: it can be defined as a constant using `VISCOSITY`, it can be defined as a temperature vs. viscosity ramp using `RAMP_MU`, or it can be computed using the Lennard-Jones potential parameters σ (`SIGMALJ`) and ϵ/k (`EPSILONKLJ`). If no viscosity inputs are provided, FDS will use the Lennard-Jones values for nitrogen.

Radiative Properties

Species that can absorb and emit thermal radiation are defined via the parameter `RADCAL_ID` on the `SPEC` line. Some of the predefined species have this parameter already defined as shown in Appendix A. There are, however, many other species which are absorbing. For absorbing species not listed in Appendix A, `RADCAL_ID` can be used to identify a RadCal [6] species to serve as a surrogate. For example:

```
&SPEC ID='ETHANOL', RADCAL_ID='METHANOL' /
```

would use the RadCal absorptivities for `METHANOL` when computing the absorptivity of `ETHANOL`. For absorbing species not present in RadCal, it is recommended to choose a RadCal surrogate with similar molecular functional groups and molecular mass. The infrared spectrum is greatly affected by the species molecular functional groups.

Species molecular mass also affects the spectrum: a heavier species of a given molecular functional group tends to absorb and emit more infrared radiation than a lighter species of the same functional group. For simple chemistry, if the fuel is not present in Appendix A and no `FUEL_RADCAL_ID` is provided on the `REAC` line, then the absorption properties of methane will be used.

Gibbs Energy

If a reverse chemical reaction is specified using `REVERSE=T` on a `REAC` input, the equilibrium constant is determined from the Gibbs free energy. The Gibbs free energy (kJ/mol) as a function of temperature for a species can be specified with `RAMP_G_F` on the `SPEC` line. An example reverse reaction is provided in Sec. 13.3 on finite rate kinetics.

Liquids

The only instance where detailed liquid properties are needed is when defining a liquid droplet for a species not listed in Appendix A. The necessary properties are described in Sec. 15.3.1.

Prandtl Number

The Prandtl number for a gas species can be specified using `PR_GAS` on `SPEC`. If no value is given, the default value of `PR` on `MISC` will be used. If `PR_GAS` is defined for a known species, it will override the existing value.

12.1.4 Air

There are two predefined species for air in FDS. The first predefined species is the default background species of `AIR`. This is a lumped species consisting of oxygen, nitrogen, carbon dioxide, and water vapor whose mass fractions are controlled by the `Y_CO2_INFTY`, `Y_O2_INFTY`, and `HUMIDITY` inputs. This lumped

species is automatically defined by FDS if no other SPEC input is defined as the BACKGROUND. Note that ID='AIR' cannot be used on a SPEC input unless that input or some other SPEC input is defined as the BACKGROUND. The second predefined species is the primitive species of LJ AIR. This is an effective gas species whose molecular weight and enthalpy are defined based on the Y_CO2_INFTY and Y_O2_INFTY inputs, and whose other thermo-physical properties use the Lennard-Jones parameters for air. For simulations without combustion, using LJ AIR as the BACKGROUND species will slightly reduce the computational cost.

12.1.5 Two Gas Species with the Same Properties

In general only one species for a given ID can be defined; however, you may wish to model multiple inlet streams of a species and be able to identify how well the streams are mixing. This can be done by defining a new species with a single component. For example, the lines:

```
&SPEC ID='CARBON DIOXIDE', LUMPED_COMPONENT_ONLY=T/
&SPEC ID='CO2 1', SPEC_ID='CARBON DIOXIDE'/
&SPEC ID='CO2 2', SPEC_ID='CARBON DIOXIDE'/
&DEVC XYZ=..., QUANTITY='MASS FRACTION', SPEC_ID='CO2 1', ID='Device 1'/
&DEVC XYZ=..., QUANTITY='MASS FRACTION', SPEC_ID='CO2 2', ID='Device 2'/'
```

define two duplicate species, both of which are CARBON DIOXIDE. Both will use the built in property data for CO₂ (note specifying one or more properties for a duplicate species will override the default properties). The ID for each duplicate species can then be used in the remainder of the input file. In this example, the LUMPED_COMPONENT_ONLY was given so that FDS only tracks CO2 1 and CO2 2 and not CARBON DIOXIDE. Note that the ID you provide for a duplicate species cannot match the ID of any other primitive or lumped SPEC input. Also note that this feature can only be used to duplicate a predefined species (i.e. a species listed in Appendix A).

In the above example, the two DEVC lines refer to the IDs of the duplicate species. If instead the primitive species ID was used, then the output would sum the mass fractions over all species containing that primitive species. For example, if the output quantities in the example above are changed as shown below, then Device 1 would just output the mass fraction of CO2 1 and Device 2 would output the sum of both.

```
&DEVC XYZ=..., QUANTITY='MASS FRACTION', SPEC_ID='CO2 1', ID='Device 1'/
&DEVC XYZ=..., QUANTITY='MASS FRACTION', SPEC_ID='CARBON DIOXIDE', ID='Device 2'/'
```

Another application of this would be if you wanted to track the water that evaporated from sprinklers, separately from the water that resulted from combustion. The following inputs would allow you to do that:

```
&REAC FUEL='PROPANE', ... /
&SPEC ID='WATER VAPOR SPK', SPEC_ID='WATER VAPOR' /
&PART ID='Sprinkler Droplets', SPEC_ID='WATER VAPOR SPK'/
&DEVC XYZ=..., QUANTITY='MASS FRACTION', SPEC_ID='WATER VAPOR SPK', ID='Spr H2O'/
&DEVC XYZ=..., QUANTITY='MASS FRACTION', SPEC_ID='WATER VAPOR', ID='All H2O'/'
```

The gas species called WATER VAPOR SPK has the same properties as WATER VAPOR. The first device records only water that results from droplet evaporation, and the second device records water that originates from both sprinklers and combustion.

By default, duplicate species are considered to be a lumped species and not a primitive species. That is, by default, a duplicate species cannot be used in a lumped species definition. Setting PRIMITIVE=T will have FDS treat the duplicate species as a primitive species and allow it to be used in a lumped species definition. Note that when this is done, one can no longer aggregate SPEC_ID based outputs over the original and the duplicate species as the aggregation is done on the basis of the primitive species names. This is

demonstrated in the example below. The species O2 LUMPED and O2 DUPLICATE PRIMITIVE are duplicate species of OXYGEN. O2 DUPLICATE PRIMITIVE is defined as a primitive species and O2 LUMPED is considered only a lumped species. The initial DEVC outputs in order would be 0.3 (0.1 for OXYGEN plus 0.2 for O2 LUMPED), 0.2 (the O2 LUMPED initial value), and 0.3 (the O2 DUPLICATE PRIMITIVE initial value). Note the OXYGEN output is not 0.6 since the O2 DUPLICATE PRIMITIVE is defined as a new primitive species.

```
&SPEC ID='NITROGEN',BACKGROUND=T/
&SPEC ID='OXYGEN',MASS_FRACTION_0=0.1/
&SPEC ID='O2 LUMPED',SPEC_ID='OXYGEN',MASS_FRACTION_0=0.2/
&SPEC ID='O2 DUPLICATE PRIMITIVE',SPEC_ID='OXYGEN',MASS_FRACTION_0=0.3,PRIMITIVE=T/

&DEVC XYZ=0.5,0.5,0.5,QUANTITY='MASS FRACTION',SPEC_ID='OXYGEN'/
&DEVC XYZ=0.5,0.5,0.5,QUANTITY='MASS FRACTION',SPEC_ID='O2 LUMPED'/
&DEVC XYZ=0.5,0.5,0.5,QUANTITY='MASS FRACTION',SPEC_ID='O2 DUPLICATE PRIMITIVE'/
```

12.1.6 Soot

The predefined species of SOOT' is defined with with FORMULA='C'. To specify soot that contains other atoms, use FORMULA. If new thermo-physical properties are not specified, define the FORMULA to keep the molecular weight near 12 g/mol, i.e FORMULA='C0.86H1.7' rather than FORMULA='C10H20'.

12.2 Specifying Lumped Species (Mixtures of Primitive Species)

The `SPEC` namelist group also allows you to define species mixtures. The purpose of a species mixture is to reduce the number of species transport equations that are explicitly solved. Air, for example, is composed of nitrogen, oxygen, water vapor, and carbon dioxide. If we define the four component species of air, we will have four total species. Alternatively, we can define a “lumped species” that represents the air mixture, which saves on computational time because we need only solve one transport equation. A lumped species is a group of species that transport and react together in the same proportion. This implies the molecular diffusivities of each component of the mixture are the same, which is an approximation².

The following inputs define equivalent initial mixtures. But in the first example three species are explicitly tracked and in the second example only a background mixture is specified. Note that this example represents the background species created by FDS when no background is explicitly defined in the input file. It is also noted that any implicitly defined species (such as the nitrogen, oxygen, water vapor, and carbon dioxide for the air background species or the species in the lumped product species for simple chemistry, see 13.1.1), can be referenced by any of the outputs such as `DEVC` or `SLCF`.

Example 1: All primitive species

```
&SPEC ID='NITROGEN', BACKGROUND=T / Note: The background must be defined first.
&SPEC ID='OXYGEN',      MASS_FRACTION_0=0.23054 /
&SPEC ID='WATER VAPOR',  MASS_FRACTION_0=0.00626 /
&SPEC ID='CARBON DIOXIDE', MASS_FRACTION_0=0.00046 /
```

Example 2: Defining a background species

```
&SPEC ID='NITROGEN',      LUMPED_COMPONENT_ONLY=T /
&SPEC ID='OXYGEN',        LUMPED_COMPONENT_ONLY=T /
&SPEC ID='WATER VAPOR',    LUMPED_COMPONENT_ONLY=T /
&SPEC ID='CARBON DIOXIDE', LUMPED_COMPONENT_ONLY=T /

&SPEC ID='AIR', BACKGROUND=T,
      SPEC_ID(1)='NITROGEN',      MASS_FRACTION(1)=0.76274,
      SPEC_ID(2)='OXYGEN',        MASS_FRACTION(2)=0.23054,
      SPEC_ID(3)='WATER VAPOR',    MASS_FRACTION(3)=0.00626,
      SPEC_ID(4)='CARBON DIOXIDE', MASS_FRACTION(4)=0.00046 /
```

The logical parameter `LUMPED_COMPONENT_ONLY` indicates that the species is only present as part of a lumped species. When `T`, FDS will not allocate space to track that species individually. The parameters to define a lumped species are:

- `BACKGROUND` Denotes that this lumped species is to be used as the background species.
- `ID` Character string identifying the name of the species. You must provide this. This cannot be the same as an `ID` of another `SPEC` input.
- `SPEC_ID` Character array containing the names of the primitive species that make up the lumped species.

²Turbulent diffusion usually dominates by a couple orders of magnitude making equal diffusivities a good approximation.

- `MASS_FRACTION` The mass fractions of the components of the lumped species in the order listed by `SPEC_ID`. FDS will normalize the values to 1. Alternatively, `VOLUME_FRACTION` can be specified. Do not use both on an `SPEC` line.
- `MASS_FRACTION_0` The initial mass fractions of lumped species.

When defining a lumped species, either `MASS_FRACTION` or `VOLUME_FRACTION` must be used to define the component species. The addition of lumped species to FDS has changed the meaning of `SPEC_ID` on some FDS inputs. For `INIT`, `MATL`, `PART`, and `SURF`, `SPEC_ID` refers to either a tracked primitive species or a lumped species. For outputs and devices, `DEVC`, that require a `SPEC_ID`, the `SPEC_ID` input can refer to either a tracked primitive species, a lumped species, or a lumped species component that is not tracked. For `REAC` see the discussion on specifying reactions in Chapter 13.

It is possible to create a separately tracked copy of a lumped species by using `COPY_LUMPED`. For example, if in Example 2 above one added the line:

```
&SPEC ID='AIR2', COPY_LUMPED=T, SPEC_ID='AIR' /
```

then FDS would create a copy of the `AIR` lumped species called `AIR2`.

12.2.1 Combining Lumped and Primitive Species

There are cases where you may wish to have a single primitive species be both part of a lumped species and also a separately tracked species. For example, when using simple chemistry, Sec. 13.1.1, FDS will include water vapor in the product species and in the air background species. If you also wish to have sprinklers in the simulation, then you will need to track water vapor from the sprinklers separately from that in the air or products. This is simply done by adding the line:

```
&SPEC ID='WATER VAPOR' /
```

This will override the implicitly created water vapor species defined with `LUMPED_COMPONENT_ONLY=T` and cause FDS to track water vapor as a separate species. Note that in this case if you requested an output for the mass fraction of `WATER VAPOR` you would get the water vapor in the air and product lumped species as well as that which evaporated from sprinkler droplets. If you wanted in this case (where water vapor is implicitly defined), to be able to track the water vapor from sprinklers separately you could follow the example in Sec. 12.1.5 and define:

```
&SPEC ID='SPRINKLER WATER VAPOR', SPEC_ID='WATER VAPOR' /
```

Using the species `SPRINKLER WATER VAPOR` for the sprinklers would allow you to track sprinkler generated water vapor separately.

Chapter 13

Combustion

Combustion can be modeled in two ways. By default, the reaction of fuel and oxygen is infinitely fast and controlled only by mixing with zero ignition delay—you will often see this model described as “mixed is burnt”. The alternative is that the reaction is *finite-rate*. The latter approach usually requires very fine grid resolution that is not practical for large-scale fire applications. This chapter describes both methods, with an emphasis on the more commonly used mixing-controlled model.

There are two groups of parameters that govern combustion. The first, called the `COMB` namelist group, contains parameters that pertain to any and all reactions. Specific parameters about a particular reaction are specified using the `REAC` namelist group. There can only be one `COMB` line, but multiple `REAC` lines if there are multiple reactions. If you are modeling a fire, you *must* specify the fuel and basic stoichiometry using a `REAC` line. You need not specify a `COMB` line unless you want to modify mainly numerical parameters.

13.1 Single-Step, Mixing-Controlled Combustion

This approach to combustion, referred to below as the “simple chemistry” combustion model, considers a single fuel species that is composed primarily of C, H, O, and N that reacts with oxygen in one mixing-controlled step to form H₂O, CO₂, soot, and CO. Information about the reaction is provided on the `REAC` line. Starting with FDS 6, you *must* specify a `REAC` line to model a fire. You are responsible for defining the basic fuel chemistry and the post-combustion yields of CO and soot (the default yields are 0). More than one simple chemistry reaction can be defined; however, each simple chemistry reaction must have a unique fuel.

13.1.1 Simple Chemistry Parameters

For the simple chemistry model, each reaction is assumed to be of the form:



You need only specify the chemical formula of the fuel along with the yields of CO and soot, and the volume fraction of hydrogen in the soot, X_H . FDS will use that information and calculate the stoichiometric coefficients automatically as follows:

$$\begin{aligned} v_{O_2} &= v_{CO_2} + \frac{v_{CO}}{2} + \frac{v_{H_2O}}{2} - \frac{z}{2} \\ v_{CO_2} &= x - v_{CO} - v_{HCN} - (1 - X_H) v_S \\ v_{H_2O} &= \frac{y}{2} - \frac{X_H}{2} v_S - \frac{v_{HCN}}{2} \end{aligned}$$

$$\begin{aligned}
v_{\text{CO}} &= \frac{W_F}{W_{\text{CO}}} y_{\text{CO}} \\
v_s &= \frac{W_F}{W_s} y_s \\
v_{\text{HCN}} &= \frac{W_F}{W_{\text{HCN}}} y_{\text{HCN}} \\
v_{\text{N}_2} &= \frac{v}{2} - \frac{v_{\text{HCN}}}{2} \\
W_s &= X_H W_H + (1 - X_H) W_C
\end{aligned}$$

The following parameters may be prescribed on the `REAC` line when using the simple chemistry model. Note that the various `YIELDS` are for well-ventilated, post-flame conditions. There are options to predict various species yields in under-ventilated fire scenarios, but these special models still require the post-flame yields for CO, soot and any other species listed below.

- **FUEL** (Required) A character string that identifies fuel species for the reaction. The `FUEL` species must be either a predefined species or defined on a `SPEC` input. Appendix A provides a listing of the available predefined species. If the `FUEL` is in the table, then FDS will use the predefined thermo-physical properties and chemical formula for the fuel. See Sec. 12.1 for details on specifying a user-defined species.
- **ID** A character string that identifies the reaction. Normally, this label is not used by FDS, but it is useful to label the `REAC` line if more than one reactions are specified.
- **CO_YIELD** The fraction of fuel mass converted into carbon monoxide, y_{CO} . Note that this parameter is only appropriate when the simple chemistry model is applied. (Default 0.)
- **SOOT_YIELD** The fraction of fuel mass converted into smoke particulate, y_s . Note that this parameter is only appropriate when the simple chemistry model is applied. (Default 0.)
- **HCN_YIELD** The fraction of fuel mass converted into hydrogen cyanide, y_{HCN} . Note that this parameter is only appropriate when the simple chemistry model is applied and the fuel contains nitrogen. (Default 0.)
- **FUEL_RADCAL_ID** RadCal species to be used for the fuel. The default is the default RadCal species for the fuel species or 'METHANE' if there is no species default. See Sec. 14.3.1 for details.

The ambient mass fractions for the constituents of air are specified on `MISC` using the inputs:

- **Y_O2_INF** Ambient mass fraction of oxygen (Default for dry air is 0.232378)
- **Y_CO2_INF** Ambient mass fraction of carbon dioxide (Default for dry air is 0.000595)
- **HUMIDITY** Relative humidity of the background air species, in units of %. (Default 40 %).

A few sample `REAC` lines are given here.

```
&REAC FUEL = 'METHANE' /
```

In this case, there is no need for a `FORMULA` or atom count because the `FUEL` is listed in Appendix A. It is assumed that the soot and CO yields are zero. FDS will compute the yields of product species and the heat of combustion based upon predefined values.

```
&REAC FUEL          = 'PROPANE'
      SOOT_YIELD     = 0.01
      CO_YIELD       = 0.02
      HEAT_OF_COMBUSTION = 46460. /
```

In this case, the fuel species is again predefined. However, here the heat of combustion is specified explicitly rather than calculated. Additionally, minor species yields have been specified with the soot yield specified as 0.01 and the CO yield specified as 0.02. See Sec. 13.1.2 for more details on the heat of combustion.

```
&SPEC ID            = 'MY FUEL'
      FORMULA        = 'C3H8O3N4' /

&REAC FUEL          = 'MY FUEL'
      HEAT_OF_COMBUSTION = 46124. /
```

In this case, the fuel is not predefined. Therefore, either the `FORMULA` or the atom counts must be defined on `SPEC`. This input defined the `FORMULA`. In this case, the heat of combustion is known and specified; however, if it weren't FDS would compute it using `EPUMO2` and the fuel chemistry. Note that simple chemistry can also be used for cases where the fuel is a lumped species so long as the defining primitive species contain only C, H, N, and O atoms. An example can be found in Section 13.2.2.

When simple chemistry is being used, FDS will automatically create three lumped species: `AIR`, `FUEL`, and `PRODUCTS`. The actual name of the fuel species will be the name given on the `REAC` line (for example `MY FUEL` in the last sample above). FDS creates these lumped species in the same manner as you would in an input file. FDS first defines the primitive species and then defines the lumped species. In essence FDS internally creates input lines like those shown in Example 2 of Sec. 12.2. This means when doing simple chemistry, that even though you did not explicitly define oxygen in the input file, you can request an output for oxygen since it was implicitly defined by FDS.

Multiple simple chemistry reactions can be defined. Each new reaction will require that FDS track a new fuel and product species mixture.

13.1.2 Heat of Combustion

In FDS, the energy release per unit volume (kJ/m^3) from a gas phase chemical reaction (or system of reactions) is found by taking the sum of the net change in density for each species in that volume in a given time step multiplied by the respective species' enthalpy of formation (kJ/kg). In this formulation, the enthalpy of formation for all participating species needs to be specified. If a reaction (simple chemistry or user-defined) contains only species defined in Appendix A, then all of the enthalpies of formation are known. These values can be found in the FDS source code in `data.f90`. For reactions with species that are not included in Appendix A, there are several options to ensure that all of the enthalpies of formation are specified.

Option 1: Specify Enthalpy of Formation

You can specify unknown enthalpies on the `SPEC` line in units of kJ/mol :

```
&SPEC ID = 'GLUCOSE', FORMULA = 'C6H12O6', ENTHALPY_OF_FORMATION=-1.297E3 /
```

Note FDS will also use `REFERENCE_ENTHALPY`, `SPECIFIC_HEAT`, or `RAMP_CP` to define the enthalpy of formation as discussed in Section 12.1.3.

Option 2: Specify Heat of Combustion

For a given reaction, if the only species missing an enthalpy of formation is the fuel, the missing value can be found if the `HEAT_OF_COMBUSTION` in kJ/kg is specified on `REAC`. Sec. 13.2.2 provides an example for which the fuel, polyvinyl chloride, has an unspecified enthalpy of formation but a specified heat of combustion on the `REAC` line. Specifying the `HEAT_OF_COMBUSTION` will result in FDS recomputing any fuel species enthalpy of formation based on `SPEC` inputs. If this occurs for the same `FUEL` in multiple reactions a warning message will be written.

Option 3: Specify Heat of Combustion Based on Oxygen Consumption (Simple Chemistry)

If the enthalpy of formation of the fuel and heat of combustion are not specified, for simple chemistry cases only, the heat of combustion is assumed to be

$$\Delta h \approx \frac{v_{O_2} W_{O_2}}{v_F W_F} \text{ EPUMO2} \quad \text{kJ/kg} \quad (13.2)$$

The quantity `EPUMO2` (kJ/kg) is the amount of energy released per unit mass of oxygen consumed. Its default is 13,100 kJ/kg. Typically, a chemical reaction is balanced by setting the stoichiometric coefficient of the fuel v_F to 1. In FDS, the stoichiometric coefficients of the chemical reaction are normalized by the stoichiometric coefficient of the fuel, effectively setting v_F to 1. Note that if both `EPUMO2` and `HEAT_OF_COMBUSTION` are specified that FDS will ignore the value for `EPUMO2`. Note that the assumed value of `EPUMO2` will not be used if Option 1 has been used. If you wish to keep using the default value, add `EPUMO2=13100` to the `REAC` line.

If heats of reaction have been specified on the `MATL` lines and the heats of combustion of the materials differ from that specified by the governing gas phase reaction, then add a `HEAT_OF_COMBUSTION` (kJ/kg) to the `MATL` line. In a realistic fire scenarios, there may be many fuel gases generated by the various burning objects in the building. Specify the stoichiometry of the predominant reaction via the `REAC` namelist group. If the stoichiometry of the burning material differs from the global reaction, the `HEAT_OF_COMBUSTION` is used to ensure that an equivalent amount of fuel is injected into the flow domain from the burning object.

Values of the heat of combustion tabulated for pure fuels (e.g., one species) in handbooks are often computed using the enthalpies of formation for the the fuel and products assuming complete combustion. This ideal heat of combustion does not account for the `SOOT_YIELD`, `CO_YIELD`, or `HCN_YIELD` that occurs in a real fire. Carbon and hydrogen that go to soot, CO, and HCN rather than CO_2 and H_2O result in a lower effective heat of combustion. Setting `IDEAL=T` will reduce the `HEAT_OF_COMBUSTION` based upon the inputs for `SOOT_YIELD`, `CO_YIELD`, and `HCN_YIELD`.

Another approach to determining the heat of combustion is to burn a known mass of the material in a calorimeter and divide the heat release rate by the mass loss rate (known as the effective heat of combustion). In this approach, represented by `IDEAL=F`, the measured value of the heat release rate includes the effects of any soot, CO, or HCN that is produced and no adjustment is needed. The default value is `IDEAL=F`. Note that predefined fuel species have their heat of formation defined; therefore, the heat of combustion for those species will be appropriately adjusted for soot and CO production.

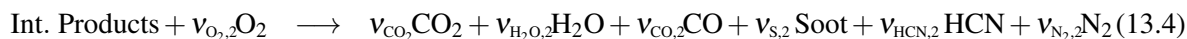
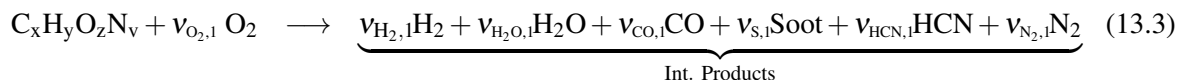
If `EPUMO2` is specified instead of `HEAT_OF_COMBUSTION`, then the `EPUMO2` will not be changed, (i.e., FDS will always treat `EPUMO2` as `IDEAL=F`).

If the reaction is under defined, FDS will return an error at the start of the calculation.

13.1.3 Two-Step Simple Chemistry

The default simple chemistry single-step combustion model has a two-step option. This option is invoked by setting `N_SIMPLE_CHEMISTRY_REACTIONS` to 2 on the `REAC` line for each simple chemistry reaction where two-steps are desired. All of the other parameters that are appropriate for the default single-step model are

still applicable. The two-step scheme basically takes all of the carbon in the fuel molecule and converts it to CO and Soot in the first step, and then oxidizes most of the CO and Soot to form CO₂ in the second step. The hydrogen in the fuel molecule can form either H₂ or H₂O in the first step as well.

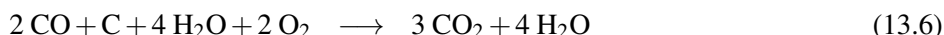
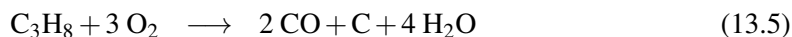


By default, in the first step, two out of three carbon atoms in the fuel are converted to CO. There is not yet a solid basis for this assumption, and the distribution of carbon to CO and Soot can be changed via the parameter `FUEL_C_TO_CO_FRACTION` on the `REAC` line. It is 2/3, by default. Also by default one out of five nitrogen atoms in the fuel are converted to HCN. This value is based on testing of nitrogen containing fuels over a range of equivalence ratios [16] and can be changed via the parameter `FUEL_N_TO_HCN_FRACTION`. In addition, a fraction of the hydrogen in the fuel molecule can form H₂ in the first step. The controlling parameter is called `FUEL_H_TO_H2_FRACTION` and it is zero by default because this chemistry is not well understood and has been added to the two-step scheme as a placeholder for future research.

Note that the parameters `CO_YIELD`, `SOOT_YIELD`, and `HCN_YIELD` retain their meanings from the single-step simple chemistry model; that is, they represent the *post-flame* yields of these species. Essentially, the two-step model acknowledges the fact that CO and Soot are present at much higher concentrations within the flame envelop than their post-flame yields would suggest.

The two-step simple chemistry option should only be invoked when you have an interest in near-flame phenomena where the increased concentration of CO and Soot play an important role in the flame chemistry and radiative emission. The resolution of the fire should be reasonably good, as well. What “reasonably good” means depends on the particular circumstances, but suffice it to say that you ought to experiment by running simple simulations with and without the two-step option to see if it leads to significantly different results. The cost of the two-step scheme is an additional transport equation for the scalar variable referred to as “Intermediate Products.”

A simple demonstration of two-step simple chemistry is given by the example cases in the `Species` folder: `propane_flame_2reac.fds` and `propane_flame_2reac_simple.fds`. Both cases employ the same two-step reaction scheme:



The simple version of the input file specifies the combustion parameters as follows:

```
&REAC FUEL='PROPANE', SOOT_YIELD=0., CO_YIELD=0., N_SIMPLE_CHEMISTRY_REACTIONS=2,
      FUEL_C_TO_CO_FRACTION=0.6667 /
```

For simplicity in setting up the complex form of the input file, the post-flame Soot (C) and CO yields are set to zero. Within the flame envelop, Soot and CO are to be generated following the specified `FUEL_C_TO_CO_FRACTION` by which 2 of the 3 carbon atoms in the fuel molecule make up CO, and 1 carbon atom forms. To check that the two formulations are the same, Fig. 13.1 displays the heat release and radiative heat release rates for the first few seconds of simulation.

A more complex demonstration of the two-step simple chemistry is given by the `multiple_reac_n_simple.fds` case also located in the `Species` folder. This case contains three reactions, two of which have two-step simple chemistry (with different values for `FUEL_C_TO_CO_FRACTION` and `FUEL_N_TO_HCN_FRACTION`), and the third which is simple chemistry only (with a `CO_YIELD` of 0.3). In the

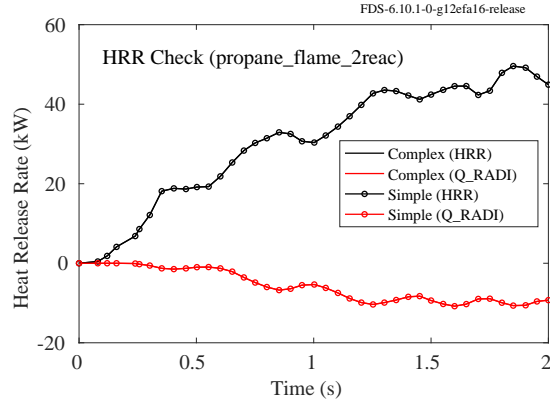


Figure 13.1: Demonstration that the simplified form of the two-step simple chemistry input parameters are equivalent to a more complex form.

case, three boxes are each initialized with a small amount of the fuel from one reaction and insufficient oxygen to complete the first (or only) step. The results in Fig. 13.2 shows that FDS correctly performs each reaction as one or two steps with the appropriate yields.

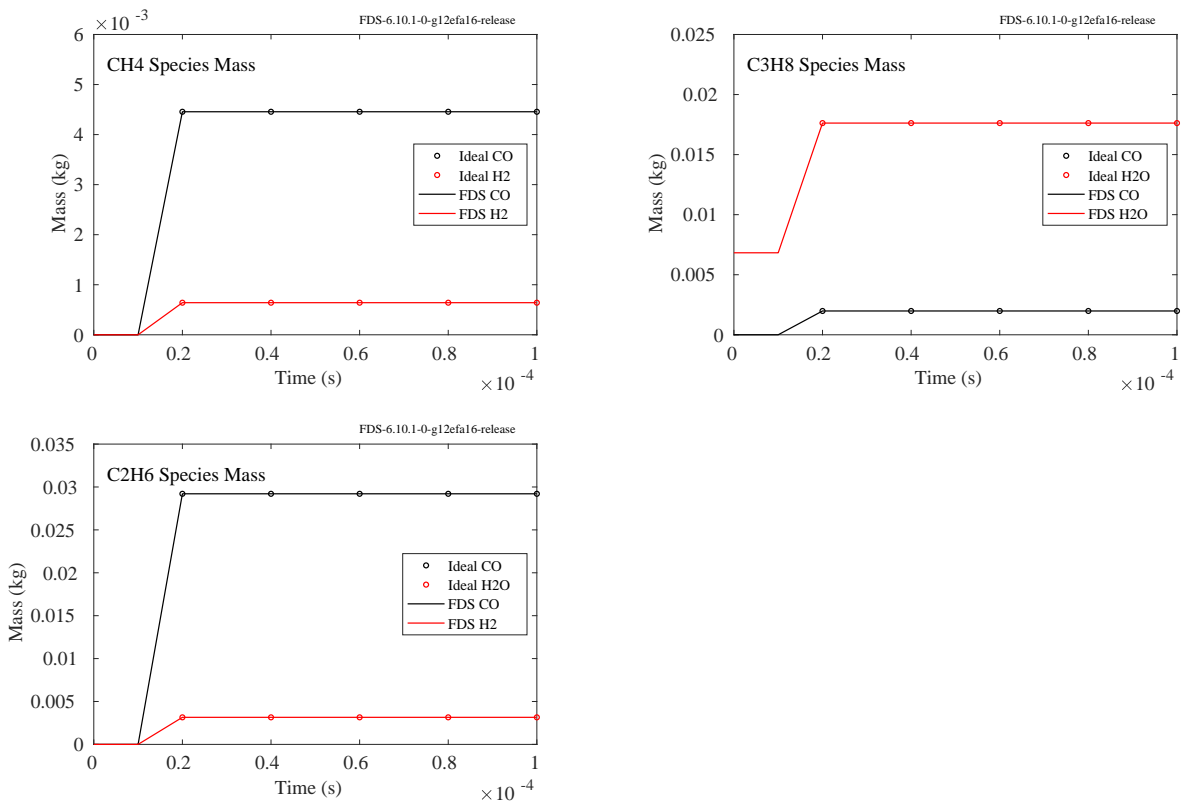


Figure 13.2: Results of the `multiple_reac_n_simple` test case; demonstration that FDS correctly assigns one or two step reactions with the correct yields. From top left cases are methane with two-step chemistry and `FUEL_X_TO_Y_FRACTIONS` of 0.2, propane with single-step chemistry and a `CO_YIELD` of 0.3, and ethane with two-step chemistry and fractions of 0.8.

13.1.4 Complete Heat of Combustion

As discussed in Sec. 9.2 and Sec. 15.3.1, mass loss of fuel from solids or droplets is adjusted in the gas phase so that the correct total heat release rate is obtained. If a multiple-step reaction scheme, such as discussed in the prior section, is being used, then the `HEAT_OF_COMBUSTION` for the `REAC` line of the first step does is not the complete heat of combustion for that fuel. Using this value to adjust mass loss rates would result in an incorrect heat release rate. If a multi-step reaction is being defined and the fuel is being produced by pyrolysis or evaporation, the parameter `HOC_COMPLETE` should be set on the `REAC` line for the first step. This value will be used for adjusting mass loss rate instead of the `HEAT_OF_COMBUSTION`. Note that if two-step simple chemistry is being used, then FDS will automatically compute the value of `HOC_COMPLETE`.

13.1.5 Turbulent Combustion

Unless you are performing a Direct Numerical Simulation (DNS), the reaction rate of fuel and oxygen is not based on the diffusion of fuel and oxygen at a well-resolved flame sheet. Instead, semi-empirical rules are invoked by FDS to determine the rate of mixing of fuel and oxygen within a given mesh cell at a given time step. Each computational cell can be thought of as a batch reactor where only the mixed composition can react. The variable, $\zeta(t)$, denotes the unmixed fraction, ranging from zero to one and governed by the equation:

$$\frac{d\zeta}{dt} = \frac{-\zeta}{\tau_{\text{mix}}} \quad (13.7)$$

Here, τ_{mix} is the mixing time scale. The change in mass of a species is found from the combination of mixing and the production/destruction rate found from the chemical reaction. If a cell is initially unmixed, $\zeta_0 = 1$ by default, then combustion is considered non-premixed within the grid cell. In this case, the fuel and air are considered completely separate at the start of the time step and must mix together before they burn. This means if the mixing is slow enough, that unburned fuel may exist at the end of the time step even if sufficient oxygen is present in the grid cell to burn all the fuel. If the cell is initially fully mixed, $\zeta_0 = 0$, then the combustion is considered premixed (e.g., equivalent to infinitely fast mixing). You can set the amount of mixing in each cell at the beginning of every time step using the parameter `INITIAL_UNMIXED_FRACTION` on the `COMB` line.

The mixing time, τ_{mix} , is a function of the the level of turbulence in the neighborhood of the point of interest. However, you may override the calculation of τ_{mix} by setting a `FIXED_MIX_TIME` (s) on the `COMB` line. Alternatively, you can bound the computed value of τ_{mix} by setting a lower bound `TAU_CHEM` and/or an upper bound `TAU_FLAME` on the `COMB` line.

The Technical Reference Guide [3] contains more detailed information about the turbulent combustion model.

13.1.6 Flame Extinction

Modeling suppression of a fire due to the introduction of a suppression agent like CO_2 or water mist, or due to the exhaustion of oxygen within a closed compartment is challenging because the relevant physical mechanisms typically occur at subgrid-scale. Flames are extinguished due to lowered temperatures and dilution of the fuel or oxygen supply. There are two flame extinction models in FDS that determine whether or not combustion is viable based on the cell temperature and the oxygen and fuel concentrations. In brief, for combustion to occur there must be sufficient oxygen and fuel to raise the cell temperature from its current value to a *critical flame temperature* (CFT). The CFT is based on the *oxygen index* (OI) concept discussed in Beyler's chapter in the SFPE Handbook [17]. The oxygen index is the volume fraction of oxygen in the

oxidizer stream when extinguishment occurs. The adiabatic flame temperature¹ of a stoichiometric mixture of fuel and oxygen at this limiting oxygen concentration, T_{OI} , is taken as the CFT. Values for the CFT (T_{OI}) are listed in Table 13.1. Both extinction models make use of the CFT.

The `EXTINCTION_MODEL` is specified on the `COMB` line². '`EXTINCTION 1`' is the default for Very Large Eddy Simulation (VLES) and for Simple Very Large Eddy Simulation (SVLES) mode, whereas '`EXTINCTION 2`' is used for DNS and LES. The difference between the two models is that for '`EXTINCTION 1`', only the cell temperature and oxygen concentration are considered because detailed thermo-physical gas species properties are not invoked, as they are in the '`EXTINCTION 2`' model. Generally speaking, '`EXTINCTION 2`' requires a relatively fine grid because it uses the bulk cell temperature in its calculation of the critical flame temperature.

Extinction Model 1

The '`EXTINCTION 1`' model is summarized by the left hand plot of Fig. 13.3, which indicates the ranges of cell temperature and oxygen concentration where combustion is viable. This model is meant to be used for simulations where the grid cells are relatively large and flames cannot be resolved. In particular, this means that above approximately 600 °C, no flame extinction is assumed unless the oxygen concentration drops to zero. There are a few parameters associated with this model. First, the `LOWER_OXYGEN_LIMIT`, which is sometimes referred to as the *lower oxygen index*, is the oxygen volume fraction at the left end of the solid line in Fig. 13.3. Values for various fuels are given in Table 13.1. Second, above the `FREE_BURN_TEMPERATURE`, whose default value is 600 °C, combustion is not suppressed. The default value is a typical indicator of flashover. Finally, the `CRITICAL_FLAME_TEMPERATURE` establishes the point of intersection of the sloped line and the x axis. Values for various fuels are listed in Table 13.1, but given the simplicity of this model, the default value is recommended.

Extinction Model 2

The '`EXTINCTION 2`' model is summarized by the right hand plot of Fig. 13.3. For this model, the `CRITICAL_FLAME_TEMPERATURE` plays a larger role because there must be sufficient fuel and oxygen within a cell to raise its temperature beyond the CFT. The calculations of species component enthalpies is more exact with this model as well. For unlisted fuels, you can set the CFT on the `REAC` line or simply accept the default value, 1427 °C (1700 K). If you know the oxygen index X_{OI} , for a particular fuel, the CFT be calculated from the following equation:

$$T_{OI} = T_0 + X_{OI} \frac{\Delta H_c / r}{n \bar{c}_p} \quad (13.8)$$

where

T_0	Initial temperature of the fuel/air mixture (ambient conditions) (K)
X_{OI}	Limiting oxygen volume fraction
$\Delta H_c / r$	Heat of combustion per mole of oxygen consumed (kJ/mol)
n	Number of moles of products of combustion per mole of fuel/air mixture
\bar{c}_p	Average heat capacity (kJ/(mol·K)) of products of combustion in the range T_0 to T_{OI}

¹If you want to know the actual adiabatic flame temperature for each of your specified reactions, add `COMPUTE_ADIABATIC_FLAME_TEMPERATURE=T` to the `COMB` line. The result is written to the file called `CHID.out`.

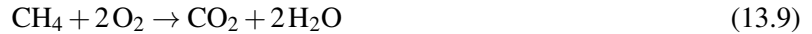
²To eliminate any gas phase suppression, set `SUPPRESSION=F` on the `COMB` line.

Table 13.1: Default Critical Flame Temperatures for common fuels, from Table 17.3 of Ref. [17].

Fuel	Formula	X_{OI}	T_{OI} (°C)
ACETONE	C_3H_6O	0.129	1457
ACETYLENE	C_2H_2	0.085	1267
BENZENE	C_6H_6	0.133	1537
BUTANE	C_4H_{10}	0.134	1557
ETHANE	C_2H_6	0.118	1357
ETHANOL	C_2H_6O	0.126	1397
ETHYLENE	C_2H_4	0.105	1337
HYDROGEN	H_2	0.054	807
ISOBUTANE	C_4H_{10}	0.134	1557
ISOOCTANE	C_8H_{18}	0.134	1507
ISOPROPANOL	C_3H_8O	0.128	1427
METHANE	CH_4	0.139	1507
METHANOL	CH_3OH	0.111	1257
DECANE	$C_{10}H_{22}$	0.135	1507
N-HEPTANE	C_7H_{16}	0.134	1497
N-HEXANE	C_6H_{14}	0.134	1497
N-OCTANE	C_8H_{18}	0.134	1507
N-PENTANE	C_5H_{12}	0.134	1487
PROPANE	C_3H_8	0.127	1447
PROPANOL	C_3H_8O	0.128	1427
All other species		0.135	1427

Extinction Model Verification

In the verification cases called `Extinction/extinction_x.fds`, 100 small, sealed boxes are initialized with temperatures from 300 K to 1875 K and oxygen mass fractions from 0.0115 to 0.2185, combined with the stoichiometric amount of fuel, and the remainder nitrogen. A one-step, complete reaction of methane is assumed:



Based on the oxygen and methane concentration and the temperature in each box, combustion can or cannot occur according to the 'EXTINCTION 1' and 'EXTINCTION 2' models, as shown in Fig. 13.3. The FDS predictions based on the 'EXTINCTION 2' model (right) confirm that conditions that sustain burning (red crosses) or cause extinction (blue stars) fall closely on the expected results using thermo-physical properties of the reactants and products. The simpler model, 'EXTINCTION 1' (left), merely conforms to the specified Burn-No Burn criterion. More detailed information on the extinction models can be found in the Technical Reference Guide [1].

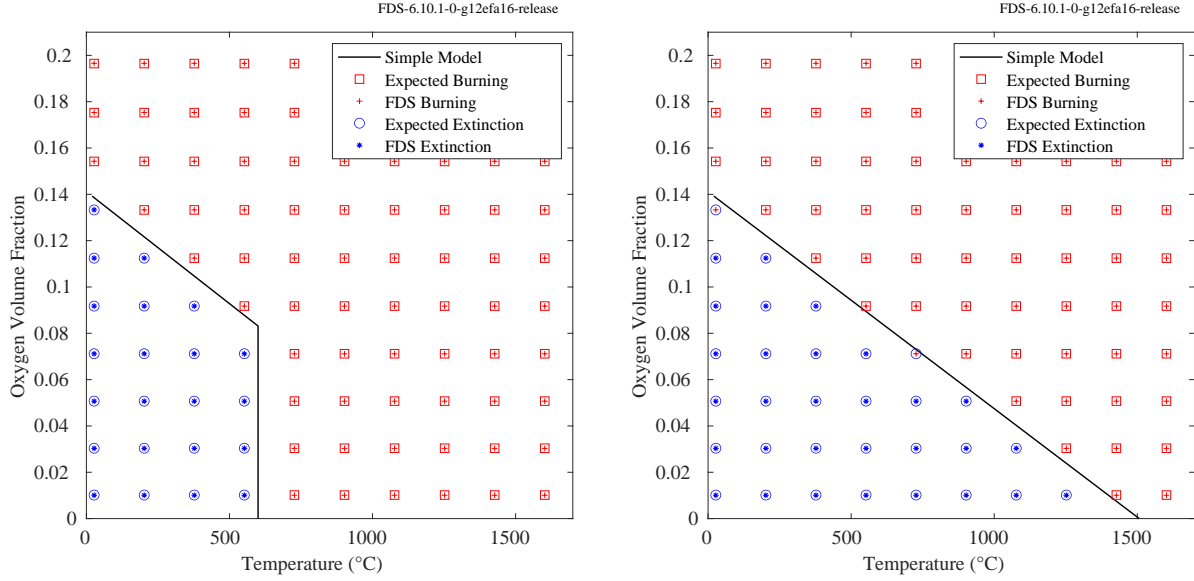


Figure 13.3: Result of 'EXTINCTION 1' (left) and 'EXTINCTION 2' (right) for the given initial temperatures and oxygen concentrations.

13.1.7 Piloted Ignition

By default, FDS employs either the 'EXTINCTION 1' or 'EXTINCTION 2' model to determine if combustion is viable. If local extinction occurs, the unburned fuel gas can re-ignite if it mixes with sufficient oxygen elsewhere in the domain, even if this re-ignition might not happen in reality because the temperature might be too low to sustain combustion. To prevent spurious re-ignition from happening, you can set the `AUTO_IGNITION_TEMPERATURE` on the `REAC` line, in °C, below which combustion will not occur. For most fuels of interest, the AIT may be found in Beyler's chapter of the SFPE Handbook [17]. The default AIT is -273°C , meaning that fuel and oxygen will burn when mixed providing that either of the extinction models allows combustion to occur.

The value of AIT may need to be lowered for cases where the grid size is greater than 10 cm. The purpose of AIT is simply to prevent fuel and oxygen from spontaneously igniting, which is the default behavior of FDS. If an AIT is specified, then you must also specify some form of heat/ignition source must be present to start the fire; or you can specify the real sextuplet `AIT_EXCLUSION_ZONE` on the `REAC` line to designate a volume in which ignition occurs spontaneously even when the AIT is enforced everywhere else. Note that you can specify multiple exclusion zones as follows:

```
&REAC ..., AIT_EXCLUSION_ZONE(1:6,1)=..., AIT_EXCLUSION_ZONE(1:6,2)=... /
```

You can control the turning on and off of an exclusion zone using either `AIT_EXCLUSION_ZONE_DEVC_ID(N)` or `AIT_EXCLUSION_ZONE_CTRL_ID(N)` where `N` refers to the index of the listed exclusion zones. If there is only one exclusion zone, you need not add an index. If the status of the specified device or controller is true, the exclusion zone is in effect.

An alternative to the ignition zone method is to specify a "pilot fuel" with a low AIT. In cells where the pilot fuel is present and the local cell temperature is above the AIT specified for the specific pilot fuel reaction, combustion is allowed to proceed. For example, you can initialize a series of particles with a controlled mass flux of fuel:

```
&REAC FUEL='METHANE', ..., AUTO_IGNITION_TEMPERATURE=540. /
&REAC FUEL='ETHYLENE', ..., AUTO_IGNITION_TEMPERATURE=-273. /
```

With the appropriate chemistry defined, the above REAC lines will allow for ethylene to be used as a pilot fuel, avoiding an ignition threshold temperature for any cell where ethylene and its oxidizer are present.

You can also simulate a spark ignition using the following lines of input:

```
&REAC FUEL='METHANE', AUTO_IGNITION_TEMPERATURE=540 /
&PROP ID='spark', SPARK=T /
&DEVC ID='spark-1', XYZ=..., QUANTITY='TIME', SETPOINT=10, PROP_ID='spark' /
```

In this case, methane and air will not combust unless the grid cell temperature exceeds 540 °C. However, within the grid cell containing the specified point XYZ, a “spark” is generated at 10 s which effectively causes the AIT to go to 0 K and ignition of fuel and air is allowed, assuming a suitable amount of fuel and air is present.

13.1.8 Local Quenching (AIT per ZONE)

There may be special circumstances where you would like to specify that a localized region of the domain cannot support combustion. In this case, you may use the AIT_EXCLUSION_ZONE construct to specify a locally high value of the ignition temperature threshold, which will prevent combustion in this region. For example,

```
&REAC ..., AIT_EXCLUSION_ZONE(1:6,1)=..., AIT_EXCLUSION_ZONE_TEMPERATURE(1)=1000.
      AIT_EXCLUSION_ZONE(1:6,2)=..., AIT_EXCLUSION_ZONE_TEMPERATURE(2)=3000./
```

13.2 Complex Stoichiometry

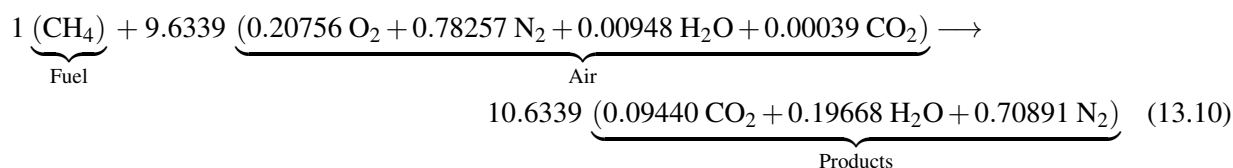
The “simple chemistry” parameters described above can only be used when the fuel molecule contains only C, O, H, and N. For any other situation, you must specify the reaction stoichiometry in greater detail. This means that you must explicitly specify the gas species, or species mixtures, along with the stoichiometry of the reaction.

13.2.1 Balancing the Atoms

Consider a single reaction involving methane. When you specify the REAC line to be:

```
&REAC FUEL='METHANE' /
```

FDS assumes the following reaction:



By default, there are trace amounts of carbon dioxide and water vapor in the air, which, like the nitrogen, is carried along in the reaction. This is important, because the more complicated way to specify a single step reaction of methane is as follows:

```
&SPEC ID='NITROGEN',          LUMPED_COMPONENT_ONLY=T /
&SPEC ID='OXYGEN',            LUMPED_COMPONENT_ONLY=T /
&SPEC ID='WATER VAPOR',       LUMPED_COMPONENT_ONLY=T /
&SPEC ID='CARBON DIOXIDE',    LUMPED_COMPONENT_ONLY=T /
&SPEC ID='METHANE' /

&SPEC ID='AIR', BACKGROUND=T,
  SPEC_ID(1)='OXYGEN',          VOLUME_FRACTION(1)=0.20756,
  SPEC_ID(2)='NITROGEN',        VOLUME_FRACTION(2)=0.78257,
  SPEC_ID(3)='WATER VAPOR',     VOLUME_FRACTION(3)=0.00948,
  SPEC_ID(4)='CARBON DIOXIDE',  VOLUME_FRACTION(4)=0.00039 /

&SPEC ID='PRODUCTS',
  SPEC_ID(1)='CARBON DIOXIDE',   VOLUME_FRACTION(1)=0.09438,
  SPEC_ID(2)='WATER VAPOR',     VOLUME_FRACTION(2)=0.19663,
  SPEC_ID(3)='NITROGEN',        VOLUME_FRACTION(3)=0.70899/

&REAC FUEL='METHANE', SPEC_ID_NU='METHANE','AIR','PRODUCTS',
      NU=-1,-9.6357,10.6357, HEAT_OF_COMBUSTION=50000. /
```

The reaction stoichiometry is specified using the stoichiometric coefficients, $\text{NU}(N)$, corresponding to the tracked³ species, $\text{SPEC_ID_NU}(N)$. If complex stoichiometry is being used, a FUEL input is not required. If no FUEL input is given, FDS will assume it is the first species in SPEC_ID_NU with a net negative value for NU (i.e., if a species appeared on both sides of the reaction as a collision species it would not be used as the FUEL). There are several parameters on the REAC line that control the specification of the stoichiometry:

³A “tracked” species is one for which LUMPED_COMPONENT_ONLY is F

- **CHECK_ATOM_BALANCE** If chemical formulas are provided for all species that participate in a reaction, then FDS will check the stoichiometry to ensure that atoms are conserved. Setting this flag to **F** will bypass this check. (Default **T**)
- **REAC_ATOM_ERROR** Error tolerance in units of atoms for the reaction stoichiometry check. (Default 1.E-4)
- **REAC_MASS_ERROR** Relative error tolerance computed as (mass of products - mass of reactants)/(mass of products) for the reaction stoichiometry mass balance check. (Default 1.E-4)

13.2.2 Complex Fuel Molecules

Fuels Compatible with Simple Chemistry For complex fuel molecules that contain only C, H, N, and O the simple chemistry reaction parameters can still be applied. Consider natural gas, which is often a mixture of several component gases. To build the fuel mixture, the primitive (component) species need be defined. Each primitive species will get a **LUMPED_COMPONENT_ONLY** designation, which means that each of these species will not be explicitly tracked as they are components to a mixture. Note that these lumped components can only contain C, H, N, and O atoms. The lumped fuel, 'natural gas', can be created using the defined primitive components and subsequently the reaction can be defined using simple chemistry (Section 13.1.1):

```
&SPEC ID='METHANE',          LUMPED_COMPONENT_ONLY=T/
&SPEC ID='ETHYLENE',         LUMPED_COMPONENT_ONLY=T/
&SPEC ID='NITROGEN',         LUMPED_COMPONENT_ONLY=T/
&SPEC ID='CARBON DIOXIDE', LUMPED_COMPONENT_ONLY=T/

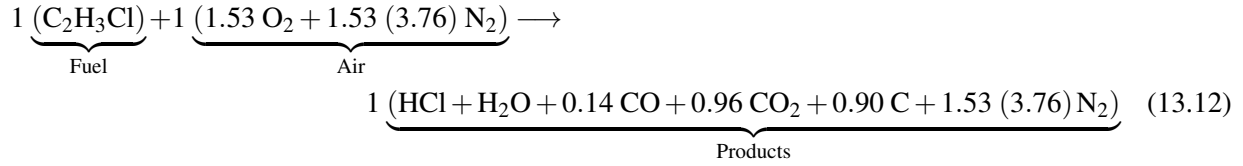
&SPEC ID='natural gas'
  SPEC_ID(1)='METHANE',      VOLUME_FRACTION(1)=92.2
  SPEC_ID(2)='ETHYLENE',     VOLUME_FRACTION(2)= 3.3
  SPEC_ID(3)='NITROGEN',     VOLUME_FRACTION(3)= 3.9
  SPEC_ID(4)='CARBON DIOXIDE',VOLUME_FRACTION(4)= 0.6/

&REAC FUEL='natural gas'
  SOOT_YIELD=0.01 /
```

Fuels Not Compatible with Simple Chemistry Fires, however, often involve fuels that do not just consist of C, H, N, and O. For example, chlorine is commonly found in building and household materials, and because of its propensity to form the acid gas HCl, you may want to account for it in the basic reaction scheme. Suppose the predominant fuel in the fire is polyvinyl chloride (PVC). Regardless of its detailed polymeric structure, it can be regarded as C_2H_3Cl for the purpose of modeling. Assuming that all of the Cl in the fuel is converted into HCl, you can derive a single-step reaction mechanism using appropriate soot and CO yields for the specified fuel. In this example, the SFPE Handbook [18] is used to find soot and CO yields for PVC; 0.172 and 0.063, respectively. For a given species, α , its stoichiometric coefficient, ν_α , can be found from its yield, y_α , and its molecular weight, W_α , according to the formula:

$$\nu_\alpha = \frac{W_F}{W_\alpha} y_\alpha \quad (13.11)$$

Since it is assumed that all of the Cl is converted to HCL, the remainder of the stoichiometric coefficients come from an atom balance. An equation can now be written to include the appropriate numerical values for the stoichiometric coefficients.



The choice of fuel in this example, PVC, is not defined in Appendix A, therefore its properties must be defined on a `SPEC` line. In this example, we use the species' chemical formula. The example will also use the lumped species formulation to minimize the number of scalar transport equations that need to be solved. Therefore, each species that does not have an explicit transport equation is a `LUMPED_COMPONENT_ONLY`.

```

&SPEC ID = 'PVC', FORMULA = 'C2H3Cl' /
&SPEC ID = 'OXYGEN',          LUMPED_COMPONENT_ONLY = T /
&SPEC ID = 'NITROGEN',        LUMPED_COMPONENT_ONLY = T /
&SPEC ID = 'HYDROGEN CHLORIDE', LUMPED_COMPONENT_ONLY = T /
&SPEC ID = 'WATER VAPOR',      LUMPED_COMPONENT_ONLY = T /
&SPEC ID = 'CARBON MONOXIDE',  LUMPED_COMPONENT_ONLY = T /
&SPEC ID = 'CARBON DIOXIDE',   LUMPED_COMPONENT_ONLY = T /
&SPEC ID = 'SOOT',             LUMPED_COMPONENT_ONLY = T /

```

For the oxidizer and products, which are both composed of multiple primitive species, `SPEC` lines are needed to define the composition of the lumped species. You can define the `SPEC` using either the `MASS_FRACTIONS` of the component gases or the `VOLUME_FRACTIONS`. If Eq. (13.12) is properly balanced, you can directly use the stoichiometric coefficients of the primitive species to define the lumped species.

```

&SPEC ID='AIR', BACKGROUND=T
  SPEC_ID(1)='OXYGEN',  VOLUME_FRACTION(1)=1.53,
  SPEC_ID(2)='NITROGEN', VOLUME_FRACTION(2)=5.76 /

&SPEC ID='PRODUCTS',
  SPEC_ID(1)='HYDROGEN CHLORIDE', VOLUME_FRACTION(1)=1.0,
  SPEC_ID(2)='WATER VAPOR',       VOLUME_FRACTION(2)=1.0,
  SPEC_ID(3)='CARBON MONOXIDE',   VOLUME_FRACTION(3)=0.14,
  SPEC_ID(4)='CARBON DIOXIDE',    VOLUME_FRACTION(4)=0.96,
  SPEC_ID(5)='SOOT',              VOLUME_FRACTION(5)=0.90,
  SPEC_ID(6)='NITROGEN',          VOLUME_FRACTION(6)=5.76 /

```

To set the initial concentration of fuel, an `INIT` line is used:

```

&INIT MASS_FRACTION(1)=0.229, SPEC_ID(1)='PVC' /

```

Since this is not a simple chemistry problem, either the enthalpy of formation of PVC or the heat of combustion of the reaction should be specified. In this case, the heat of combustion for PVC is taken from the SFPE Handbook [18].

```

&REAC FUEL='PVC', HEAT_OF_COMBUSTION=16400, SPEC_ID_NU='PVC','AIR','PRODUCTS',
  NU=-1,-1,1 /

```

Note that the sign of `NU` corresponds to whether that species is consumed (-) or produced (+). Figure 13.4 displays the mass fractions of the product species for the sample case `PVC_Combustion`. Also note that the values for `NU` reflect the composition of the species as defined on the `SPEC` lines. If, for example, the PVC `SPEC` was defined as `CH1.5Cl0.5`, then the `REAC` would require `NU=-2,-1,1`. A fixed turbulent mixing time (`FIXED_MIXED_TIME`) of 0.1 s is used for this example only because the reactants are initially mixed within a chamber with no imposed flow. Normally, this parameter is not necessary.

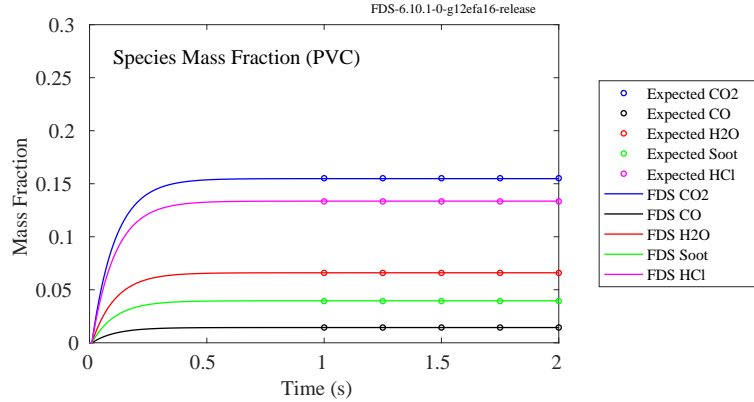
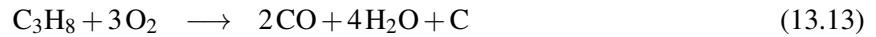


Figure 13.4: Product species mass fractions for model PVC example.

13.2.3 Multiple Fast Reactions

For large-scale fire simulations, the numerical grid is usually too coarse to resolve detailed, finite-rate flame chemistry. However, you may still specify a combustion scheme that consists of multiple fast reactions. For example, consider this three step scheme that describes the formation of CO and soot as intermediate species in the combustion of propane:



In the first step, the fuel is converted quickly to CO, H₂O and soot (C), and then the CO and soot are converted to CO₂ if there is available oxygen. All the reactions are “fast” relative to the mixing time of the reactants within a grid cell, but the second and third reactions are assumed to occur more slowly than the first. By default, FDS forces all fast reactions to occur instantaneously and simultaneously. However, if you want to specify that the reactions occur serially, use the parameter `PRIORITY` as in the following set of input lines:

```
&SPEC ID='NITROGEN',          BACKGROUND=T /
&SPEC ID='PROPANE',           MASS_FRACTION_0=0.0 /
&SPEC ID='OXYGEN',            MASS_FRACTION_0=0.23 /
&SPEC ID='WATER VAPOR',       MASS_FRACTION_0=0.0 /
&SPEC ID='CARBON MONOXIDE',    MASS_FRACTION_0=0.0 /
&SPEC ID='CARBON DIOXIDE',     MASS_FRACTION_0=0.0 /
&SPEC ID='SOOT',              MASS_FRACTION_0=0.0 /

&REAC ID='R1'
  FUEL='PROPANE'
  SPEC_ID_NU='PROPANE','OXYGEN','CARBON MONOXIDE','WATER VAPOR','SOOT'
  NU=-1,-3,2,4,1 /
&REAC ID='R2'
  FUEL='CARBON MONOXIDE'
  SPEC_ID_NU='CARBON MONOXIDE','OXYGEN','CARBON DIOXIDE'
  NU=-1,-0.5,1
  PRIORITY=2 /
&REAC ID='R3'
  FUEL='SOOT'
```

```

SPEC_ID_NU='SOOT','OXYGEN','CARBON DIOXIDE'
NU=-1,-1,1
PRIORITY=2 /

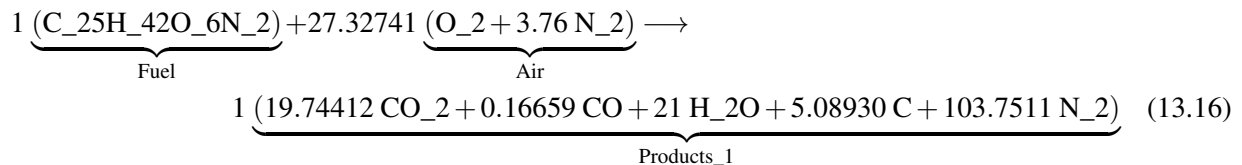
```

By specifying `PRIORITY=2` for reactions `R2` and `R3`, you are assuming that the first reaction, `R1`, occurs first, followed by `R2` and `R3`. This sequence allows for the build-up of soot and CO in compartments that are oxygen starved.

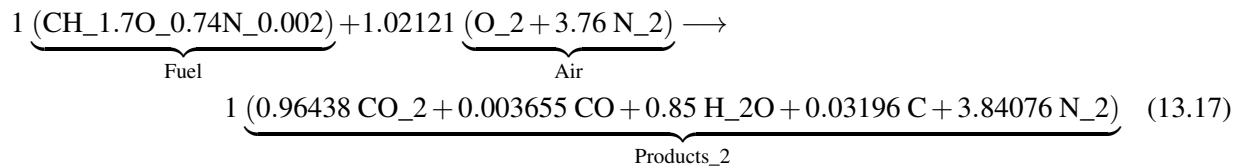
Note that in this reaction scheme, the extinction model is applied to the first reaction but not the second and third because the default extinction model is only applied to the first of multiple fast reactions. For more information on extinction, see Sec. 13.1.6.

13.2.4 Multiple Fuels

There may be times when your design/model fire is best described by more than one fuel or by more than a single-step chemical reaction. For this example consider two simultaneous, mixing-controlled reactions of polyurethane and wood. Both of these fuels are complex molecules not included in Appendix A, so they must be defined on the `SPEC` line. Polyurethane is defined by the chemical formula $C_{25}H_{42}O_6N_2$ and wood is defined by $CH_{1.7}O_{0.74}N_{0.002}$. Consider a combustion reaction for polyurethane with a soot yield of $y_s = 0.131$ and a CO yield of $y_{CO} = 0.01$ [18] is:



and the reaction for wood with a soot yield of $y_s = 0.015$ and a CO yield of $y_{CO} = 0.004$ [18] is:



Similar to example in Sec. 13.2.2, we will use the lumped species approach to minimize the number of species that FDS needs to transport. Therefore, the species are defined in the following manner:

```

&SPEC ID = 'POLYURETHANE', FORMULA = 'C25H42O6N2' /
&SPEC ID = 'WOOD', FORMULA = 'CH1.7O0.74N0.002' /
&SPEC ID = 'OXYGEN', LUMPED_COMPONENT_ONLY = T /
&SPEC ID = 'NITROGEN', LUMPED_COMPONENT_ONLY = T /
&SPEC ID = 'WATER VAPOR', LUMPED_COMPONENT_ONLY = T /
&SPEC ID = 'CARBON MONOXIDE', LUMPED_COMPONENT_ONLY = T /
&SPEC ID = 'CARBON DIOXIDE', LUMPED_COMPONENT_ONLY = T /
&SPEC ID = 'SOOT', LUMPED_COMPONENT_ONLY = T /

```

Examination of Eq. (13.16) and Eq. (13.17) shows that, while `Products_1` and `Products_2` are composed of the same species, the species do not exist in the same proportion. As a result, we must construct two separate product lumped species.

```

&SPEC ID = 'AIR',
SPEC_ID(1) = 'OXYGEN', VOLUME_FRACTION(1)=1,
SPEC_ID(2) = 'NITROGEN', VOLUME_FRACTION(2)=3.76,

```

```

BACKGROUND=T /

&SPEC ID = 'PRODUCTS_1',
  SPEC_ID(1) = 'CARBON DIOXIDE', VOLUME_FRACTION(1) = 19.74412,
  SPEC_ID(2) = 'CARBON MONOXIDE', VOLUME_FRACTION(2) = 0.16659,
  SPEC_ID(3) = 'WATER VAPOR', VOLUME_FRACTION(3) = 21.,
  SPEC_ID(4) = 'SOOT', VOLUME_FRACTION(4) = 5.08930,
  SPEC_ID(5) = 'NITROGEN', VOLUME_FRACTION(5) = 103.7511 /

&SPEC ID = 'PRODUCTS_2',
  SPEC_ID(1) = 'CARBON DIOXIDE', VOLUME_FRACTION(1) = 0.96438,
  SPEC_ID(2) = 'CARBON MONOXIDE', VOLUME_FRACTION(2) = 0.00365,
  SPEC_ID(3) = 'WATER VAPOR', VOLUME_FRACTION(3) = 0.85,
  SPEC_ID(4) = 'SOOT', VOLUME_FRACTION(4) = 0.03196,
  SPEC_ID(5) = 'NITROGEN', VOLUME_FRACTION(5) = 3.84076 /

```

Once we have constructed the lumped species, we can define the REAC lines.

```

&REAC ID = 'plastic',
  FUEL = 'POLYURETHANE',
  HEAT_OF_COMBUSTION=26200,
  SPEC_ID_NU = 'POLYURETHANE', 'AIR', 'PRODUCTS_1'
  NU=-1,-27.32741,1 /

&REAC ID = 'wood'
  FUEL = 'WOOD',
  HEAT_OF_COMBUSTION=16400,
  SPEC_ID_NU = 'WOOD', 'AIR', 'PRODUCTS_2'
  NU=-1,-1.02121,1 /

```

In both of these reactions, the ENTHALPY_OF_FORMATION of the chosen fuels is unknown to FDS, so the HEAT_OF_COMBUSTION is specified for each reaction. Typically, the heat release per unit area (HRRPUA) parameter is used to specify the fire size on the SURF line along with SPEC_ID if there are multiple REAC inputs and MASS_FRACTION if the HRRPUA is a mixture of fuels. Multiple fuels can also be used by specifying the mass flux of each fuel. In this example, we want both fuels to flow out of the same burner and ramp up to a 1200 kW fire after 60 s. First, we create a 1 m² burner by defining a VENT line:

```
&VENT XB=4.0,5.0,4.0,5.0,0.0,0.0, SURF_ID='FIRE1', COLOR='RED' /
```

The VENT points to the SURF_ID='FIRE1' which we can define using both fuels. The mass flux values for each fuel are determined by the desired heat release, the proportion of the total heat release rate each fuel contributes, the burner area, and the heat of combustion of each fuel. In this case we want a 1200 kW fire where each fuel contributes 50% to the total heat release rate (600 kW).

$$\dot{m}''_{\text{poly}} = \frac{600 \text{ kW}}{1 \text{ m}^2} \frac{1}{26200 \text{ kJ/kg}} = 0.022901 \text{ kg/(m}^2\text{s)} \quad (13.18)$$

$$\dot{m}''_{\text{wood}} = \frac{600 \text{ kW}}{1 \text{ m}^2} \frac{1}{16400 \text{ kJ/kg}} = 0.036585 \text{ kg/(m}^2\text{s)} \quad (13.19)$$

```

&SURF ID='FIRE1', SPEC_ID(1)='POLYURETHANE', MASS_FLUX(1)=0.022901, RAMP_MF(1)='poly'
  SPEC_ID(2)='WOOD', MASS_FLUX(2)=0.036585, RAMP_MF(2)='wood' /

```

Note that the SPEC_ID, MASS_FLUX, and RAMP_MF correspond to one another for each fuel. It does not matter which fuel is (1), just that the numbering is consistent. We also want each fuel to follow a ramp such that

fire starts at 0 kW at the initial time, reaches 1200 kW at 100 s, and remains at 1200 kW for the remainder of a 600 s simulation.

```
&RAMP ID='poly', T = 0, F = 0.0 /
&RAMP ID='poly', T = 50 , F = 0.5 /
&RAMP ID='poly', T = 100, F = 1.0 /
&RAMP ID='poly', T = 600, F = 1.0 /

&RAMP ID='wood', T = 0 , F = 0.0 /
&RAMP ID='wood', T = 50 , F = 0.5 /
&RAMP ID='wood', T = 100, F = 1.0 /
&RAMP ID='wood', T = 600, F = 1.0 /
```

Here, T corresponds to the time and F is the fraction of the mass flux specified on the `SURF` line. Fig. 13.5 compares the resulting FDS heat release rate (HRR) to the expected HRR from the defined ramp. Note that in this simulation, there is 10 s averaging on the FDS output HRR (`&DUMP DT_HRR=10`) and the expected results account for this averaging.

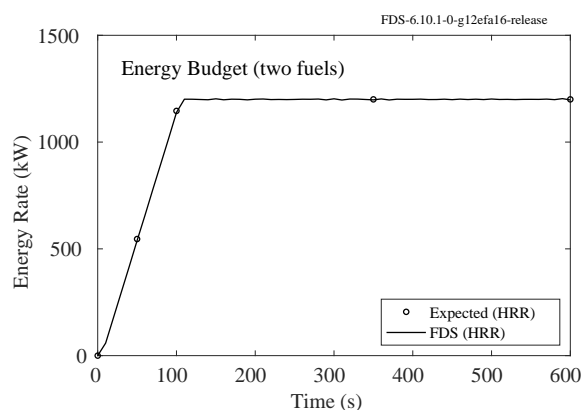


Figure 13.5: HRR for `energy_budget_adiabatic_two_fuels` test case.

Note that when using multiple chemical reactions, you must set `SUPPRESSION=F` on the `COMB` line.

13.2.5 Using the `EQUATION` input parameter

When specifying a reaction scheme using all primitive variables (*i.e., no lumped species*), a convenient shortcut for specifying the reaction is via the input parameter `EQUATION`, which allows the specification of the reaction in text form. The rules are:

- The species must be explicitly tracked.
- The species name is its chemical formula as defined on the `SPEC` line or by the species `ID`.
- The stoichiometry is given before each species and is separated by an asterisk. Real numbers are allowed but exponential notation is not (*i.e.*, 201.1 but not 2.011E2).
- The reactants and products are separated by an equals sign.

For example, if the reaction defines the complete combustion of methane using primitive species, then the following would be equivalent:

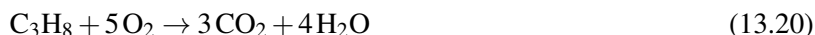
```
&REAC FUEL='METHANE', EQUATION = 'METHANE+2*OXYGEN=CARBON DIOXIDE+2*WATER VAPOR' /  
&REAC FUEL='METHANE', EQUATION = 'CH4+2*O2=CO2+2*H2O' /  
&REAC FUEL='METHANE', EQUATION = 'METHANE+2*O2=CO2+2*H2O' /
```

13.3 Finite Rate Combustion

By default, FDS uses a mixing-controlled combustion model, meaning that the reaction rate is infinite and limited only by species concentrations. However, FDS can also employ finite-rate reactions using an Arrhenius model. Since finite-rate reactions require a fully-resolved temperature field (i.e., a simulation on a grid capable of resolving the flame temperature), it is recommended that finite-rate reactions be invoked only when FDS is running in DNS mode (`SIMULATION_MODE='DNS'` on the `MISC` line). A specific finite-rate reaction will only be computed if all species consumed or otherwise required (third-body or catalyst) in the reaction are present with a mass fraction greater than `ZZ_MIN_GLOBAL` and the local cell temperature is above `FINITE_RATE_MIN_TEMP`, both specified on `COMB`. The default value for `ZZ_MIN_GLOBAL` is 1×10^{-10} , which is generally adequate. An exception might be for detailed chemistry where equations involving free-radicals are used. The user should consider setting `FINITE_RATE_MIN_TEMP` to be around 100 °C to avoid unnecessary and expensive time integration of the ODEs for detailed kinetics.

Single Step Reaction

Consider a single-step reaction mechanism for complete propane combustion:



In this case, we explicitly define all primitive species:

```
&SPEC ID='NITROGEN', BACKGROUND=T/  
&SPEC ID='PROPANE' /  
&SPEC ID='OXYGEN' /  
&SPEC ID='WATER VAPOR' /  
&SPEC ID='CARBON DIOXIDE' /
```

The rate expression for the fuel, C_3H_8 , is

$$\frac{dC_{\text{C}_3\text{H}_8}}{dt} = -k \prod C_{\alpha}^{N_{S,\alpha}} \quad (13.21)$$

where C_{α} is the concentration of species α in units of mol/cm³ and the rate constant is defined as

$$k = A T^{N_T} e^{-E_a/RT} \quad (13.22)$$

- A is the pre-exponential factor [((mol/cm³)¹⁻ⁿ)/s], where $n = \sum N_{S,\alpha}$ is the *order* of the reaction,
- E_a is the activation energy [J/mol],
- $N_{S,\alpha}$ is an array containing the concentration exponents (default values are the stoichiometric coefficients),
- N_T is the temperature exponent (default 0, meaning no temperature dependence),
- R is the universal gas constant, 8.314 J/(mol·K).

If we use Arrhenius parameters found from experiments or the literature (for this case Westbrook and Dryer [19]), the rate expression for the single-step propane reaction defined by Eq. (13.20) becomes:

$$\frac{dC_{\text{C}_3\text{H}_8}}{dt} = -8.6 \times 10^{11} T^0 e^{-125520/RT} C_{\text{C}_3\text{H}_8}^{0.1} C_{\text{O}_2}^{1.65} \quad (13.23)$$

and the resulting `REAC` line is:


```

&REAC ID = 'R1'
      FUEL = 'PROPANE'
      A = 8.6e11
      E = 125520
      SPEC_ID_NU = 'PROPANE', 'OXYGEN', 'CARBON DIOXIDE', 'WATER VAPOR'
      NU = -1, -5, 3, 4
      SPEC_ID_N_S = 'PROPANE', 'OXYGEN'
      N_S = 0.1, 1.65 /

```

The array `SPEC_ID_NU` contains the list of tracked species participating in the reaction as a product or reactant. Note that a primitive species with `LUMPED_COMPONENT_ONLY=T` cannot be used in a reaction since it is not tracked separately. The array `SPEC_ID_N_S` contains the list of species with the exponents, `N_S`, in Eq. (13.21). This array must contain only primitive species, i.e., no species mixtures. The array of stoichiometric coefficients, `NU`, can be taken directly from Eq. (13.20). If any of `SPEC_ID_NU` is a lumped species and a reactant (this would include product species when `REVERSE=T`), then `SPEC_ID_N_S` and `N_S` must be defined to identify which specific species inside the lumped species are reactants.

13.3.1 Multiple Step Reaction

If we extend Eq. (13.20) to include reactions from Westbrook and Dryer [19] that account for carbon monoxide production, we obtain:



In this case we will combine fast chemistry with finite-rate chemistry to create a mixed reaction mechanism. As before, we use primitive species rather than mixtures. A `REAC` line is needed for each reaction including the reverse reaction. The propane oxidation reaction is a fast chemistry while the reversible carbon monoxide reaction is finite-rate. The Arrhenius parameters are modified from Andersen et al. [20]:

```

&REAC ID = 'R1'
      FUEL = 'PROPANE'
      SPEC_ID_NU = 'PROPANE', 'OXYGEN', 'CARBON MONOXIDE', 'WATER VAPOR'
      NU = -1, -3.5, 3, 4 /

&REAC ID = 'R2'
      FUEL = 'CARBON MONOXIDE'
      A = 1.5e9
      E = 41840
      SPEC_ID_NU = 'CARBON MONOXIDE', 'OXYGEN', 'CARBON DIOXIDE'
      NU = -1, -0.5, 1
      SPEC_ID_N_S = 'OXYGEN', 'CARBON MONOXIDE', 'WATER VAPOR'
      N_S = 0.25, 1, 0.5 /

&REAC ID = 'R3'
      FUEL = 'CARBON DIOXIDE'
      A = 6.16e13
      E = 328026
      SPEC_ID_NU = 'CARBON DIOXIDE', 'OXYGEN', 'CARBON MONOXIDE'
      NU = -1, 0.5, 1
      SPEC_ID_N_S = 'OXYGEN', 'CARBON DIOXIDE', 'WATER VAPOR'
      N_S = -0.25, 1, 0.5
      N_T = -0.97 /

```

For the reaction R2, the reaction rate depends on the water vapor concentration, as indicated by its assignment of a value of `N_S`. However, it does not explicitly participate in the reaction because it is not assigned a stoichiometric coefficient, `NU`. Reactions of this sort are often written in textbooks as



However, for the purposes of inputting into FDS, since H_2O is neither a product nor a reactant it would not be specified in the chemical reaction using either `EQUATION` or `NU`. It would instead be given a rate exponent using `N_S` to indicate that its presence is required.

As discussed previously (Sec. 13.1.2), energy release is calculated using the net change of species in a time step along with the enthalpy of formation of each species. This approach makes specifying an endothermic reaction, such as the reversible CO_2 reaction (R3), no different than typical FDS exothermic combustion reactions.

13.3.2 Finite Rate Chemistry using a Detailed Chemical Mechanism

A detailed chemical mechanism file can be specified to solve finite rate chemistry problems. For example, the GRI-3.0 [21] mechanism can be used to simulate methane combustion. Such a mechanism file contains various species and the relevant reactions representing the chemical pathways of fuel combustion. Users must ensure that the chosen mechanism is well-suited for their specific problem. In FDS, an FDS-formatted mechanism file can be specified using the following line:

```
&CATF OTHER_FILES= '<path to the fds chemical mechanism file>' /
```

FDS-formatted chemical mechanism files can be generated from Chemkin [22, 23] or Cantera YAML format mechanism files following the instructions described below. For user convenience, a few example mechanism files are already included in the `Utilities/Input_Libraries/Chemical_Mechanisms/FDS/` directory of the GitHub repository `firemodels/fds`. The steps to generate the FDS chemical mechanism file are as follows:

1. Ensure that the chemistry reactions file, thermodynamic properties file, and transport properties file are available in Chemkin format.
2. Use the following Cantera [24] command to convert these three files to a YAML file

```
ck2yaml --input=<reactions file> --thermo=<thermodynamic properties file>
--transport=<transport properties file> --output=<output.yaml>
```

3. Using the utility Python script `Utilities/Python/scripts/cantera_yaml_2_fds.py` from the `firemodels/fds` GitHub repository, convert the `output.yaml` file to an FDS-formatted chemical mechanism file. Refer to the comments in the Python script for detailed usage.

```
python Utilities/Python/scripts/cantera_yaml_2_fds.py output.yaml N2 > output.fds
```

If the YAML file of the mechanism is directly available, steps 1 and 2 can be skipped.

A detailed chemical mechanism is currently solved using the Sundials CVODE ODE solver [25]. For details on configuring the FDS input for the CVODE solver, please refer to Section 13.3.7.

13.3.3 Reaction Rates from Equilibrium Constants

In some cases, only the forward rate parameters will be known for a reversible reaction, and the reverse rate parameters must be obtained from equilibrium. This can be achieved by adding `REVERSE=T` on the `REAC` line. FDS will create a reverse reaction where the `NU` values for the reverse reaction are set to `-NU` for the forward reaction. The reverse rate constant will be determined using the change in Gibbs free energy of the reaction. The `ID` for the reverse reaction will be the `ID` of the forward reaction with `_R` appended. Below is an example (taken from [26]) for hydrogen combustion with oxygen and the corresponding reverse water vapor dissociation reaction. In this case the reverse reaction `ID` would be `R1_R`.



```
&REAC ID = 'R1'
  REVERSE = T
  FUEL = 'HYDROGEN'
  A = 0.85e16
  E = 167360.
  RADIATIVE_FRACTION=0.0
  SPEC_ID_NU='HYDROGEN', 'OXYGEN', 'WATER VAPOR'
  SPEC_ID_N_S='HYDROGEN', 'OXYGEN'
  NU= -1,-0.5,1
  N_S = 0.25,1.5
  N_T = -1 /
```

13.3.4 Third Body Reactions

Some reactions require collision with a third body (any other molecule) to activate the reaction. Reactions of this type are often write in textbooks as



This type of reaction is specified by adding `THIRD_BODY=T` to the `REAC` line for the reaction. The species `M` should not otherwise be defined. In some third body reactions one or more gas species may be more or less effective as a collision species. Any species with a non-unity collision efficiency can be listed using `THIRD_EFF_ID(N)` and `THIRD_EFF(N)` to define the collision efficiencies of specific species.

13.3.5 Fall-off Reactions

Fall-off reactions are a type of reaction where rates depend on both pressure and temperature, as described in the FDS Tech Guide [3]. These reactions are prevalent in detailed chemical mechanisms (see Section 13.3.2). Following are the example of Lindemann-fall-off and Troe-fall-off reaction entries from GRI-3 mechanism [21] in FDS format. SRI-fall-off reactions are not supported yet, and will be incorporated in future.

```
&REAC ID='R12',
  REACTYPE='FALLOFF-LINDEMANN',
  REVERSE=T,
  RADIATIVE_FRACTION=0,
  A= 1.80000e+10 ,
  THIRD_BODY=T,
  THIRD_EFF_ID= 'AR', 'C2H6', 'CH4', 'CO', 'CO2', 'H2', 'H2O', 'O2' ,
```

```

THIRD_EFF= 0.5,3.0,2.0,1.5,3.5,2.0,6.0,6.0 ,
E= 9978.84 ,
N_T= 0.0 ,
A_LOW_PR= 6.02000e+14 ,
E_LOW_PR= 12552.0 ,
N_T_LOW_PR= 0.0 ,
SPEC_ID_NU= 'CO','O' , 'CO2' ,
NU= -1.0,-1.0 , 1.0 /

&REAC ID='R50',
  REACTYPE='FALLOFF-TROE',
  REVERSE=T,
  RADIATIVE_FRACTION=0,
  A= 6.00000e+14 ,
  THIRD_BODY=T,
  THIRD_EFF_ID= 'AR','C2H6','CH4','CO','CO2','H2','H2O' ,
  THIRD_EFF= 0.7,3.0,2.0,1.5,2.0,2.0,6.0 ,
  E= 0.0 ,
  N_T= 0.0 ,
  A_LOW_PR= 1.04000e+26 ,
  E_LOW_PR= 6694.4 ,
  N_T_LOW_PR= -2.76 ,
  A_TROE= 0.562 ,
  T1_TROE= 5836.0 ,
  T2_TROE= 8552.0 ,
  T3_TROE= 91.0 ,
  SPEC_ID_NU= 'CH2','H' , 'CH3' ,
  NU= -1.0,-1.0 , 1.0 /

```

For all fall-off reactions, `REACTYPE` need to be specified as shown above. The low-pressure reaction rate coefficient is specified using `A_LOW_PR`, `N_T_LOW_PR`, and `E_LOW_PR`, similar to Eq. (13.22). In the case of Troe reactions, the fall-off blending factor is calculated using `A_TROE`, `T1_TROE`, `T2_TROE`, and `T3_TROE`. For details, please refer to the FDS Tech Guide [3].

13.3.6 Catalysts

Some reactions require the presence of a specific molecule to stabilize the reaction rather than any species as in a third body reaction. An example is given below:



A catalyst can be specified by listing the catalyst species twice in `SPEC_ID_NU` and giving the `NU` values for each side of the equation. The reaction will move forward in FDS using the net `NU` and the default value for `NU_S` will be the left-hand side stoichiometric coefficient. For a reverse reaction, the value or `NU_S` for the catalyst will be the right-hand side stoichiometric coefficient.

13.3.7 Chemical Time Integration

The set of ordinary differential equations (ODEs) for the combustion reactions are solved over the course of a time step using one of a variety of solvers. The solver is specified via the character string `ODE_SOLVER` on the `COMB` line. The chosen solver governs all reactions. If all reactions are fast, i.e. occur instantaneously following the mixing process, then the default solver is `'EXPLICIT EULER'`. If any of the reactions are finite-rate, then the default solver is `'RK2 RICHARDSON'`, a fourth-order scheme generated from three second-order

Runge-Kutta steps with Richardson extrapolation (the third step provides a means of error control; details are provided in the of the FDS Tech Guide [3]). Two other schemes are available—'RK2' and 'RK3', second and third-order Runge-Kutta schemes without Richardson extrapolation.

An important consideration for the time integration of chemical reactions is the number of iterations the ODE solver takes for a given FDS time step. For infinitely fast chemistry, the number of iterations is 1. For finite rate chemical reactions, the number of iterations can vary between 1 and the lesser of `MAX_CHEMISTRY_SUBSTEPS` (default 20) and that needed to satisfy the error tolerance of the ODE solver, `RICHARDSON_ERROR_TOLERANCE`. If the maximum value is reached, you can increase the number of iterations or decrease the error tolerance constraint of the integrator. You may also just fix the number of iterations by setting `N_FIXED_CHEMISTRY_SUBSTEPS`. Its default value is -1, meaning that the number of time steps is determined by the ODE solver. These various numerical parameters are all set on the `COMB` line, and they pertain to all reactions.

If you have multiple reactions with widely ranging time-scales, you might want to specify `CHECK_REALIZABILITY` to be `T` on the `COMB` line, which forces the program to issue a warning if the species mass fractions do not remain between 0 and 1 during the integration scheme.

To solve chemistry using a detailed chemical mechanism (see Sec. 13.3.2), it is recommended to use the CVODE ODE solver from Sundials [25]. The following input line configures FDS to use CVODE for chemistry calculations:

```
&COMB ODE_SOLVER='CVODE' /
```

Internally, CVODE assigns default values to several key parameters, as listed below:

- `FINITE_RATE_MIN_TEMP=300 °C`
- `ODE_MIN_ATOL=1E-10`
- `ODE_REL_ERROR=1E-6`
- `ZZ_MIN_GLOBAL=1E-10`
- `EQUIV_RATIO_CHECK=F`
- `MIN_EQUIV_RATIO=0.0`
- `MAX_EQUIV_RATIO=20.0`
- `DO_CHEM_LOAD_BALANCE=T`

The parameter `FINITE_RATE_MIN_TEMP` defines the minimum temperature (in °C) above which chemistry calculation will be performed. CVODE allows specification of relative and absolute tolerances at the species level using the `ODE_REL_ERROR` and `ODE_ABS_ERROR` parameters in the `SPEC` input line. These tolerances can also be set globally in the `COMB` input line, with species-level settings taking precedence. Currently, CVODE does not allow relative tolerance at the species level. The minimum concentration of a species is determined as the product of `ODE_REL_ERROR` and `ZZ_MIN_GLOBAL`. Concentrations below this threshold are treated as zero.

Additional optional parameters include `EQUIV_RATIO_CHECK`, `MIN_EQUIV_RATIO`, and `MAX_EQUIV_RATIO`. When `EQUIV_RATIO_CHECK` is enabled (set to true), the chemistry calculation is performed only for those cells for which equivalence ratio is within the specified `MIN_EQUIV_RATIO` and `MAX_EQUIV_RATIO` limits, reducing computational time. Enabling `DO_CHEM_LOAD_BALANCE` significantly accelerates chemistry calculations by distributing the computational load evenly across all MPI processes. User can modify the default values of any or all of these parameters as needed using the following line in the FDS input file:

```
&COMB
  ODE_SOLVER='CVODE'
  FINITE_RATE_MIN_TEMP=300,
```

```
ZZ_MIN_GLOBAL=1E-15,  
ODE_MIN_ATOL=1E-12,  
ODE_REL_ERROR=1E-6,  
EQUIV_RATIO_CHECK=T,  
MIN_EQUIV_RATIO=0.0,  
MAX_EQUIV_RATIO=20.0,  
DO_CHEM_LOAD_BALANCE=T /
```

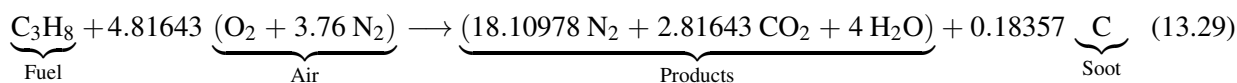
13.4 Aerosol Deposition

It is possible within FDS to model the deposition of smoke and aerosols onto solid surfaces. The aerosol deposition model is invoked by defining a species with the parameter `AEROSOL=T` on the `SPEC` line along with the parameters `DENSITY_SOLID`, `CONDUCTIVITY_SOLID`, and `MEAN_DIAMETER`. By default, with `AEROSOL=T`, FDS will compute all of the aerosol deposition mechanisms discussed in the Technical Reference Guide [3]. For diagnostic purposes, each surface deposition mechanism can be selectively disabled by using the logical parameters `GRAVITATIONAL_DEPOSITION`, `THERMOPHORETIC_DEPOSITION`, and `TURBULENT_DEPOSITION`. All surface deposition can be disabled by the logical parameter `DEPOSITION`. In the gas phase, aerosol transport is affected by gravity and temperature gradients. These effects can be selectively disabled with `GRAVITATIONAL_SETTLING` and `THERMOPHORETIC_SETTLING`. All the deposition parameters are on the `MISC` line. The deposition velocity at the wall can be output using the solid phase output `QUANTITY` called `'DEPOSITION VELOCITY'`.

The parameter `THERMOPHORETIC_DIAMETER` can be used to define a particle diameter to use in lieu of the `MEAN_DIAMETER` when computing the thermophoretic force. This may be appropriate for flaky or string like aerosol particles when the thermophoretic force can operate on each of the primary particles composing the larger aerosol. The default value is $0.03\ \mu\text{m}$ which represents a typical soot primary particle diameter. If this value is set to a negative number, the `MEAN_DIAMETER` will be used (taken as the bin diameter if agglomeration is being modeled) for thermophoresis.

13.4.1 Example Case: Soot Deposition from a Propane Flame

The `propane_flame_deposition` example shows how to define a reaction that invokes the aerosol deposition model in FDS. The fuel is propane with a specified soot yield of 0.05. Note that this is a fabricated soot yield that is used only for demonstration and verification purposes. The reaction is given by:



Note that the stoichiometric coefficient for soot ensures that the mass of soot produced is 0.0544 times the mass of fuel consumed. This example uses the lumped species formulation to minimize the number of scalar transport equations that need to be solved. Note that for soot to deposit it must be explicitly tracked by defining `AEROSOL=T` on the `SPEC` line.

```
&SPEC ID = 'PROPANE' /
&SPEC ID = 'OXYGEN',          LUMPED_COMPONENT_ONLY = T /
&SPEC ID = 'NITROGEN',        LUMPED_COMPONENT_ONLY = T /
&SPEC ID = 'WATER VAPOR',     LUMPED_COMPONENT_ONLY = T /
&SPEC ID = 'CARBON DIOXIDE',  LUMPED_COMPONENT_ONLY = T /
&SPEC ID = 'SOOT',            AEROSOL = T /
```

If Eq. (13.29) is properly balanced, you can directly use the stoichiometric coefficients of the primitive species to define the lumped species:

```
&SPEC ID = 'AIR', SPEC_ID = 'NITROGEN', 'OXYGEN',
      VOLUME_FRACTION = 3.76, 1., BACKGROUND = T /
&SPEC ID = 'PRODUCTS', SPEC_ID = 'NITROGEN', 'CARBON DIOXIDE', 'WATER VAPOR',
      VOLUME_FRACTION = 18.10978, 2.81643, 4.0 /
```

The heat of combustion for propane is found in the SFPE Handbook [18].

```
&REAC FUEL = 'PROPANE', HEAT_OF_COMBUSTION=44715.,
      SPEC_ID_NU = 'PROPANE', 'AIR', 'PRODUCTS', 'SOOT',
      NU=-1.,-4.81643,1.,0.18357/
```

Note: The sign of `NU` corresponds to whether that species is consumed (-) or produced (+). Figure 13.6 shows the soot surface deposition on the wall. This boundary quantity is given by the input below.

```
&BNDF QUANTITY='SURFACE DEPOSITION', SPEC_ID='SOOT' /
```



Figure 13.6: Wall soot deposition for the propane_flame_deposition test case.

13.4.2 Soot Surface Oxidation

If soot deposition is being computed, the soot present on surfaces can undergo oxidation if the local temperature is high enough and oxygen is present. Soot oxidation is enabled with the logical flag `SOOT_OXIDATION` on the `MISC` input. Oxidation is computed as a single step, Arrhenius reaction using the soot surface density and the local oxygen mole fraction. The reaction uses a rate constant of 4.7×10^{12} and an activation energy of 211 kJ/mol. These values are based upon work performed by Hartman, Beyler, Riahi, and Beyler [27].

Soot oxidation cannot be used with simple chemistry; you must use primitive species. See the sample case `soot_oxidation_wall` in the Verification Guide.

13.5 Aerosol Agglomeration

Aerosol particles will, over time, collide with one another and stick together to form larger particles. This process is called agglomeration. The agglomeration model in FDS is automatically invoked by defining an aerosol species with size bins. The agglomeration model can be disabled the logical flag `AGGLOMERATION` on the `MISC` input. Particle bins are defined by giving a minimum particle diameter, `MIN_DIAMETER` in m, a

maximum particle diameter, `MAX_DIAMETER` in m, and the desired number of size bins, `N_BINS`, on the `SPEC` line for an aerosol species. Note that if multiple aerosol species are defined, the agglomeration model will agglomerate each species independently, that is particles of different species do not agglomerate together into a larger mixed-species particle. The following input defines a soot species as an aerosol with 10 bins between 1 and 100 μm . Each bin will be created as its own lumped species with names `SOOT_1` through `SOOT_10`. A summary of the particle bins will be written to the `CHID.out` file.

```
&SPEC ID='SOOT', AEROSOL=T,MIN_DIAMETER=1.E-6,MAX_DIAMETER=100.E-6,
      N_BINS=10,LUMPED_COMPONENT_ONLY=T/
```

It may be desirable to combine aerosol agglomeration with chemical reaction. That is, one or more of the aerosol species may participate in a chemical reaction, either as a reactant or product. Keep in mind that tracked species for the aerosol distribution are appended with the bin number, as stated above. Within FDS, it is these subspecies that are being tracked and you must explicitly specify the stoichiometry of the subspecies within the reaction (see Sec. 13.2). Below is an example for combustion of *n*-heptane generating soot as an aerosol with five size bins.

```
&SPEC ID='N-HEPTANE' /
&SPEC ID='NITROGEN',LUMPED_COMPONENT_ONLY=T/
&SPEC ID='OXYGEN',LUMPED_COMPONENT_ONLY=T/
&SPEC ID='CARBON DIOXIDE',LUMPED_COMPONENT_ONLY=T/
&SPEC ID='CARBON MONOXIDE',LUMPED_COMPONENT_ONLY=T/
&SPEC ID='WATER VAPOR',LUMPED_COMPONENT_ONLY=T/
&SPEC ID='SOOT',LUMPED_COMPONENT_ONLY=T,
      AEROSOL=T,N_BINS=5,
      MIN_DIAMETER=0.023,MAX_DIAMETER=11.5/

&SPEC ID='AIR',BACKGROUND=T,
      SPEC_ID='NITROGEN','OXYGEN','CARBON DIOXIDE','WATER VAPOR',
      VOLUME_FRACTION=40.94283132,10.85911,0.020216389,0.497368688/
&SPEC ID='PRODUCTS',
      SPEC_ID='NITROGEN','CARBON DIOXIDE','WATER VAPOR','CARBON MONOXIDE',
      VOLUME_FRACTION=40.94283132,6.868439402,8.490526567,0.028618803/

&REAC FUEL='N-HEPTANE',
      SPEC_ID_NU='N-HEPTANE',
      'AIR',
      'PRODUCTS',
      'SOOT_1',
      'SOOT_2',
      'SOOT_3',
      'SOOT_4',
      'SOOT_5',
      NU=-1,-1,1,0.035837089,0.030715589,0.026822578,0.023205152,0.02026202,
      RADIATIVE_FRACTION=0.44 /
```

13.6 Aerosol Scrubbing

Water sprays can remove aerosols from the air. For example, rain can lower the concentrations pollen and other particulates in the air. This behavior can be modeled with FDS by setting `AEROSOL_SCRUBBING=T` on `MISC`. Setting this turns on an aerosol scrubbing model that removes aerosols from the gas phase. Once removed the mass is no longer tracked; i.e. the scrubbing model simply acts as a sink for aerosols.

13.7 Vapor Condensation

If a species with liquid properties, e.g., `WATER VAPOR`, is defined with `AEROSOL=T`, then FDS will predict condensation for that species. FDS creates a second species called `ID_COND`. For example if the species is `WATER VAPOR`, then the second species will be called `WATER VAPOR_COND`. This second species will be an aerosol species that tracks the condensed phase. The particle diameter for the condensed vapor is given by `MEAN_DIAMETER`, the condensed species specific heat will use the `SPECIFIC_HEAT_LIQUID` liquid values, and the particle density will use `DENSITY_LIQUID` rather than `DENSITY_SOLID`. The initial mass fraction of condensed phase can be defined with `MASS_FRACTION_COND_0`. Radiative properties should be assigned using `REAL_REFRACTIVE_INDEX` and `COMPLEX_REFRACTIVE_INDEX` unless the species is either `WATER VAPOR` or a predefined fuel species (these are the organic fuels listed in Table 13.1 that have liquid properties predefined, see Table A.1 or Table A.2). For example, the input below defines `WATER VAPOR` as a condensable species with an initial vapor mass fraction of 0.1 and an initial condensate mass fraction of 0.05.

The condensation routine requires the presence of nucleation sites to start the condensation process. If condensed gas is already present, the number of condensate particles is used. Otherwise, FDS assumes a background concentration of fine aerosols (pollen, bacterial spores, etc.). The default value is 10,000,000 nucleation sites per cubic meter. This value can be changed by setting `NUCLEATION_SITES` on `MISC`.

```
&SPEC ID='WATER VAPOR', AEROSOL=T,MEAN_DIAMETER=1.E-6,  
      MASS_FRRACTION_0=0.1,MASS_FRACTION_COND=0.05/
```

Condensation can occur in both gas cells and on wall cells. The heat transfer to a wall due to condensation can be output with the `QUANTITY` of `CONDENSATION HEAT FLUX`. The amount of vapor that condenses can be output with the `QUANTITY` of `SURFACE DEPOSITION` using the condensed phase `SPEC_ID`.

Chapter 14

Radiation

14.1 Basic Radiation Parameters: The `RADI` Namelist Group

`RADI` is the namelist group that contains parameters related to the radiation solver. There can be only one `RADI` line in the input file.

An important quantity in fire science is the fraction of the fire's heat release rate released in the form of thermal radiation, commonly referred to as the *radiative fraction*, symbolically denoted χ_r . It is a function of the fire size, flame temperature, and the chemical composition of the fuel and combustion products. The flame temperature, as opposed to the average cell temperature, is not reliably calculated in a large scale fire simulation because the flame sheet is not well-resolved on a relatively coarse numerical grid. Thus, the source term in the radiation transport equation (RTE), because of its T^4 dependence, cannot be reliably calculated. As a practical alternative, the parameter `RADIATIVE_FRACTION` on the `REAC` line allows you to set a lower bound on the fraction of the total energy due to combustion that is released in the form of thermal radiation, i.e. `RADIATIVE_FRACTION` only applies to gas cells with combustion and not to gas in a hot upper layer where the cell temperature can reliably be used in the RTE. By default, the `RADIATIVE_FRACTION` is set to a specific value that is based on the reaction's `FUEL` for an LES calculation, and it is set to zero for DNS, in which case the amount of energy radiated by the fire is predicted rather than prescribed. Table 14.1 lists the default radiative fraction for some common pure fuels. Many of these values are based on measurements performed on relatively small flames by Tewarson [18]. These values may change with increasing fire size; thus, the measurements cited by Beyler [28] may be more appropriate for larger fires. If in doubt, select the value that is appropriate for your fire. There is no single value of radiative fraction for a given fuel.

There are four ways of treating thermal radiation in FDS, as explained in the following sections.

14.1.1 Radiation Option 1. No Radiation Transport

It is possible to turn off the RTE solver (saving roughly 20 % in CPU time) by adding the statement `RADIATION=F` to the `RADI` line. If burning is taking place and radiation is turned off, then the total heat release rate is reduced by the `RADIATIVE_FRACTION`, which is an input on the `REAC` line. Default values for various pure fuels are listed in Table 14.1. With `RADIATION` set to `F`, the radiated energy from a fire simply disappears from the computational domain. For fire simulations or any scenario with temperature increases of hundreds of degrees Celsius, it is not recommended that you turn off the radiation transport. This feature is used mainly for diagnostic purposes or when the changes in temperature are relatively small.

Table 14.1: Default radiative fraction for some common fuels.

Species	χ_r	Reference
ACETONE	0.27	[18]
ACETYLENE	0.49	[18]
BENZENE	0.60	[18]
BUTANE	0.31	[18]
DODECANE	0.40	[28]
ETHANE	0.25	[18]
ETHANOL	0.25	[18]
ETHYLENE	0.34	[18]
HYDROGEN	0.20	[28]
ISOPROPANOL	0.29	[18]
METHANE	0.20	[28]
METHANOL	0.21	[29]
N-DECANE	0.40	[28]
N-HEPTANE	0.40	[28]
N-HEXANE	0.40	[28]
N-OCTANE	0.40	[28]
N-PENTANE	0.40	[28]
PROPANE	0.29	[18]
PROPYLENE	0.37	[18]
TOLUENE	0.40	[28]
All other species	0.35	

14.1.2 Radiation Option 2. Optically-Thin Limit; Specified Radiative Fraction

There are some fire scenarios where you might want to set the absorption coefficient in the RTE, κ , to zero by setting `OPTICALLY_THIN` to `T` on the `RADI` line, in which case re-absorption of thermal radiation by all gases is neglected. Essentially, the fire radiates the user-specified `RADIATIVE_FRACTION` of energy, and this energy is transported to the domain boundaries without being reabsorbed by colder gases. This is not something you want to do for a compartment fire scenario, where absorption and emission of thermal radiation by hot and cold smoke is an important consideration. Rather, you might want to choose this option for scenarios where you have a fire outside of a compartment or in a relatively large open space and you want to explicitly dictate the net radiation energy that is transported to targets and solid boundaries. For example, the radiative fraction of very large hydrocarbon fuel fires can decrease with increasing diameter because of the re-absorption of radiated energy by the smoke and combustion products. This is why a large oil fire appears to be comprised mostly of smoke, and this smoke shields external objects from the thermal radiation. Predicting this phenomenon is difficult and can be grid-dependent. In such cases, you may want to just specify the `RADIATIVE_FRACTION` on the `REAC` line and then `OPTICALLY_THIN=T` on the `RADI` line, in which case the fire will radiate the given fraction of energy and FDS will not attempt to calculate the re-absorption of this energy by the combustion products or surrounding gases.

14.1.3 Radiation Option 3. Optically-Thick; Specified Radiative Fraction (LES Default)

In its normal operation, the RTE transfers energy from hot, emitting gases, like flames, to colder, absorbing gases like water vapor or soot particulate. The absorption coefficient, κ , computed using RadCal, governs both the emission and absorption of thermal radiation. Because flame temperatures are not well-resolved for typically large-scale fire simulations, the source term in the RTE is adjusted in grid cells for which the radiative fraction, χ_r , times the local heat release rate per unit volume, \dot{q}''' , is greater than 1 kW/m^3

$$\chi_r \dot{q}''' > 1 \text{ kW/m}^3 \quad (14.1)$$

The adjustment ensures that the net radiative emission from the combusting region (i.e. the fire) is the specified `RADIATIVE_FRACTION` multiplied by the total combustion energy generated in this region. Elsewhere, hot and cold gases emit and absorb thermal radiation according to their bulk temperature and radiative properties, in particular the absorption coefficient, κ .

The net radiative loss from the computational domain is reported as a function of time in the column `Q_RAD` in the output file `CHID_hrr.csv`. The absolute value of `Q_RAD` divided by the total heat release rate, `HRR`, is usually not exactly equal to the specified `RADIATIVE_FRACTION`. The reason for this is that the specified radiative fraction of the fire's energy can be reabsorbed by colder combustion products such as smoke and water vapor, thereby decreasing the absolute value of `Q_RAD`. Or, hot layer smoke and combustion products can heat up and emit thermal radiation, adding to the absolute value of `Q_RAD`.

The correction factor that is applied to the RTE source term in the region defined by Eq. (14.1) by default is bound between `C_MIN` and `C_MAX` (default 0.1 and 100). You can change the default behavior of the correction as follows. First, you can force the RTE source term to be modified in all grid cells by changing the 1 in Eq. (14.1) to -1 via `QR_CLIP` on the `RADI` line, in which case the solver will apply the radiative fraction to the entire domain, not just the cells where combustion occurs. This will essentially force the net radiative loss from the entire domain to obey the `RADIATIVE_FRACTION`. Second, you can specify that the `RADIATIVE_FRACTION` serves only as a *minimum* value by changing the lower limit of the correction factor, `C_MIN`, from its default value of 0.1 to 1.0 on the `RADI` line. This change recognizes the fact that a fire's radiative fraction might change over the course of a simulation, where it initially exhibits the specified `RADIATIVE_FRACTION` but this might change as the fire grows larger and engulfs a significant fraction of the compartment.

The time-varying correction factor can be output using a device with the `QUANTITY` called '`RTE_SOURCE_CORRECTION_FACTOR`'. Since this is a global value, the position of the device does not matter.

14.1.4 Radiation Option 4. Optically-Thick; Unspecified Radiative Fraction (DNS Default)

If it can be assumed that the temperature field is well-resolved and the radiation absorption coefficient and RTE source term can be calculated without correction, then `RADIATIVE_FRACTION` ought to be set to zero. This is the default for a DNS (Direct Numerical Simulation), and you can set it for LES calculations that are highly resolved, like where the grid cells are a few millimeters in size. Of course, do a grid resolution study to determine if the resulting radiative fraction, i.e. $-\text{Q_RAD}/\text{HRR}$, is grid independent.

14.2 Spatial and Temporal Resolution of the Radiation Transport Solver

There are several ways to improve the spatial and temporal accuracy of the discrete radiation transport equation (RTE), but each typically increases the computation time. The spatial accuracy can be improved by increasing the number of angles from the default 100 with the integer parameter `NUMBER_RADIATION_ANGLES`. A good way to determine if you need better spatial resolution is to add slice (`SLCF`) files of the quantity '`INTEGRATED_INTENSITY`'. Typically, you see high values of this quantity near sources of heat,

and these values decrease farther away. Ideally, you should see a somewhat smooth and circular pattern far from the heat source, but because of the finite number of solid angles along which the radiation intensity is tracked, you will see in the far field a star-like pattern that is not physical. If there are no important “targets” far from the heat source, this nonphysical pattern can be ignored, but if there are important targets, then increase the number of angles until you see a relatively smooth pattern over the region of interest.

The requirements for accurate angular resolution have been studied by Kim and Trouvé [30]. They suggest the number of solid angles needed for accurate resolution ($NRA_{critical}$) to a target at a separation distance, H , with emission source surface area, S , is given by

$$NRA_{critical} = N_{\Omega} \times \frac{4\pi H^2}{S} \quad (14.2)$$

where N_{Ω} should be between 5 and 10.

The frequency of calls to the radiation solver can be changed from every 3 time steps with an integer called `TIME_STEP_INCREMENT`. The increment over which the angles are updated can be reduced from 5 with the integer called `ANGLE_INCREMENT`. If `TIME_STEP_INCREMENT` and `ANGLE_INCREMENT` are both set to 1, the radiation field is completely updated in a single time step, but the cost of the calculation increases significantly. By default, the radiation transport equation is fully updated every 15 time steps. Given the relatively small time steps dictated by the CFL constraint, 15 time steps is usually still a relatively short interval of time. Increasing the temporal resolution of the radiation solver rarely adds to the overall accuracy of the calculation. Spatial resolution is far more important.

If you are using multiple meshes, the radiation solver cannot transfer energy from mesh to mesh within a single time step. If you notice an obvious delay in the propagation of radiative intensity from one mesh to another, you can increase the number of times the radiative intensity is updated within a single time step using `RADIATION_ITERATIONS`, which is 1 by default. You rarely need to do this, unless it is obvious from the animation of various slice files.

The radiation solver is called before the start of the calculation to establish the radiation field in the event that you specify something to have a non-ambient temperature initially. By default, the radiation and wall boundary routines are iterated three times to establish thermal equilibrium. To change the number of iterations, set `INITIAL_RADIATION_ITERATIONS` on the `RADI` line.

14.3 Absorption Coefficient of Gases and Soot

By default FDS employs a gray gas model for the radiation absorption coefficient. Other options are the wide band model (often called Box Model in the radiation literature), and the Weighted Sum of Gray Gases (WSGG) model. The models are discussed in detail in the FDS Technical Reference Guide [3]. The radiation properties of most common gases involved in combustion processes (water vapor, carbon dioxide, carbon monoxide, fuel) and soot particles are automatically taken into account if the simulation involves combustion. In simulations with no combustion nor radiating species, it is possible to use a constant absorption coefficient by specifying `KAPPA0` on the `RADI` line.

You can output the gray absorption coefficient using `QUANTITY='ABSORPTION COEFFICIENT'`. The Wide Band and WSGG models employ several values of the absorption coefficients for each location, and the `'ABSORPTION COEFFICIENT'` output quantity only shows one of them, making this output impractical for multi-band models.

If aluminum oxide is being modeled, the optical properties of soot can be replaced by the optical properties of aluminum oxide by setting `AEROSOL_AL2O3=T` on `MISC`. Any species with `RADCAL_ID='SOOT'` will be treated as aluminum oxide.

14.3.1 Gray Gas Model (default)

The absorption coefficient is a function of gas composition and temperature. In the beginning of the simulation, FDS calls the RadCal narrow band model several times to build a look-up table of temperature dependent, gray specific absorption coefficients of all gases present in the simulation plus soot. There are several considerations with regard to RadCal:

Path Length

For a given gas temperature and species composition, RadCal computes a single effective absorption coefficient that is independent of wavelength. To calculate this coefficient, a user-specified `PATH_LENGTH` (m) is needed. Its default value is 0.1 m. The choice of `PATH_LENGTH` can be based on the physical size of the fire, the compartment, or the overall computational domain, depending on the application. The default value has been chosen to capture accurately radiation heat transfer in and around the fire itself. A useful “rule of thumb” for this length scale is $4V/A$, where V is the volume of the region of interest and A is the encompassing surface area. This region might be the volume occupied by the fire itself or a flashed-over compartment. Alternatively, if the application involves calculating the heat flux to distant targets, a more appropriate `PATH_LENGTH` might be the distance from the fire to the target. A sensitivity analysis should be done in any case to determine how the chosen `PATH_LENGTH` affects the predicted values.

Fuel Species

The original version of RadCal included only absorption data for methane, which was used as a surrogate for any fuel. However, more fuel species have been added to RadCal. The current list of fuels includes: METHANE, ETHYLENE, ETHANE, PROPANE, N-HEPTANE, METHANOL, TOLUENE, PROPYLENE, and MMA. These species are in addition to the RadCal species of: CARBON DIOXIDE, CARBON MONOXIDE, WATER VAPOR, and SOOT.

14.3.2 Wide Band Model (Box Model)

To use the optional wide band (box) model, set `WIDE_BAND_MODEL=T` on the `RADI` line. It is recommended that this option only be used when the fuel is relatively non-sooting because it adds significantly to the cost of the calculation. Do not specify a `RADIATIVE_FRACTION` because it will not be used.

When using the wide band model, the electromagnetic spectrum is divided in a relatively small number of bands, and within each band, the absorption coefficient is assumed constant, thus the name ‘box’. These values are calculated with RadCal. The default operation with `WIDE_BAND_MODEL=T` assumes six bands, and the sets of band limits have been pre-defined for the six bands of water, carbon dioxide, carbon monoxide, and a handful of fuel species. These fuels are ‘ETHANE’, ‘ETHYLENE’, ‘METHANE’, ‘METHANOL’, ‘N-HEPTANE’, ‘PROPANE’, ‘PROPYLENE’, and ‘TOLUENE’.

It is possible to set your own band limits by specifying `BAND_LIMITS` on the `RADI` line. The limits should be given in ascending order, in units of microns (μm). The maximum number of bands is 9, in which case you would specify 10 real numbers separated by commas.

14.3.3 Weighted Sum of Gray Gases (WSGG) Model (Experimental)

To activate the WSGG model, set `WSGG_MODEL=T` on the `RADI` line. The WSGG model calculates the absorption coefficient as a weighted sum of four gray coefficients, each being a function of gas composition (water vapor, carbon dioxide and soot) and temperature. Other gases are ignored. Do not set `BAND_LIMITS`.

Note that the WSGG model is an experimental feature. It has very limited verification and validation that is focused mainly on simple fuels and geometric configurations.

14.4 Radiative Absorption and Scattering by Particles

The absorption and scattering of thermal radiation by Lagrangian particles is included in the radiation transport equation. The radiative properties can be given by specifying the components of the material refractive index on the corresponding `PART` line, using keywords `REAL_REFRACTIVE_INDEX` and `COMPLEX_REFRACTIVE_INDEX`. Alternatively, wavelength dependent values of these two quantities can be tabulated in a `TABLE` and called using the `RADIATIVE_PROPERTY_TABLE`. More details can be found in Sec. 15.3.2.

The radiative properties of the water and fuel particles (droplets) are determined automatically. For fuel, the properties of heptane are assumed. The heptane values can be overridden by specifying them on the `PART` line.

Other parameters affecting the computations of particle-radiation interaction are listed here. `RADTMP` is the assumed radiative source temperature. It is used in the spectral weighting during the computation of the mean scattering and absorption cross sections. The default is 900 °C. `NMIEANG` is the number of angles in the numerical integration of the Mie-phase function. Increasing `NMIEANG` improves the accuracy of the radiative properties of water droplets. The cost of the better accuracy is seen in the initialization phase, not during the actual simulation. The default value for `NMIEANG` is 15. For each class of particles, the Mie coefficients are calculated for a wide range of droplet diameters to ensure that the all possible runtime situations can be covered. To speed up the initialization phase, the range of diameters can be limited by parameters `MIE_MINIMUM_DIAMETER` and `MIE_MAXIMUM_DIAMETER`. Also, the size of the Mie coefficient tables can be specified using `MIE_NDG` parameter.

14.5 Other Considerations

Multi-fuel radiative fraction If multiple fuels are present, e.g., one fuel for cardboard and one fuel for polystyrene combustion, then the radiant fraction will be computed locally as a reaction-weighted value, similar to the approach described by Gupta et al. [31]. For example, if the two fuels are 20% and 40% radiative fraction with, respectively, 80% and 20% of the heat in a grid cell, χ_r would be $0.8 \times 0.2 + 0.2 \times 0.4 = 0.24$. Thus, χ_r will vary in space and time. The value of the multi-fuel radiative fraction can be output using the output `QUANTITY` of `CHI_R`.

Time variation of radiative fraction Even for a single fuel species the global flame radiative fraction may depend on other parameters of the problem like global equivalent ratio. If a time variation of the radiative fraction is necessary, it may be added through a ramp function, `RAMP_CHI_R`, on the `REAC` line. The results of running the `ramp_chi_r` test case containing the `RAMP` given below is shown in Fig. 14.1.

```
&REAC FUEL='METHANE', RADIATIVE_FRACTION=1.0, RAMP_CHI_R='CHI_R RAMP' /
&RAMP ID='CHI_R RAMP', T= 0.0, F=0.238 /
&RAMP ID='CHI_R RAMP', T= 2.0, F=0.238 /
&RAMP ID='CHI_R RAMP', T= 4.0, F=0.182 /
&RAMP ID='CHI_R RAMP', T= 6.0, F=0.158 /
&RAMP ID='CHI_R RAMP', T= 8.0, F=0.140 /
&RAMP ID='CHI_R RAMP', T= 10.0, F=0.140 /
```

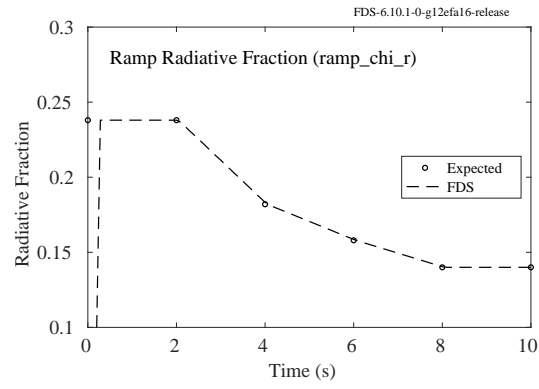



Figure 14.1: Results of the `ramp_chi_r` test case.

Chapter 15

Particles and Droplets

Lagrangian particles can be used to represent a wide variety of objects that are too small to resolve on the numerical grid. FDS considers three major classes of Lagrangian particles: massless tracers, liquid droplets, and everything else. The parameters describing particles are found on the `PART` line.

15.1 Basics

Properties of different types of Lagrangian particles are designated via the `PART` namelist group. Once a particular type of particle has been described using a `PART` line, then the name of that particle type is invoked elsewhere in the input file via the parameter `PART_ID`. There are no reserved `PART_IDS` – all must be defined. For example, an input file may have several `PART` lines that include the properties of different types of Lagrangian particles:

```
&PART ID='my smoke',... /  
&PART ID='my water',... /
```

Particles are introduced into the calculation in several different ways: they may be introduced via a sprinkler or nozzle (liquid droplets are usually introduced this way), they may be introduced at a blowing vent or burning surface (mass tracer particles or particles representing embers are usually introduced this way), and they may be introduced randomly or at fixed points within a designated volume (solid particles that represent subgrid-scale objects are usually introduced this way). Details are found below.

The way to describe particles depends on the type. If you simply want massless tracers, specify `MASSLESS=T` on the `PART` line. If you specify a `SPEC_ID`, then FDS automatically assumes that you want relatively small, thermally-thin evaporating liquid droplets. For any other type of particle, such as particles that represent subgrid-scale objects, like office clutter or vegetation, you add a `SURF_ID` to the `PART` line. All of these different types of particles are described below.

Computational versus Physical Particles

In order to conserve mass and energy while minimizing computational cost, FDS *computational* particles usually represent a larger number of *physical* particles with the same geometric and thermo-physical properties. Within FDS, each particle is assigned a *weighting factor* representing the number of physical particles per computational particle. You can output the `'PARTICLE WEIGHTING FACTOR'` using a `DEVC` or visualize the `'PARTICLE WEIGHTING FACTOR'` in Smokeview.

15.2 Massless Particles

The simplest use of Lagrangian particles is for visualization, in which case the particles are considered massless tracers. In this case, the particles are defined via the line

```
&PART ID='tracers', MASSLESS=T, ... /
```

Note that if the particles are `MASSLESS`, it is not appropriate to color them according to any particular property. Particles are not colored by gas phase quantities, but rather by properties of the particle itself. For example, `'PARTICLE TEMPERATURE'` for a non-massless particle refers to the temperature of the particle itself rather than the local gas temperature. Also note that if `MASSLESS=T`, the `SAMPLING_FACTOR` (Sec. 22.9) is set to 1 unless you say otherwise, which would be pointless since `MASSLESS` particles are for visualization only.

Turbulent Dispersion (Massless Tracers)

Massless tracer particles may also be useful in modeling dispersion of a tracer gas that does not affect the mean flow field (passive scalar). The number density of the particles then may be translated into a local mass concentration. See how to output number concentration in Sec. 22.10.16.

To account for subgrid-scale turbulent motions, add `TURBULENT_DISPERSION=T` to the `PART` line. The particles will then undergo a random walk based on the subgrid diffusivity. As an example, see the `random_walk` test cases in the `WUI` directory of the verification suite.

Turbulent Dispersion (Massive Particles)

`TURBULENT_DISPERSION=T` may also be used with massive particles. In this case, a random walk model is not used; instead, the fluid velocity used in the drag calculation is augmented with a fluctuating component that is taken from an estimate of the subgrid kinetic energy. Details of the formulation are discussed in the FDS Technical Reference Guide [3].

15.3 Liquid Droplets

Lagrangian particles that are not just massless tracers fall into two main categories – liquid droplets or solid particles. This section describes the former; and the next, the latter.

To define an evaporating liquid droplet, you must specify the name of the gas species created by evaporation via the `SPEC_ID` on the `PART` line. For example,

```
&PART ID='my droplets', SPEC_ID='WATER VAPOR', ... /
```

By specifying a `SPEC_ID`, you are implicitly invoking the droplet evaporation model. Alternatively, if you specify a `SURF_ID`, you are designating a solid particle that behaves according to the given `SURF` line. Solid particles are described in Sec. 15.4.

By default liquid droplet evaporation uses the Ranz-Marshall correlation [32] with a B-number. Additional correlations are available by setting `EVAP_MODEL` on `PART`. The available options are:

- 'RANZ-MARSHALL NO B-NUMBER' uses the Ranz-Marshall correlation with no B-number.
- 'RANZ-MARSHALL B-NUMBER' uses Ranz-Marshall correlation along with an evaporating species B-number applied to both the Nusselt and Sherwood number correlations. This is the default value. This is the M0 model from Sazhin [33].
- 'RANZ-MARSHALL LEWIS B-NUMBER' is the default correlation with the Sherwood number correlation using a B-number based on the Lewis number computed at the film condition. This is the M1 model from Sazhin [33].
- 'RANZ-MARSHALL FLUX-LIMITED LEWIS NO B-NUMBER' is the previous model plus a flux limiting factor applied to the B-number. This is the M2 model from Sazhin [33].

15.3.1 Thermal Properties

The properties you need to supply for a liquid droplet depend on whether or not the `SPEC_ID` is listed in Appendix A, and if the column labeled “Liquid” is checked with a “Y”, meaning that its liquid properties are known. The following sub-sections provide instructions on what to do in various instances. Note that the `INITIAL_TEMPERATURE` (°C) of the liquid droplets is specified on the `PART` line. Its default value is the overall ambient temperature of the simulation, `TMPA`, from the `MISC` line.

Heat transfer and evaporation from liquid droplets to the gas uses correlations for the heat and mass transfer coefficients. These can instead be respectively specified using `HEAT_TRANSFER_COEFFICIENT_GAS` and `MASS_TRANSFER_COEFFICIENT` on the `PART` line.

Known Liquid/Gas Species

If the `SPEC_ID` is listed in Appendix A, and if the column labeled “Liquid” is checked with a “Y”, you need only list the following:

```
&SPEC ID='WATER VAPOR' /  
&PART ID='water droplets', SPEC_ID='WATER VAPOR', ... /
```

There might be other optional parameters listed on the `PART` line, but these are the ones that are absolutely necessary. In addition, if `SPEC_ID='WATER VAPOR'` the droplets are not only assigned the thermo-physical properties of water, but also the radiation absorption properties are set to that of water, and the droplets are colored blue in Smokeview.

Unknown Liquid/Gas Species

If the `SPEC_ID` is not listed in Appendix A with a “Y” in the “Liquid” column, you must provide all of the following properties of the liquid on the `SPEC` line:

- `DENSITY_LIQUID` (kg/m^3).
- `SPECIFIC_HEAT_LIQUID` ($\text{kJ}/(\text{kg K})$). If the specific heat of the liquid is a function of temperature, use `RAMP_CP_L` to provide the function $c_p(T)$.
- `VAPORIZATION_TEMPERATURE` Boiling temperature of the liquid, T_{boil} ($^{\circ}\text{C}$).
- `MELTING_TEMPERATURE` Melting (solidification) temperature of the liquid ($^{\circ}\text{C}$).
- `ENTHALPY_OF_FORMATION` The heat of formation of the gas (kJ/mol).
- `HEAT_OF_VAPORIZATION` Latent heat of vaporization of the liquid, h_v (kJ/kg).
- `H_V_REFERENCE_TEMPERATURE` The temperature corresponding to the `HEAT_OF_VAPORIZATION` ($^{\circ}\text{C}$).
- `VISCOSITY_LIQUID` The viscosity of the liquid ($\text{kg}/(\text{m s})$).
- `CONDUCTIVITY_LIQUID` The conductivity of the liquid ($\text{W}/(\text{m K})$).
- `BETA_LIQUID` The coefficient of thermal expansion of the liquid ($1/\text{K}$).

Note that FDS will adjust the liquid enthalpy so that the following relationship holds:

$$h_{\text{gas}}(T_{\text{boil}}) = h_{\text{liquid}}(T_{\text{boil}}) + h_v \quad (15.1)$$

Custom Liquid/Gas Species

Note that only one gas species can be used in the droplet evaporation model because the model does not consider a distillation curve for evaporation. However, custom properties can be assigned to a `SPEC`. In addition to the liquid properties mentioned above, see Sec. 12.1.3 on user-defined gas properties.

Fuel Liquid/Gas Species

If the liquid droplets burn, the `SPEC_ID` on the `PART` line should designate the `FUEL` on the `REAC` line. Depending on the combustion model, you may not need to create a separate `SPEC` line. Fuel droplets will be colored yellow by default in Smokeview and any resulting fuel vapor will burn according to the combustion model specified on the `REAC` line. The droplets evaporate into an equivalent amount of fuel vapor such that the resulting heat release rate (assuming complete combustion) is equal to the evaporation rate multiplied by the `HEAT_OF_COMBUSTION`. The `HEAT_OF_COMBUSTION` can be specified on the `PART` line, or the `REAC` line, or it will be calculated automatically by FDS. If it is specified on the `PART` line, the burning rate will be adjusted to account for the difference between the heats of combustion of this particular droplet and the other fuels in the model. If you let FDS calculate the heat of combustion automatically, run the model for a few time steps and look for the “Heat of Combustion” listed in the `CHID.out` file. Use this value to specify the flow rate from the nozzle to achieve your desired heat release rate.

If a spray nozzle is used to generate the fuel droplets, its characteristics are specified in the same way as those for a sprinkler. If the fuel species is listed in Appendix A and the column labeled “RadCal Surrogate” is not blank, then the droplets will be assigned fuel radiation absorption properties corresponding to the listed species.

Note that to limit the computational cost of sprinkler simulations, liquid droplets disappear when they hit the “floor” of the computational domain, regardless of whether it is solid or not. However, this may not be desired when using liquid fuels. To stop FDS from removing liquid droplets from the floor of the domain, add the phrase `POROUS_FLOOR=F` to the `MISC` line. An alternate solution is make sure the `OBST` the fuel is landing on is at least one cell thick.

In the example file (`Fires/spray_burner.fds`), heptane from two nozzles is sprayed downward into a steel pan. The flow rate is increased linearly so that the fire grows to 2 MW in 20 s, burns steadily for another 20 s, and then ramps down linearly in 20 s. The key input parameters are given here:

```
&REAC FUEL='N-HEPTANE',SOOT_YIELD=0.01 /

&DEVC ID='nozzle_1', XYZ=4.0,-.3,0.5, PROP_ID='nozzle', QUANTITY='TIME', SETPOINT=0. /
&DEVC ID='nozzle_2', XYZ=4.0,0.3,0.5, PROP_ID='nozzle', QUANTITY='TIME', SETPOINT=0. /

&PART ID='heptane droplets', SPEC_ID='N-HEPTANE',
      QUANTITIES(1:2)='PARTICLE DIAMETER','PARTICLE TEMPERATURE',
      DIAMETER=1000., SAMPLING_FACTOR=1 /

&PROP ID='nozzle', PART_ID='heptane droplets', HEAT_OF_COMBUSTION=44500.,
      FLOW_RATE=1.97, FLOW_RAMP='fuel', PARTICLE_VELOCITY=10.,
      SPRAY_ANGLE=0.,30., SMOKEVIEW_ID='nozzle' /

&RAMP ID='fuel', T= 0.0, F=0.0 /
&RAMP ID='fuel', T=20.0, F=1.0 /
&RAMP ID='fuel', T=40.0, F=1.0 /
&RAMP ID='fuel', T=60.0, F=0.0 /
```

Many of these parameters are self-explanatory. Note that a 2 MW fire is achieved via 2 nozzles flowing heptane at 1.96 L/min each:

$$2 \times 1.97 \frac{\text{L}}{\text{min}} \times \frac{1}{60} \frac{\text{min}}{\text{s}} \times 684 \frac{\text{kg}}{\text{m}^3} \times \frac{1}{1000} \frac{\text{m}^3}{\text{L}} \times 44500 \frac{\text{kJ}}{\text{kg}} = 2000 \text{ kW} \quad (15.2)$$

The parameter `HEAT_OF_COMBUSTION` over-rides that for the overall reaction scheme. Thus, if other droplets or solid objects have different heats of combustion, the effective burning rates are adjusted so that the total heat release rate is that which you expect. However, exercises like this ought to be conducted just to ensure that this is the case. The HRR curve for this example is given in Fig. 15.1.

Note also that this feature is subject to mesh dependence. If the mesh cells are too coarse, the evaporating fuel can be diluted to such a degree that it may not burn. Proper resolution depends on the type of fuel and the amount of fuel being ejected from the nozzle. Always test your burner at the resolution of your overall simulation.

15.3.2 Radiative Properties

The radiative properties of water and fuel droplets are determined automatically. For fuel, the properties of heptane are assumed. For other types of particles, the radiative properties can be given by specifying the components of the material refractive index on the corresponding `PART` line, using keywords `REAL_REFRACTIVE_INDEX` and `COMPLEX_REFRACTIVE_INDEX`. Alternatively, wavelength dependent values of these two quantities can be specified using a spectral property table (`TABL`) whose `ID` is indicated on the `PART` line via `RADIATIVE_PROPERTY_TABLE`. Each row of a spectral property table contains three real numbers: wavelength (μm), and the real and complex components of the refractive index. The real part of the refractive index should be a positive number. If it is greater than 10, the particles are treated as perfectly reflecting spheres. The complex part should be a non-negative number. Values less than 10^{-6} are treated

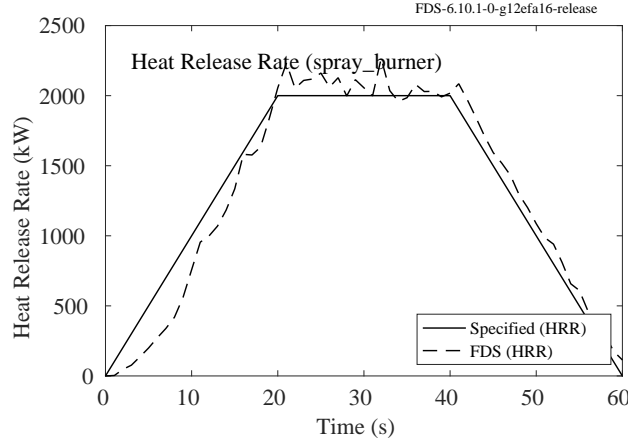


Figure 15.1: Heat Release Rate (HRR) of spray burner test.

as non-absorbing. Below is an example of the use of the spectral property table, listing the properties at wavelengths 1, 5 and 10 μm .

```
&PART ID='my particles',..., RADIATIVE_PROPERTY_TABLE='radtab' /
&TABL ID='radtab', TABLE_DATA= 1.0,1.33,0.0001 /
&TABL ID='radtab', TABLE_DATA= 5.0,1.33,0.002 /
&TABL ID='radtab', TABLE_DATA=10.0,1.33,0.001 /
```

For calculating the absorption of thermal radiation by particles, FDS uses a running average of particle temperature and density. The `RUNNING_AVERAGE_FACTOR` is set on the `PART` line, and its default value is 0.5 for liquid droplets and 0 for solid particles. The value of 0 indicates that no running average is used. Similarly, the parameter `RUNNING_AVERAGE_FACTOR_WALL`, also set on the `PART` line, controls the running average of droplet or particle mass on solid surfaces. Its defaults are the same as for `RUNNING_AVERAGE_FACTOR`.

15.3.3 Size Distribution

The size distribution of liquid droplets is specified using a cumulative volume fraction (CVF)¹ indicated by the character string `DISTRIBUTION` on the `PART` line. The default is `'ROSIN-RAMMLER-LOGNORMAL'`:

$$F(D) = \begin{cases} \frac{1}{\sqrt{2\pi}} \int_0^D \frac{1}{\sigma D'} \exp\left(-\frac{[\ln(D'/D_{v,0.5})]^2}{2\sigma^2}\right) dD' & (D \leq D_{v,0.5}) \\ 1 - \exp\left(-0.693 \left(\frac{D}{D_{v,0.5}}\right)^\gamma\right) & (D > D_{v,0.5}) \end{cases} \quad (15.3)$$

Alternatively, you can specify `'LOGNORMAL'` or `'ROSIN-RAMMLER'` alone rather than the combination of the two. Figure 15.2 displays the possible size distributions. Notice that the `'LOGNORMAL'` and `'ROSIN-RAMMLER'` distributions have undesirable attributes at opposite tails, which is why the combination of the two is commonly used. Figure 15.2 also shows a comparison between the prescribed distribution and the actual realized distribution of droplet sizes. The dashed lines show the measured droplet size distributions, while the solid lines show the prescribed sampling distributions. The sampled distributions are measured with the `HISTOGRAM` function. A sample size of 10000 droplets was used.

¹The CVF indicates the fraction of total mass carried by droplets less than the given diameter.

The median volumetric diameter, $D_{v,0.5}$, is specified via the parameter `DIAMETER` (μm) on the `PART` line. You must specify the `DIAMETER` in cases where the droplets evaporate (in which case you also need to specify a `SPEC_ID` to indicate the gas species generated by the evaporating droplets). The width of the log-normal distribution, σ , is specified with `SIGMA_D` on the `PART` line. The width of the Rosin-Rammler distribution, γ , is specified with `GAMMA_D` (default 2.4). Note that in the combined distribution, the parameter, σ , is calculated $\sigma = 2/(\sqrt{2\pi}(\ln 2) \gamma) = 1.15/\gamma$ which ensures that the two functions are smoothly joined at $D = D_{v,0.5}$. You can also add a value for `SIGMA_D` to the `PART` line if you want to over-ride this feature. The larger the value of γ , the narrower the droplet size is distributed about the median value.

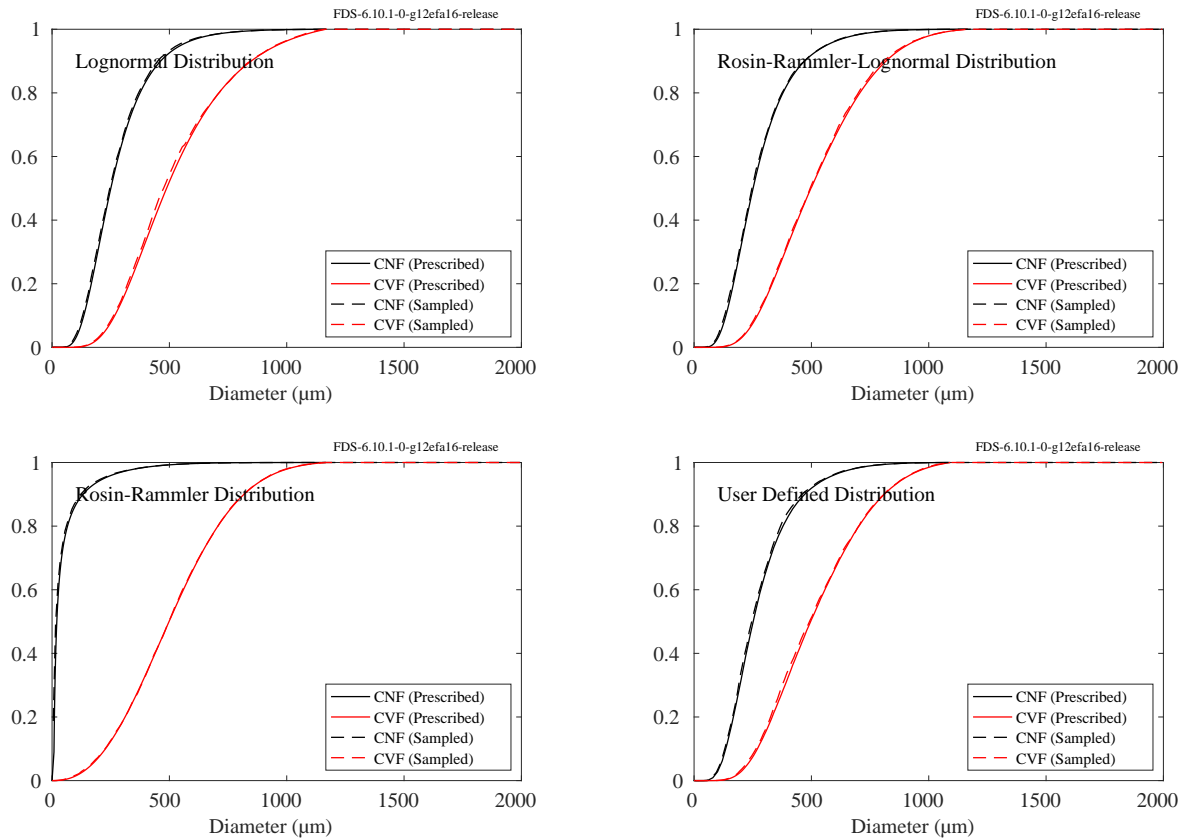


Figure 15.2: Droplet size distributions. The first three plots are based on a specified CVF (red curves) from which the CNF (black curves) is derived. The fourth plot (lower right) is an example of a specified CNF from which the CVF is derived. The solid lines show the prescribed sampling distribution, while the dashed lines show the actual sampled droplet size distribution, measured with the `HISTOGRAM` functionality.

You can specify your own cumulative number fraction (CNF)² by specifying a `CNF_RAMP_ID` on the `PART` line and including a `RAMP` that gives the CNF:

```
&PART ID='my droplets',..., CNF_RAMP_ID='my CNF' /
&RAMP ID='my CNF', T= 0.1, F=0.000000 /
&RAMP ID='my CNF', T= 200., F=0.000003 /
...
&RAMP ID='my CNF', T=2000., F=1.000000 /
```

²The CNF indicates the fraction of total droplets whose diameters are less than the given diameter.

Note that the `RAMP` variable `T` indicates the diameter and is given in micrometers. The fourth plot in Fig. 15.2 is an example of where the CNF is specified and the CVF is calculated from it. It is essentially the reverse of what is shown in the first plot, where the CVF is specified and the CNF is calculated from it.

As droplets are created in the simulation, their diameters are randomly chosen based on the given distribution. To avoid very small particle weights, the built-in distributions are clipped at the cumulative fractions of `CNF_CUTOFF` and $(1 - \text{CNF_CUTOFF})$. Note that `CNF_CUTOFF` is set on the `MISC` line. The default value of `CNF_CUTOFF` is 0.005. You can prevent excessively large droplets from being chosen by specifying a `MAXIMUM_DIAMETER`, which is assigned an extremely large value by default. The lower end of the size distribution is set using `MINIMUM_DIAMETER` whose default value is 0.005 times the value of `DIAMETER`. If a `CNF_RAMP_ID` is given, then the default value of `MINIMUM_DIAMETER` is set as the diameter of the ramp corresponding to `CNF_CUTOFF`. Note that an extremely small `MINIMUM_DIAMETER` might result in numerical issues if the weighted droplet mass of the `MINIMUM_DIAMETER` approaches the numerical precision of the gas mass in a cell. A warning will be written if there is a very low value ($<0.01 \mu\text{m}$) of `MINIMUM_DIAMETER`. Droplets are removed when their diameter decreases below `KILL_DIAMETER`. For a `MONODISPERSE` distribution, `KILL_DIAMETER` is set to 0.5 % of the mass of the droplet at its initial `DIAMETER`. Otherwise droplets are removed when the droplet mass is 0.5 % of the mass of the `MINIMUM_DIAMETER`. The droplet diameter range is divided into a series of bins³.

To prevent FDS from generating a distribution of droplets altogether, set `MONODISPERSE` to `T` on the `PART` line, in which case every droplet will be assigned the same `DIAMETER`.

If you set `CHECK_DISTRIBUTION=T` on the `PART` line, FDS will write out the cumulative distribution function for that particular particle class in a file called `CHID_PART_ID_cdf.csv`. If you do this, you might want to avoid spaces in the `ID` of the `PART` line.

15.3.4 Dense Clouds of Droplets

If the local liquid droplet density is relatively high, as at a sprinkler or hose nozzle, the drag on individual droplets is reduced by wake effects. FDS includes a sub-model that accounts for drag reduction due to wake effects. This model is invoked if the local liquid volume fraction is greater than a threshold value called `DENSE_VOLUME_FRACTION`, a parameter that is set on the `PART` line. Setting this parameter to a relatively large number (1 is sufficient) turns off the wake reduction model. Its default value is 1×10^{-5} .

15.3.5 Primary Breakup

Using `DENSE_VOLUME_FRACTION` for wake reduction may not provide enough drag reduction for a fire hose stream. To accommodate this scenario, FDS introduces a “primary breakup” method. If `PRIMARY_BREAKUP_LENGTH` is set on the `PART` line, then until a particle travels that distance its drag coefficient is multiplied by `PRIMARY_BREAKUP_DRAG_REDUCTION_FACTOR`, also set on `PART`, and typically a small value, even zero. In this way, the particle stream behaves like a hose stream prior to instability. The parameters are empirical and the model should be tuned for a specific hose nozzle, see [34], for example.

```
&PART ID='hose stream',..., PRIMARY_BREAKUP_LENGTH=10, PRIMARY_BREAKUP_DRAG_REDUCTION_
    FACTOR=0 /
```

³By default, the range of particle sizes is divided into six bins, and the sampled particles are divided among these bins. This ensures that a reasonable number of particles are assigned to the entire spectrum of sizes. To change the default number of bins, set `N_STRATA` on the `PART` line.

15.3.6 Secondary Breakup

If `BREAKUP=T` is set on the `PART` line, particles may undergo secondary breakup. In this case you should also specify the `SURFACE_TENSION` (N/m) of the liquid and the resulting ratio of the median volumetric diameter, $D_{v,0.5}$, `BREAKUP_RATIO`. Its default is 3/7. Optionally, specify the distribution parameters `BREAKUP_GAMMA_D` and `BREAKUP_SIGMA_D`. These parameters are defined the same as `GAMMA_D` and `SIGMA_D` in sec. 15.3.3 but instead apply after breakup.

15.3.7 Warning Messages Related to Droplets

An important parameter for any simulation involving liquid droplets is `PARTICLES_PER_SECOND` on the `PROP` line, which is specified by the `DEVC` line that includes the sprinkler or nozzle coordinates. The default value is 5000 particles per second. In some cases, this value might be too low, and you may want to raise it. The reason is that each liquid droplet that FDS explicitly tracks represents many, many more actual droplets, and the effect on the simulation of this one “super drop” may be relatively large. For example, a single droplet within a grid cell might decrease the gas temperature rapidly over the duration of just one time step—something that would not happen if there were many more droplets within the cell. You might even see within the output file named `case_name.err` warning messages like:

```
WARNING Delta TMP_G. Mesh: 5 Particle: 3245
```

This means that during the temperature update of a particular grid cell in Mesh 5, the droplet whose index is 3245 caused the cell gas temperature (`TMP_G`) to change too rapidly. If you only see a few of these messages, it is not a problem given how many droplets and time steps there are. However, if you see these messages persistently, you might want to increase the `PARTICLES_PER_SECOND` to reduce the likelihood that a single droplet will make such a dramatic change in a single time step.

15.4 Solid Particles

Lagrangian particles can represent a wide variety of subgrid-scale objects, from office clutter to vegetation. To create solid, non-liquid particles, you must add a `SURF_ID` to the `PART` line. The specified `SURF` line contains the parameters that describe the thermo-physical properties and geometric parameters of the particle. These properties are the same as those you would apply to an `OBST` or `VENT`. FDS uses the same solid phase conduction and pyrolysis algorithm for particles as it does for solid walls.

If the `SURF` line that is associated with the particle class calls for it, the particles will heat up due to convection from the surrounding gases and radiation from near and distant sources. The convective heat transfer coefficient takes into account the particle geometry, and the radiative heat flux is based on the integrated intensity. That is, the radiation heat flux is the average over all angles.

15.4.1 Basic Geometry and Boundary Conditions

To demonstrate the basic syntax for solid particles, the following input lines create a collection of hot spheres:

```
&PART ID='spheres', SURF_ID='HOT', STATIC=T, PROP_ID='ball' /
&SURF ID='HOT', TMP_FRONT=500., RADIUS=0.005, GEOMETRY='SPHERICAL' /
&PROP ID='ball', SMOKEVIEW_ID='SPHERE', SMOKEVIEW_PARAMETERS(1)='D=0.01' /
&INIT PART_ID='spheres', XB=0.25,0.75,0.25,0.75,0.25,0.75, N_PARTICLES=10 /
```

The `PART` line establishes the class of particles. In this case, the presence of a `SURF_ID` indicates that the particles are solids with the properties given by the `SURF` line 'HOT'. `STATIC` is a logical parameter whose default is `F` that indicates if the particles are stationary. The `PROP_ID` references a `PROP` (property) line that just tells Smokeview that the particles are to be drawn as spheres of diameter 0.01 m. See Sec. 18.7.3 for details and options. The `INIT` line randomly fills the given volume with 10 of these hot spheres. See Sec. 15.5.3 for details.

If the `SURF` line includes a `MATL_ID`, the particle mass will be based upon the value(s) of `DENSITY` of the referenced `MATL` line(s). If there is to be no heat conduction calculation in depth, do not specify a `MATL_ID`. Instead, you can specify, for example, the surface temperature, `TMP_FRONT` ($^{\circ}\text{C}$), heat release rate per unit area, `HRRPUA` (kW/m^2), or species `MASS_FLUX` ($\text{kg}/(\text{m}^2\text{s})$).

The `GEOMETRY` options for solid particles are 'SPHERICAL', 'CYLINDRICAL', or 'CARTESIAN'. By default, the `GEOMETRY` is 'CARTESIAN', in which case you need to provide the `LENGTH` and `WIDTH` of the rectangular plate. It is assumed that the plate is symmetric front and back (note this means you should set `BACKING='INSULATED'` on the `SURF` line). You need only specify the layers that make up the half-thickness. The array `THICKNESS(N)` indicates the thickness(es) of each layer of the plate, not the total thickness of the plate itself. If the plate is composed of only one material component, the specified `THICKNESS` is taken as the half-thickness of the plate.

For 'CYLINDRICAL' or 'SPHERICAL' particles, specify the `INNER_RADIUS` and `THICKNESS` of the individual layers. Alternatively, you can just specify the `RADIUS` if the cylinder or sphere is solid and has only one material component. The default value of `INNER_RADIUS` is 0 m, which means that the radius of the cylinder or sphere is the sum of the `THICKNESS` values. Remember that the layers are to be listed starting at the surface, not the center. For 'CYLINDRICAL' particles, specify a `LENGTH` as well.

Thermally Thick Droplet Model

Note that if the `GEOMETRY` is set to 'SPHERICAL' and a `BOILING_TEMPERATURE` is specified, thus invoking the liquid pyrolysis model, then the mass transfer relationships will follow those of an evaporating droplet.

The difference between this model and the default droplet model is that in this case the internal droplet temperature will be solved for using the 1D conduction solver. Usually, the `CELL_SIZE_FACTOR` needs to be decreased to get sufficient resolution inside the drop—you should double check the number of solid phase nodes by examining the `CHID.out` file.

15.4.2 Drag

The drag force exerted by moving or stationary particles is detailed in the FDS Technical Reference Guide, chapter “Lagrangian Particles” [3]. For solid particles, the default drag law is that of a solitary sphere. To invoke a different drag law, that of a solitary cylinder for example, set `DRAW_LAW='CYLINDER'` on the `PART` line. A summary of the available drag laws is given in table 15.1. If none of these options is applicable, you may specify a constant value of the drag coefficient for a particle class (a specific `PART_ID`) by setting a `DRAW_COEFFICIENT` on the `PART` line. The `DRAW_COEFFICIENT` over-rides the `DRAW_LAW`.

Table 15.1: Drag laws available in FDS

DRAW_LAW	Reference
'SPHERE' (default)	FDS Tech Guide [3]
'CYLINDER'	FDS Tech Guide [3]
'DISK'	FDS Tech Guide [3]
'POROUS MEDIA'	Sec. 15.4.8
'SCREEN'	Sec. 15.4.9

If you are modeling a relatively dense collection of solid particles, like vegetation, you should set the `DRAW_COEFFICIENT` explicitly and not rely on the correlations for spheres and cylinders which were developed for relatively independent bodies, not clusters.

Near Wall Particle Interpolation The momentum exchange between a particle and the fluid depends on the local fluid velocity at the particle position. Normally, the velocity at the particle position is taken from a tri-linear interpolation from the staggered velocity components nearest the particle. Thus, the particle lives in a different staggered cell for each component of velocity. When the particle position is within half a grid cell from the wall the default behavior is to use fluid velocity component tangential to the wall when computing the drag force in that component direction. This approximation is justified based on the plug flow profile seen in highly turbulent flows. However, as the flow becomes more resolved, this approximation may not be appropriate. If you specify `NEAR_WALL_PARTICLE_INTERPOLATION=T` on `MISC` then the fluid velocity will be linearly interpolated between the staggered component value and the no slip condition at the wall.

15.4.3 Radiation Absorption and Emission

Solid particles absorb and emit thermal radiation. The contribution from particles to the radiation absorption coefficient in each grid cell, ijk , is given by:

$$\kappa_{p,ijk} = \frac{\sum \epsilon_p A_p}{4 V_{ijk}} \quad (15.4)$$

where the summation is made over each grid cell of volume, V_{ijk} . The emissivity of each individual particle is given by ϵ_p and A_p is the particle surface area.

15.4.4 Size Distribution

By default, solid particles of a given class are mono-disperse; that is, have the same initial size which is specified via the `SURF` line that is designated on the `PART` line. However, it is possible to specify a distribution for the diameter of solid cylinders and spheres. Consider a case where you have a collection of cylindrical wooden rods whose diameters are uniformly distributed between $4000\ \mu\text{m}$ and $6000\ \mu\text{m}$:

```
&SURF ID='rod', MATL_ID='...', THICKNESS=0.0030, LENGTH=0.02, GEOMETRY='CYLINDRICAL' /
&PART ID='rods', SURF_ID='rod', ..., MONODISPERSE=F, CNF_RAMP_ID='dist', N_STRATA=1 /
&RAMP ID='dist', T=4000., F=0. /
&RAMP ID='dist', T=6000., F=1. /
```

The material properties of the rods are specified via the `MATL` line (not shown). The nominal radius of the rods is given by the parameter `THICKNESS` (m) on the `SURF` line, but the distribution of the rod *diameters* is specified by the parameter `CNF_RAMP_ID` (Cumulative Number Fraction) on the `PART` line. The `RAMP` lines designate the cumulative distribution function of the rod *diameters*, where the pairs (T,F) denote a uniform distribution between $4000\ \mu\text{m}$ and $6000\ \mu\text{m}$. Note that the units of the distribution are μm , not m. The parameter `N_STRATA` indicates the number of bins sub-dividing the diameter range, which is set to 1 here because the range is relatively small compared to typical liquid droplet size distributions. Note that the particle arrays for surface heat transfer are allocated based upon the `THICKNESS` on `SURF`. To avoid potential array size issues, it is suggested that `THICKNESS` on `SURF` be defined as the maximum value on the `RAMP`.

Following is an example of how to apply a distribution of solid particles. In the example case called `WUI/hot_rods.fds`, $L = 0.02\ \text{m}$ long wooden rods whose radii are uniformly distributed between $r = 0.002\ \text{m}$ and $r = 0.003\ \text{m}$ ($f(r) = 1000$, $0.002 < r < 0.003$; $f(r) = 0$ elsewhere) are poured onto a hot ($800\ ^\circ\text{C}$) floor and burned. The original density of the material is $\rho_p = 440\ \text{kg/m}^3$. In total, 4000 rods are ejected from a small vent over the course of 10 s. The total mass is

$$m = 4000 \times \pi \bar{r}^2 L \rho_p \approx 0.7\ \text{kg} \quad ; \quad \bar{r}^2 = \int_{0.002}^{0.003} r^2 f(r) dr \approx 6.333 \times 10^{-6} \quad (15.5)$$

Figure 15.3 displays the distribution of radii and the mass of rods as a function of time. Note that 8 % of the mass remains as char.

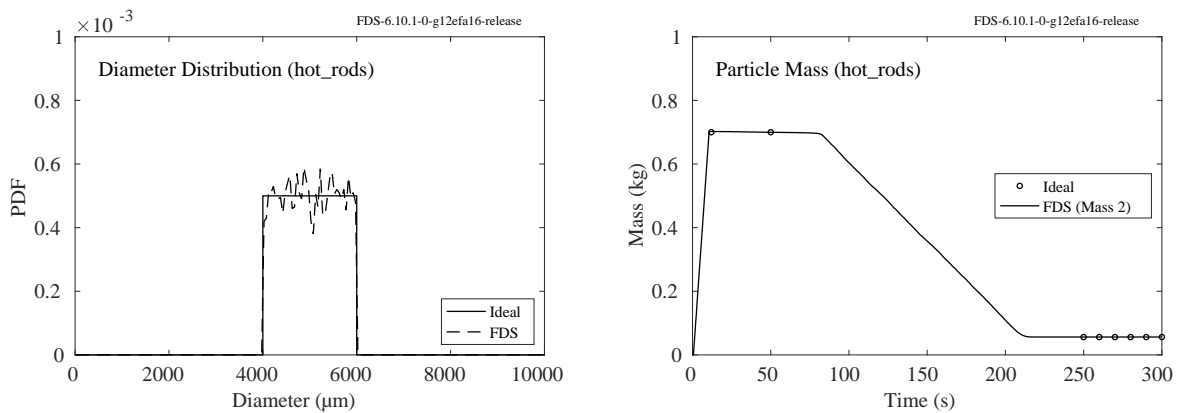


Figure 15.3: (Left) Distribution of rod diameters. (Right) Total mass of rods as a function of time.

15.4.5 Solid Particle Movement on Solid Surfaces

By default, solid particles do not “stick” or adhere to solid surfaces like liquid droplets do. However, you can make solid particles act like liquid droplets by setting `ADHERE_TO_SOLID=1` on the `PART` line, in which case the particles will stick and be reassigned a new speed and direction. If the surface is horizontal, the direction is randomly chosen. If vertical, the direction is downward. The parameters `HORIZONTAL_VELOCITY` and `VERTICAL_VELOCITY` on the `PART` line allow you to control the speed at which particles move horizontally or vertically (downward). The defaults are 0.2 m/s and 0.5 m/s, respectively. If you want the particle to truly stick and not move on a surface, set these values to 0.

15.4.6 Particle Orientation

If an `ORIENTATION` vector is assigned on a `PART` line, the radiative flux to the particle is calculated as if there is a flat plate normal to the direction of the vector, like a conventional heat flux gauge. That is, the heat flux is not an integrated average over the entire particle but rather the directional heat flux with the given orientation. The reason for this exception to the general rule is that often single particles are used as “targets” to record a heat flux at a given point in the domain with a given orientation. These particles can be thought of as tiny heat flux gauges that do not disturb the flow.

15.4.7 Gas Generating Particles

Lagrangian particles can be used to generate gases at a specified rate. The syntax is similar to that used for a solid wall. For example, the following input lines create three particles – one shaped like a rectangular plate, one a cylinder, and one a sphere – that generate argon, sulfur dioxide, and helium, respectively. The particles have no mass; they simply are used to generate the gases at a specified rate.

```
&SPEC ID='ARGON' /
&SPEC ID='SULFUR DIOXIDE' /
&SPEC ID='HELIUM' /

&INIT PART_ID='plate', XYZ=-1.,0.,1.5, N_PARTICLES=1 /
&INIT PART_ID='tube', XYZ= 0.,0.,1.5, N_PARTICLES=1 /
&INIT PART_ID='ball', XYZ= 1.,0.,1.5, N_PARTICLES=1 /

&PART ID='plate', SAMPLING_FACTOR=1, SURF_ID='plate bc', STATIC=T /
&PART ID='tube', SAMPLING_FACTOR=1, SURF_ID='tube bc', STATIC=T /
&PART ID='ball', SAMPLING_FACTOR=1, SURF_ID='ball bc', STATIC=T /

&SURF ID='plate bc', THICKNESS=0.001, LENGTH=0.05, WIDTH=0.05, SPEC_ID(1)='ARGON',
MASS_FLUX(1)=0.1, TAU_MF(1)=0.001 /
&SURF ID='tube bc', GEOMETRY='CYLINDRICAL', LENGTH=0.05, RADIUS=0.01,
SPEC_ID(1)='SULFUR DIOXIDE', MASS_FLUX(1)=0.1, TAU_MF(1)=0.001 /
&SURF ID='ball bc', GEOMETRY='SPHERICAL', RADIUS=0.01, SPEC_ID(1)='HELIUM',
MASS_FLUX(1)=0.1, TAU_MF(1)=0.001 /
```

In this case, there is no calculation of heat conduction in depth. Only the surface area is important. For the plate, the surface area is twice the length times the width. For the cylinder, the area is twice the radius times π times the length. For the sphere, the area is 4π times the radius squared. Figure 15.4 displays the output of the test case called `surf_mass_part_specified.fds`, demonstrating that the production rate of the gases is as expected.

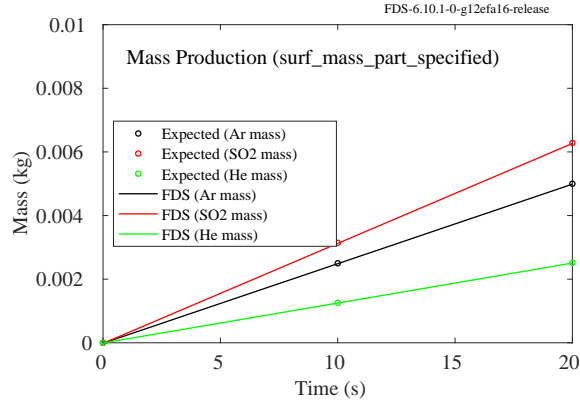


Figure 15.4: Gas production from three Lagrangian particles.

15.4.8 Porous Media

A 3-D array of particles can be used to represent the drag exerted by porous media, as in the following example for a porous aluminum matrix:

```
&SURF ID='LIGAMENT', MATL_ID='ALUMINUM ALLOY', THICKNESS=7.3E-5,
      GEOMETRY='CYLINDRICAL', HEAT_TRANSFER_COEFFICIENT=10. /
&MATL ID='ALUMINUM ALLOY', DENSITY=2690., CONDUCTIVITY=218., SPECIFIC_HEAT=0.9 /
&PART ID='FOAM', DRAG_LAW='POROUS MEDIA', SURF_ID='LIGAMENT',
      POROUS_VOLUME_FRACTION=0.12, STATIC=T,
      DRAG_COEFFICIENT=0.1,0.1,0.1, PERMEABILITY=1.0E-7,1.E-7,1.E-7 /
&INIT XB=1.010,1.095,0.0,0.5,0.0,0.5, N_PARTICLES_PER_CELL=1, CELL_CENTERED=T,
      PART_ID='FOAM' /
```

The basic geometry of the foam ligaments is defined with the `SURF` line. It is assumed that the ligaments are made of an aluminum alloy whose properties are given on the `MATL` line. The radius of the assumed cylindrical ligament is indicated by the `THICKNESS`. Note that the `LENGTH` of the cylinder which is normally required on the `SURF` line is computed automatically so that the volume fraction of the grid cell occupied by the foam, specified by `POROUS_VOLUME_FRACTION` on the `PART` line, is achieved. In a sense, the foam is modeled by a long cylinder chopped up into small pieces and represented by a single particle in each grid cell. The `SURF` inputs of `POROUS_VOLUME_FRACTION` and `THICKNESS` do not directly effect the drag calculation. They are used to define the surface area for computing heat transfer between the gas and the porous media.

The `PART` line provides information about the particles which includes the drag they impose on the flow. The `DRAG_LAW` of `POROUS MEDIA` indicates a special drag model for porous media. This model states that the pressure drop through the media is given by

$$\Delta p = \delta \left(\frac{\mu}{K} u + \rho \frac{Y}{\sqrt{K}} u^2 \right) \quad (15.6)$$

where δ is the thickness of the foam block in the flow direction (the smaller of the cell size in the flow direction or the `XB` dimension in the flow direction from the `INIT` input), μ is the viscosity of the gas, u is the velocity component in the flow direction, ρ is the density of the gas, K is the `PERMEABILITY` in units of m^2 , and Y is a dimensionless inertial term that you specify using the parameter `DRAG_COEFFICIENT`. When using the porous media model the `PERMEABILITY` and `DRAG_COEFFICIENT` must be specified for all three

directions.

The `INIT` line designates the volume occupied by the porous media using the sextuplet `XB`. A single particle is inserted into the center of each cell occupied by the foam by specifying the parameters `N_PARTICLES_PER_CELL=1` and `CELL_CENTERED=T`.

A sample calculation involving porous media is contained in the folder `Sprinklers_and_Sprays`. The input file is called `porous_media.fds`.

15.4.9 Screens

A 2-D array of particles can be used to represent the drag exerted by a window screen, as in the following example:

```
&INIT N_PARTICLES_PER_CELL=1, CELL_CENTERED=T, PART_ID='SCREEN',
      XB=1.01,1.02,0.0,1.0,0.0,1.0/
&PART ID='SCREEN', DRAG_LAW='SCREEN', FREE_AREA_FRACTION=0.4, STATIC=T,
      SURF_ID='SCREEN', ORIENTATION=1,0,0 /
&SURF ID='SCREEN', THICKNESS=0.00015, GEOMETRY='CYLINDRICAL',
      MATL_ID='ALUMINUM' /
&MATL ID='ALUMINUM', DENSITY=2700., CONDUCTIVITY=200., SPECIFIC_HEAT=0.9 /
```

The `INIT` line designates the plane of the screen using the sextuplet `XB`. A single particle is inserted into each cell by specifying the parameter `N_PARTICLES_PER_CELL=1`. The particles are defined with a `SURF_ID` containing the material properties of the screen. A special drag law for screens is specified via the `DRAG_LAW`. `ORIENTATION` is the direction normal to the screen, and `FREE_AREA_FRACTION` is the fraction of the screen's surface area that is open. In the example, an aluminum screen with a 40 % free area and an 0.0003 m wire diameter is placed normal to the x -axis. Note that the `LENGTH` parameter on the `SURF` line will be computed automatically so the fraction of the grid cell flow area occupied by the screen is equal to $1 - \text{FREE_AREA_FRACTION}$. The pressure drop across the screen is given by

$$\Delta p = l \left(\frac{\mu}{K} u + \rho \frac{Y}{\sqrt{K}} u^2 \right) \quad (15.7)$$

where l is the screen thickness (equal to the wire diameter), μ is the viscosity of the gas, u is the velocity normal to the screen, ρ is the density of the gas, and Y and K are empirical constants given by [35]

$$K = 3.44 \times 10^{-9} \text{ FREE_AREA_FRACTION}^{1.6} \text{ m}^2 \quad (15.8)$$

$$Y = 0.043 \text{ FREE_AREA_FRACTION}^{2.13} \quad (15.9)$$

This correlation was developed using screens with `FREE_AREA_FRACTION` ranging from 0.3 to 0.6.

The `MISC` input parameter `PARTICLE_CFL_MAX` controls the time step on the pressure drop across a screen (or other porous media). If a screen or other porous media introduces numerical instabilities, reducing the value of `PARTICLE_CFL_MAX` below 1 may resolve them.

15.4.10 Electrical Cables

Petra Andersson and Patrick Van Hees of the Swedish National Testing and Research Institute (SP) have proposed that the thermally-induced electrical failure (THIEF) of a cable can be predicted via a simple one-dimensional heat transfer calculation, under the assumption that the cable can be treated as a homogeneous cylinder [36]. Their results for PVC cables were encouraging and suggested that the simplification of the analysis is reasonable and that it should extend to other types of cables. The assumptions underlying the THIEF model are as follows:

1. The heat penetration into a cable of circular cross section is largely in the radial direction. This greatly simplifies the analysis, and it is also conservative because it is assumed that the cable is completely surrounded by the heat source.
2. The cable is homogeneous in composition. In reality, a cable is constructed of several different types of polymeric materials, cellulosic fillers, and a conducting metal, most often copper.
3. The thermal properties – conductivity, specific heat, and density – of the assumed homogeneous cable are independent of temperature. In reality, both the thermal conductivity and specific heat of polymers are temperature-dependent, but this information is very difficult to obtain from manufacturers.
4. It is assumed that no decomposition reactions occur within the cable during its heating, and ignition and burning are not considered in the model. In fact, thermoplastic cables melt, thermosets form a char layer, and both off-gas volatiles up to and beyond the point of electrical failure.
5. Electrical failure occurs when the temperature just inside the cable jacket reaches an experimentally determined value.

Obviously, there are considerable assumptions inherent in the Andersson and Van Hees THIEF model, but their results for various polyvinyl chloride (PVC) cables suggested that it may be sufficient for engineering analyses of a wider variety of cables. The U.S. Nuclear Regulatory Commission sponsored a study of cable failure known as CAROLFIRE [37]. The primary project objective of CAROLFIRE was to characterize the various modes of electrical failure (e.g., hot shorts, shorts to ground) within bundles of power, control and instrument cables. A secondary objective of the project was to develop a simple model to predict thermally-induced electrical failure when a given interior region of the cable reaches an empirically determined threshold temperature. The measurements used for these purposes are described in Volume II of the CAROLFIRE test report. Volume III describes the modeling.

The THIEF model can only predict the temperature profile within the cable as a function of time, given a time-dependent exposing temperature or heat flux. The model does not predict at what temperature the cable fails electrically. This information is gathered from experiment. The CAROLFIRE experimental program included bench-scale, single cable experiments in which temperature measurements were made on the surface of, and at various points within, cables subjected to a uniform heat flux. These experiments provided the link between internal cable temperature and electrical failure. The model can only predict the interior temperature and infer electrical failure when a given temperature is reached. It is presumed that the temperature of the center-most point in the cable is not necessarily the indicator of electrical failure. This analysis method uses the temperature just inside the cable jacket rather than the center-most temperature, as that is where electrical shorts in a multi-conductor cable are most likely to occur first.

To use the THIEF model in FDS, add lines similar to the following to the input file:

```
&MATL ID='plastic', DENSITY=2535., CONDUCTIVITY=0.2, SPECIFIC_HEAT=1.5 /
&SURF ID='cylinder', THICKNESS=0.00815, LENGTH=0.1, MATL_ID='plastic',
    GEOMETRY='CYLINDRICAL' /
&PART ID='Cable Segment', SURF_ID='cylinder', ORIENTATION=0,0,1,
    STATIC=T /
&INIT ID='Cable', XB=0.01,0.01,0.,0.,0.,0., N_PARTICLES=1, PART_ID='Cable Segment' /
&DEVC ID='Cable Temp', INIT_ID='Cable',
    QUANTITY='INSIDE WALL TEMPERATURE', DEPTH=0.0015 /
```

The THIEF model assumes that the cable plastic material has a thermal conductivity of 0.2 W/(mK) and a specific heat of 1.5 kJ/(kgK). If you change these values, you are no longer using the THIEF model. The density is the mass per unit length of the cable divided by its cross sectional area. The THICKNESS is

the radius of the cylindrical cable in units of m. The `LENGTH`, in m, is needed by FDS because it assumes that the cable is a cylindrical segment of a certain length. It has no impact on the simulation, and its value is typically the size of a grid cell. The `ORIENTATION` tells FDS the direction of the prevailing radiative source. `STATIC=T` prevents the cable from moving. The `INIT` line is used to position the cable within the computational domain. The `DEVC` line records the cable's inner temperature, in this case 1.5 mm below the surface. This is typically the jacket thickness.

15.4.11 Target Particles

To declare that a solid particle is only for recording the heat flux to a location away from a solid surface, specify `TARGET_ONLY=T` on the `PART` line. When this flag is set and material properties are given, FDS will compute the particle temperature using the local gas temperature and radiative conditions; however, the particle will not exchange energy or momentum with the gas. When this flag is set, a `CARTESIAN` particle's `SURF` will no longer be treated as a half-thickness plate. This allows the use of backside boundary conditions on `SURF` such as `TMP_GAS_BACK` and `EMISSIVITY_BACK`. If it is still desired to have the `THICKNESS` represent half the thickness of a plate heated from both sides, then set `BACKING='INSULATED'` on the `SURF` line. Target particles are treated as `STATIC` but can be positioned using `PATH_RAMP` and/or `ORIENTATION_RAMP` on `INIT`.

15.5 Particle Insertion

There are three ways of introducing droplets or particles into a simulation:

1. Define a sprinkler or nozzle using a `PROP` line that includes a `PART_ID` that specifies the particle or droplet parameters. The individual sprinklers or nozzles are specified via `DEVC` lines.
2. Add a `PART_ID` to a `SURF` line, in which case particles or droplets will be ejected from that surface with an outward normal velocity given by a negative value of `VEL`.
3. Create an `INIT` line that defines a volume within the computational domain in which the particles/droplets are to be introduced initially and/or periodically in time.

It is not unusual to include hundreds of thousands of particles in a simulation. Visualizing all of the particles in Smokeview can sometimes be impractical due to memory limitations. To limit the amount of particles, you can make use of the following parameters on the `PART` line:

- `SAMPLING_FACTOR` can be used to reduce the size of the particle output file used to animate the simulation. The default value is 1 for `MASSLESS` particles, meaning that every particle or droplet will be shown in Smokeview. The default is 10 for all other types of particles. `MASSLESS` particles are discussed in Sec. 15.2.
- `AGE` is the number of seconds the particle or droplet exists, after which time it is removed from the calculation. This is a useful parameter to use when trying to reduce the number of droplets or particles in a simulation.

Be careful when setting parameters related to particle/droplet insertion, as it is very easy to overwhelm the simulation with millions of them. The parameter `MAXIMUM_PARTICLES` on the `DUMP` line sets the maximum number of particles that can be included on any given mesh at any given time. Its default value is 1000000. If this value is exceeded, FDS will automatically remove particles, starting with the oldest.

By default, FDS initially allocates space for 50 particles per mesh. If more particles are needed, the arrays are expanded by `NEW_PARTICLE_INCREMENT`, an integer set on the `PART` line. Its default value is 1000. In most cases, this should be enough. However, for cases with a very large number of particles, such as a wildland fire simulation, you might want to increase this value to save on the reallocation time. This does not affect how particles are inserted into the domain (as described above), only how much new storage is reserved when the current allocation is exceeded. Be careful, as it is possible to massively over-allocate and reserve large amounts of unused memory.

15.5.1 Particles Introduced at a Solid Surface

There are three ways of introducing particles at a solid surface. You can specify a particle mass flux and inject particles regularly in time, you can restrict the production of particles to actively burning surfaces (as a means of generating embers), or you can specify a particle surface density and introduce the particles at the start of the simulation.

Particles specified using a mass flux

If the particles have mass and are introduced from a solid surface, specify `PARTICLE_MASS_FLUX` on the `SURF` line. The number of particles inserted at each solid cell every `DT_INSERT` seconds is specified by `NPPC` (Number of Particles Per Cell) on the `SURF` line defining the solid surface. The default value of `DT_INSERT` is 0.01 s and `NPPC` is 1. As an example, the following set of input lines:

```
&PART ID='particles', ... /
&SURF ID='SLOT', PART_ID='particles', VEL=-5., PARTICLE_MASS_FLUX=0.1 /
&OBST XB=-0.2,0.2,-0.2,0.2,4.0,4.4, SURF_IDS='INERT','SLOT','INERT' /
```

creates an obstruction that ejects particles out of its sides at a rate of $0.1 \text{ kg}/(\text{m}^2 \text{ s})$ and a velocity of 5 m/s (the minus sign indicates the particles are ejected from the surface). FDS will adjust the mass flux if the obstruction or vent dimensions are changed to conform to the numerical grid. The `IDS` have no meaning other than as identifiers. By default, the particle initial velocity is the surface normal velocity. This means that the surface on which particles are specified must have a non-zero normal velocity directed into the computational domain. This happens automatically if the surface is burning. If not then either specify a non-zero and negative `VEL`, or if you do not wish to also inject gas you can specify `VEL_PART`. `VEL_PART` has the same sign convention as `VEL`. If `VEL_PART` is specified, it will override the gas velocity for the surface and inject the particle at `VEL_PART`. There is a simple input file called `particle_flux.fds` that demonstrates how the above input lines can produce a stream of particles from a block. The total mass flux from the block is the product of the `PARTICLE_MASS_FLUX` times the total area of the sides of the block, $0.4 \text{ m} \times 0.4 \text{ m} \times 4$. The expected accumulated mass of particles on the ground after 10 s is expected to be 0.64 kg , as shown in Fig. 15.5.

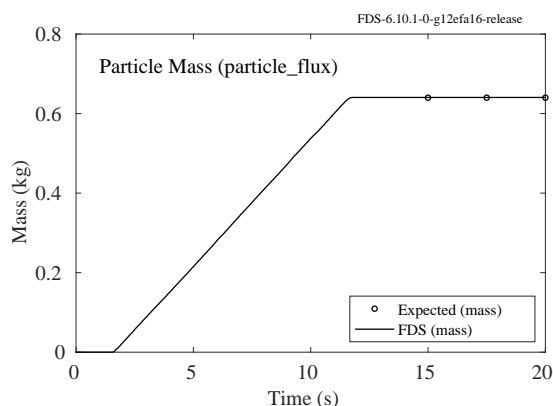


Figure 15.5: Simple test case to demonstrate mass conservation of particles ejected from an obstruction.

Note also that you can independently control particles that emanate from a solid surface. For example, a device might control the activation of a fan, but you can over-ride the device and control the particles separately. To do this, specify either a device or controller via a `DEVC_ID` or `CTRL_ID` on the `PART` line that defines the particles. For more information on devices and controls, see Sections 18.4 and 18.5.

Particles specified using an ember generation height

If you specify an `EMBER_GENERATION_HEIGHT` (m), particles are assumed to represent embers (or firebrands) and are only generated at surfaces that are actively burning (non-zero fuel mass flux). The height parameter dictates the vertical offset distance from the burning surface at which particles are produced. This is important in cases where the vertical fuel structure may not be resolved, such as level set fire spread (Sec. 17.5), but the height of ember production plays a role in the lofting potential. If a single number is provided for the `EMBER_GENERATION_HEIGHT`, all particles are generated at that offset distance. If two numbers are provided, particles are randomly generated within the specified offset range, following a uniform distribution. For example, if `EMBER_GENERATION_HEIGHT=0.0, 1.0` then particles will be produced anywhere up to 1.0 m above the burning surface.

The rate of ember production can then be set using the method described previously. A number flux can be set using `DT_INSERT` and `NPPC`, and these particles can be weighted to match a specified `PARTICLE_MASS_FLUX`. It is recommended to set `VEL_PART=0.0` when modeling ember generation in this way, so that particles start from rest and are naturally lofted by the surrounding flow.

Particles specified using a surface density

Instead of a mass flux, you can specify the `PARTICLE_SURFACE_DENSITY` (kg/m^2) for particles introduced at the start of a simulation. You can introduce `NPPC` particles per cell, although 1 is sufficient to represent the specified surface density. This feature is useful for creating subgrid-scale dry vegetative fuels.

15.5.2 Particles or Droplets Introduced at a Sprinkler or Nozzle

A sprinkler or nozzle is added to the simulation using a `PROP` line to describe the features of the device and a `DEVC` line to position and orient the device within the computational domain. `PARTICLES_PER_SECOND` is the number of droplets inserted every second per active sprinkler or nozzle (Default 5000). It is listed on the `PROP` line that includes other properties of the sprinkler or nozzle. Note that this parameter only affects sprinklers and nozzles. Changing this parameter does *not* change the flow rate, but rather the number of droplets used to represent the flow. In some simulations, it is a good idea to increase the `PARTICLES_PER_SECOND` beyond its default so that the particle/droplet mass is distributed more uniformly inside the domain. If this parameter is too small, it can lead to a non-physical evaporation pattern, sometimes even to the point of causing a numerical instability. If you encounter a numerical instability shortly after the activation of a sprinkler or nozzle, consider increasing `PARTICLES_PER_SECOND` to produce a smoother evaporation pattern that is more realistic. Keep in mind that for a real sprinkler or nozzle, there are many more droplets created per second than the number that can be simulated.

The `PARTICLE_VELOCITY` specified on the `PROP` line indicates the initial velocity of the droplets/particles. If this value is relatively large, FDS computes the trajectory of the droplet/particle by taking sub-steps of the gas phase time step to ensure that the droplet/particle does not traverse more than a single grid cell within a sub-time step. A stricter limit on the time step is given by the parameter `PARTICLE_CFL` on the `MISC` line which is set to `F` by default (see Section 19.3.3). If `T` the gas phase time step is limited by the particle CFL. Normally, it is not necessary to restrict the gas phase time step to account for fast droplets/particles—the sub-time stepping is usually adequate.

15.5.3 Particles or Droplets Introduced within a Volume

Sometimes it is convenient to introduce Lagrangian particles within a particular region of the domain. To do this, use an `INIT` line which contains the `PART_ID` for the type of particle to be inserted. Particles specified via an `INIT` line can represent a number of different kinds of subgrid-scale objects. The particles can be massless tracers or they can be solid or liquid particles with mass. If not massless, specify `MASS_PER_VOLUME` in units of kg/m^3 . Do not confuse this parameter with `DENSITY`, explained in the next section. For example, water has a `DENSITY` of 1000 kg/m^3 , whereas a liter of water broken up into droplets and spread over a cubic meter has a `MASS_PER_VOLUME` of 1 kg/m^3 . The number of Lagrangian particles inserted is controlled by the parameter `N_PARTICLES`.

Randomly Distributed Particles within a Specified Volume

The parameter `N_PARTICLES` on the `INIT` line indicates the number of particles to insert within a specified region of the domain. This region can take on a number of shapes, depending on the parameter `SHAPE`: 'BLOCK' (default), 'CYLINDER', 'CONE', 'LINE', 'RING'. For 'BLOCK', 'CONE', and 'RING' the particle

positions are randomly distributed by default. For 'RING' you can position the particles uniformly by setting `UNIFORM=T` on the `INIT` line. By default, the region is a rectangular 'BLOCK' designated with the real sextuplet `XB`. The format for `XB` is the same as that used on the `OBST` line.

```
&INIT PART_ID='my particles', XB=..., N_PARTICLES=..., MASS_PER_VOLUME=... /
```

Note that the volume of the specified region is calculated according to the `SHAPE` dimensions, regardless of whether there are solid obstructions within this region. Note also that in most applications, the number of particles, `N_PARTICLES`, is somewhat arbitrary but should be chosen to provide at least a few particles per grid cell. FDS will then automatically assign a weighting factor to each particle to ensure that the `MASS_PER_VOLUME` is achieved. In some applications, on the other hand, it may be important to specify the number of particles. For example, if using particles to model the burning of electrical cables, you may want to specify how many cables are actually burning. The `MASS_PER_VOLUME` can be ramped up and down in time using the ramp function `RAMP_PART` on the `INIT` line.

If the volume specified by the sextuplet `XB` crosses mesh boundaries, be aware that `N_PARTICLES` refers to the entire volume, not just the volume within a particular mesh. FDS will automatically compute the necessary number of particles to assign to each mesh. This is done by multiplying `N_PARTICLES` by the ratio of the volume of `XB` in each mesh by the total volume of `XB` as specified in the input file (i.e, if the input is larger than the domain, fewer than `N_PARTICLES` will be inserted) with the result rounded to the nearest integer (with at least one particle inserted for every mesh that contains `XB`).

Alternatively, you can specify `SHAPE='CONE'`, in which case the particles will be randomly distributed within a vertical cone. This is primarily used for representing trees. The dimensions of the cone are specified via the parameters `RADIUS`, `HEIGHT`, and base position `XYZ`. The latter is a triplet of real numbers designating the point at the center of the base of the cone. Here is an example of how one might make a tree:

```
&PART ID='tree crown foliage',
      DRAG_LAW='CYLINDER',
      SURF_ID='needles',
      QUANTITIES='PARTICLE TEMPERATURE','PARTICLE MASS','PARTICLE DIAMETER',
      STATIC=T,
      COLOR='FOREST GREEN' /
&INIT PART_ID='tree crown foliage',
      XYZ=0.0,0.0,0.0,
      RADIUS=1,
      HEIGHT=2,
      SHAPE='CONE',
      N_PARTICLES_PER_CELL=1,
      CELL_CENTERED=T,
      MASS_PER_VOLUME=2.0 /
```

Note that in this example, exactly one particle is specified per grid cell, positioned exactly in the center of the cell. The number of actual pine needles this single particle represents depends on the specified `MASS_PER_VOLUME` and the mass of an individual cylindrical particle. That information would be provided by the `SURF` line 'needles'.

Additionally, for 'CONE' and 'CYLINDER' shapes it is possible to specify an `INNER_RADIUS`. This will remove a concentric volume with the specified inner radius, leaving a conical or cylindrical shell. This can be useful if a tree crown has foliage concentrated toward the exterior of the volume. Note that the `MASS_PER_VOLUME` will be applied only to the occupied volume of the shell, thus the total mass of the tree crown will decrease when specifying an `INNER_RADIUS` for a fixed `MASS_PER_VOLUME`. Examples of different crown geometries can be found in the `tree_shapes.fds` verification case in the `fds/Verification/WUI/` directory.

Specifying a Fixed Number of Particles per Grid Cell

There are special applications where you might want to specify `N_PARTICLES_PER_CELL` to indicate the number of particles within each grid cell of a specified region. When using `N_PARTICLES_PER_CELL`, the particles will be randomly placed within each cell. If you set `CELL_CENTERED=T`, the particles will be placed at the center of each cell.

Specifying a Weight Factor for Particles

Use `PARTICLE_WEIGHT_FACTOR` to specify how many actual particles each of the computational particles represent. This can be used in conjunction with `N_PARTICLES_PER_CELL` to reduce the computational cost when a large number of identical particles would be placed in the same grid cell.

Single Particle Insertion

If you introduce only a single particle, which is often a handy way of creating a target, you may use the real triplet `XYZ` rather than `XB` to designate the particle's position. You can give this single particle an initial velocity using the real triplet `UVW`. You can also add `DX`, `DY`, and/or `DZ` to create a line of particles that are offset from `XYZ` by these increments in units of meters. For example,

```
&INIT PART_ID='target', XYZ=1.2,3.4,5.6, N_PARTICLES=10, DX=0.1 /
```

creates a line of 10 particles starting at the point (1.2,3.4,5.6) separated by 0.1 m. This is handy for creating arrays of devices, like heat flux gauges. See Sec. 22.10.12 for more details.

In special cases, you might want a single liquid droplet to be inserted at a particular point with a particular velocity every `DT_INSERT` s following the activation of a particular device, as follows:

```
&INIT N_PARTICLES=1, XYZ=..., UVW=..., DIAMETER=200., DT_INSERT=0.05,  
PART_ID='drops', DEVC_ID='nozzle' /
```

Note that the `DIAMETER` (μm) on the `INIT` line is only valid for liquid droplets. It over-rides the `DIAMETER` on the `PART` line labeled 'drops'. A simple test case that demonstrates this functionality is called `bucket_test_3`, in which water droplets are launched in different directions from a common point. Their size, velocity, insertion frequency, and mass flux are varied, and a check is made that water mass is conserved (see Fig. 15.6).

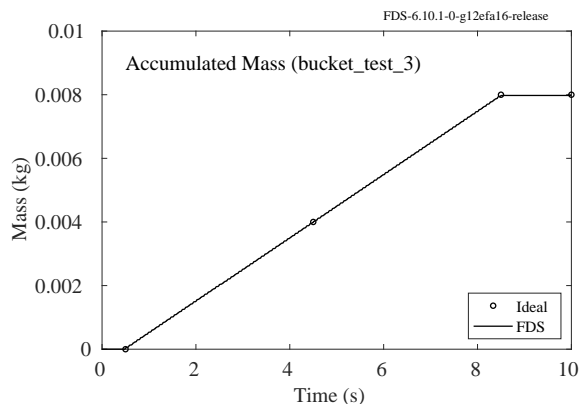


Figure 15.6: Accumulated water collected at the floor in the `bucket_test_3` case.

Periodic Insertion of Particles within a Specified Volume

If you want to introduce particles within a given region periodically in time and not just initially, set `DT_INSERT` on the `INIT` line to a positive value indicating the time increment (s) for insertion. The parameter `N_PARTICLES` now indicates the number of droplets/particles inserted every `DT_INSERT` seconds. If the droplets/particles have mass, use `MASS_PER_TIME` (kg/s) instead of `MASS_PER_VOLUME` to indicate how much mass is to be introduced per second. This parameter can be ramped up and down in time using the ramp function `RAMP_PART` on the `INIT` line.

If you want to delay the insertion of droplets, you can use either a `DEVC_ID` or a `CTRL_ID` on the `INIT` line to name the controlling device. See Sec. 18.4 for more information on controlling devices.

Controlled Particle Movement

Particles can either be static, massless tracers (go with the flow), massive droplets or thermally thick solids (experience drag), or have its position controlled by a `PATH_RAMP`. A scenario where this might be useful is when you have a heat or mass source that changes position in time. Or, you might want a `DEVC` measurement location to change with time; the `DEVC` can be linked to a particle through an `INIT` line with a `PATH_RAMP`, as we will discuss next.

The `INIT` parameter `PATH_RAMP(1:3)` is a list of three character strings for the ramp IDs associated with the *x*, *y*, and *z* positions of a particle. We use the volume insert method for the particle, with `N_PARTICLES=1`. An example may be found in the verification case `part_path_ramp_jog.fds` in the `fds/Verification/Miscellaneous/` directory. The output of the example is shown in Fig. 15.7. The basics that are needed to implement the path ramp are the following:

```
&RAMP ID='PART RAMP X', T=0, F=0/ start position at X=0 m at Time=0 s
&RAMP ID='PART RAMP X', T=10, F=5/ move linearly to X=5 m at Time=10 s
! define a particle class as a mass source
&PART ID='MASS SOURCE', SAMPLING_FACTOR=1, SURF_ID=..., PROP_ID=.../
! initialize the particle position and assign path ramp
&INIT ID='JOG', XB=..., PART_ID='MASS SOURCE',
      PATH_RAMP(1)='PART RAMP X', N_PARTICLES=1 /
! monitor the position (or other quantities) using a device
&DEVC QUANTITY='PARTICLE X', INIT_ID='JOG', ID='X', TIME_AVERAGED=F /
```

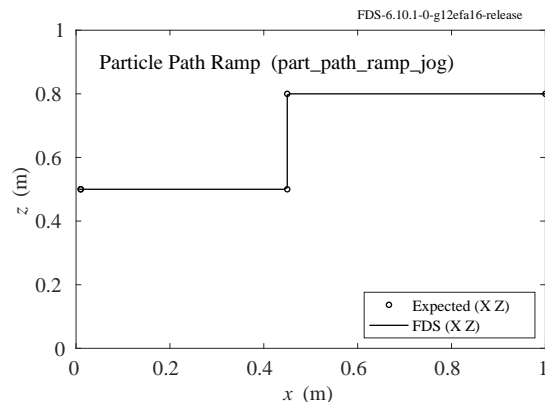


Figure 15.7: Particle path for `part_path_ramp_jog` case.

15.5.4 Controlled Particle Orientation

The `ORIENTATION` for a particle placed with an `INIT` line can be controlled by specifying `ORIENTATION_RAMP` on the `INIT` line. `ORIENTATION_RAMP` specifies a `RAMP` for each of the three components for the particle `ORIENTATION`. All three components must be defined. At each timestep, the outputs for the `RAMPS` will be normalized to a vector magnitude of one. In the example below, the particle orientation is changed from -x to +x over 10 s by rotating around the z-axis.

```
&PART ID='MY PART',SURF_ID='PART SURF',STATIC=T, ORIENTATION=1,1,1/  
&INIT ID='ONE PART',PART_ID='MY PART',XYZ=0.975,1,1,  
      ORIENTATION_RAMP(1:3)='O1','O2','O3',N_PARTICLES=1/  
&RAMP ID='O1',T=0, F=-1/  
&RAMP ID='O1',T=10,F= 1/  
&RAMP ID='O2',T=0, F= 0/  
&RAMP ID='O2',T=5, F= 1/  
&RAMP ID='O2',T=10,F= 0/  
&RAMP ID='O3',T=0, F= 0/  
&RAMP ID='O3',T=10,F= 0/
```

Note that the `PART` input must contain `ORIENTATION` as this signals to FDS to allocate array space for the angular radiative fluxes for that class of particles. This block of input was used in test case consisting of a 1 m³ box with the +x wall emitting a radiative flux of 10 kW/m², and the remaining walls emitting 1 kW/m². Results are shown in Fig. 15.8.

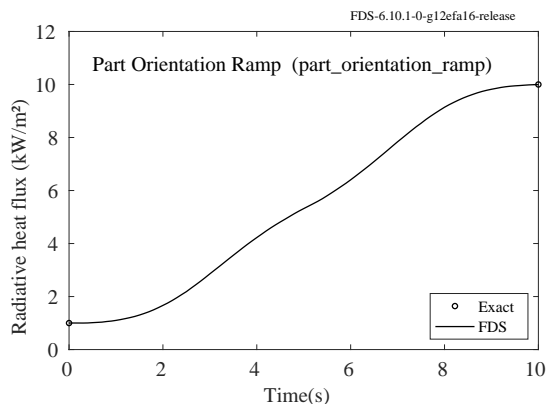


Figure 15.8: Particle radiative flux for `part_orientation_ramp` case.

15.6 Particle Removal

Lagrangian particles can be removed from a simulation in a number of ways:

1. Specify the `AGE` on the `PART` line so that particles are removed after this period of time after insertion.
2. Liquid droplets disappear when they hit the “floor” of the computational domain, regardless of whether it is solid or not. This feature mimics the presence of floor drains. To stop FDS from removing liquid droplets from the floor of the domain, add the phrase `POROUS_FLOOR=F` to the `MISC` line. Be aware, however, that droplets that land on the floor continue to move horizontally in randomly selected directions; bouncing off obstructions, and consuming CPU time. Note also that solid particles do not disappear from the floor of the domain like liquid droplets.
3. If droplets or particles are drawn toward an extracting vent, they will not be removed unless you specify a minimum `PARTICLE_EXTRACTION_VELOCITY` (m/s). That is, if the vent’s extraction velocity exceeds this value, the particles or droplets will be removed from the simulation. Otherwise, the droplets/particles will behave as if they have struck any other solid surface. By default, this parameter is a large positive value, meaning that you must specify a positive value less than the normal vent extraction velocity for this parameter to have an effect. If you want all particles/droplets to be extracted, set the value to a very small positive number.

15.7 Suppression by Water

Modeling fire suppression by water has three principal components: transporting the water droplets through the air, tracking the water along the solid surface, and predicting the reduction of the burning rate. This section addresses the latter two.

15.7.1 Droplet Movement on Solid Surfaces

When a liquid⁴ droplet strikes a solid surface⁵, it sticks and is reassigned a new speed and direction. If the surface is horizontal, the direction is randomly chosen. If vertical, the direction is downward. The parameters `HORIZONTAL_VELOCITY` and `VERTICAL_VELOCITY` on the `PART` line allow you to control the speed at which droplets move horizontally or vertically (downward). The defaults are 0.2 m/s and 0.5 m/s, respectively.

When a liquid droplet hits a solid surface, it transfers heat over the area of the cross section of a hemisphere of equal volume. When enough liquid pools on the surface to form a layer, the area governing heat transfer is based on the film, not the individual droplet. The diameter of the droplet, once a film layer is formed, is not important because it is assumed to mix with other droplets forming the film. However, when the droplet leaves the solid surface in the form of a “drip,” it reforms into a droplet, and its new diameter can be specified via `SURFACE_DIAMETER` on the `PART` line in units of μm .

The heat transfer between the solid surface and the liquid film layer makes use of an empirical turbulent heat transfer correlation that is a function of the properties of the liquid and the user-specified `HORIZONTAL_VELOCITY` or `VERTICAL_VELOCITY`. Details can be found in the FDS Technical Reference Guide chapter, “Lagrangian Particles,” section “Heating and Evaporation of Liquid Droplets.” Alternatively, you may specify a constant value on the `PART` line, `HEAT_TRANSFER_COEFFICIENT_SOLID`, in units of $\text{W/m}^2/\text{K}$.

⁴Solid particles do not stick to solid surfaces by default like liquid droplets. However, you can make solid particles act like liquid droplets by setting `ADHERE_TO_SOLID=1` on the `PART` line.

⁵If you do not want droplets to accumulate on solid surfaces, set `ADHERE_TO_SOLID=-1` on the `PART` line. It is normally 1 for liquid droplets.

There are some applications, like the suppression of racked storage commodity fires, where it is useful to allow water droplets to move horizontally along the underside of a solid object. It is difficult to model this phenomenon precisely because it involves surface tension, surface porosity and absorption, and complicated geometry. However, a way to capture some of the effect is to set `ALLOW_UNDERSIDE_PARTICLES=T` on the `SURF` line(s). It is normally false. Also, note that when droplets hit obstructions, the vertical direction is assumed to coincide with the z axis, regardless of any change to the gravity vector, `GVEC`. To stop particles or droplets from “sticking” to surfaces, set `ALLOW_SURFACE_PARTICLES=F` on the `SURF` line.

A useful sample case to demonstrate various features of droplet motion on solid obstructions is the test case called `Sprinklers_and_Sprays/cascade.fds`. The image to the left in Fig. 15.9 shows 1 L of water droplets cascading down the sides of an array of boxes. The plot to the right checks that the total water mass is conserved. Note that some of the water evaporates into the compartment that initially has zero humidity.

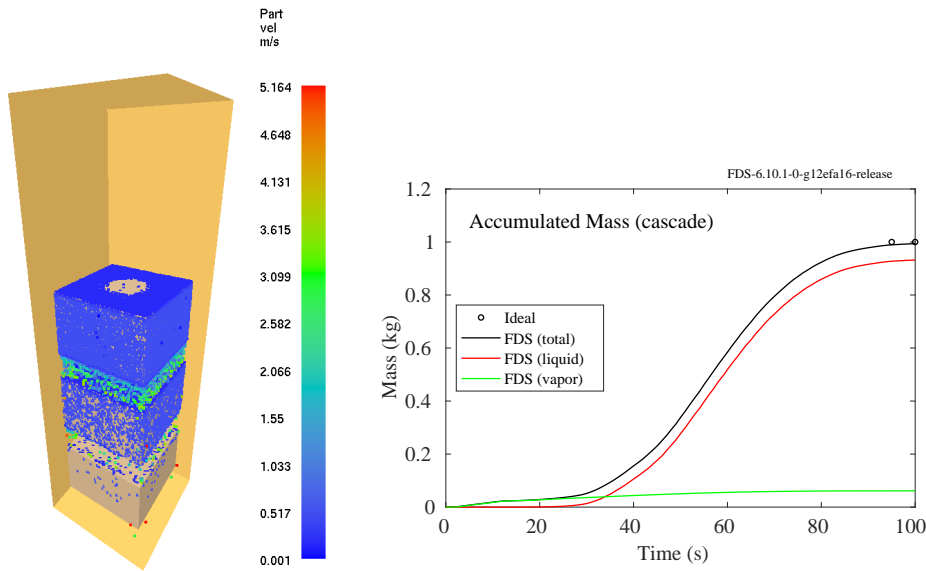


Figure 15.9: (Left) Smokeview rendering of the `cascade` test case, demonstrating that water droplets move randomly at 0.2 m/s over horizontal surfaces, and descend uniformly at 0.5 m/s over vertical surfaces. The plot at the right demonstrates that water mass is conserved in the simulation.

15.7.2 Reduction of the Burning Rate

Water reduces the fuel pyrolysis rate by cooling the fuel surface and also changing the chemical reactions that liberate fuel gases from the solid. If the solid or liquid fuel has been given reaction parameters via the `MATL` line, there is no need to set any additional suppression parameters. It is assumed that water impinging on the fuel surface takes energy away from the pyrolysis process and thereby reduces the burning rate of the fuel. If the surface has been assigned a `HRRPUA` (Heat Release Rate Per Unit Area), a parameter needs to be specified that governs the suppression of the fire by water because this type of simulated fire essentially acts like a gas burner whose flow rate is explicitly specified. An empirical way to account for fire suppression by water is to characterize the reduction of the pyrolysis rate in terms of an exponential function. The local mass loss rate of the fuel is expressed in the form

$$\dot{m}_f''(t) = \dot{m}_{f,0}''(t) e^{-\int k(t) dt} \quad (15.10)$$

Here $\dot{m}_{f,0}''(t)$ is the user-specified burning rate per unit area when no water is applied and k is a function of the local water mass per unit area, m_w'' , expressed in units of kg/m^2 .

$$k(t) = \text{E_COEFFICIENT} \dot{m}_w''(t) \quad 1/\text{s} \quad (15.11)$$

The parameter `E_COEFFICIENT` must be obtained experimentally, and it is expressed in units of $\text{m}^2/(\text{kg s})$. Usually, this type of suppression algorithm is invoked when the fuel is complicated, like a cartoned commodity. The simple example case `Sprinklers_and_Sprays/e_coefficient.fds` demonstrates the use of this parameter. A water nozzle is placed over a 0.6 m by 0.6 m gas burner defined with the `SURF` line shown below. The nozzle discharges 0.1 L of water in 0.1 s, 10 s after ignition of the fire. The water's mass per unit area is $0.1/0.36 \approx 0.278 \text{ kg}/\text{m}^2$. Figure 15.10 shows the heat release rate of the fire, decreasing from 36 kW to 0 kW in approximately 20 s. There are two cases considered—one where the burner and rim are constructed of conventional `OBSTs` and one where the burner is formed using immersed boundary `GEOMS`.

```
&SURF ID='FUEL', HRRPUA=100., E_COEFFICIENT=1. /
```

The expected HRR curve is:

$$\dot{Q}(t) = 36e^{-0.278(t-10.4)} \quad \text{kW} \quad (15.12)$$

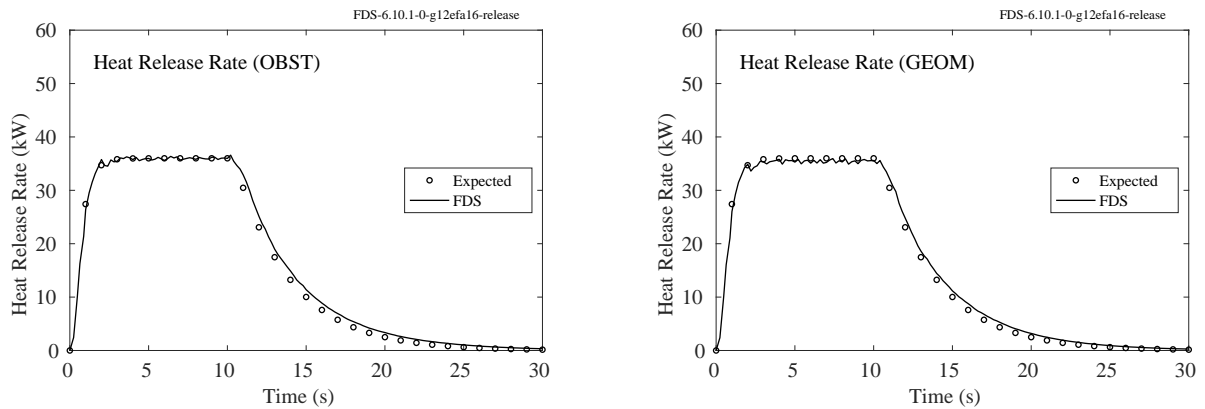


Figure 15.10: Output of the `e_coefficient` test case.

Chapter 16

Wind and Atmospheric Stratification

Most of the parameters that describe the atmosphere are entered on a single namelist line called `WIND`. There are various ways to specify a wind in FDS:

1. If you simply want to specify a wind speed and direction and perhaps vary these quantities in time and with height, see Sec. 16.1.
2. If the periphery of your computational domain is relatively flat and you have some knowledge of the surface roughness and atmospheric conditions, see Sec. 16.2 on how to apply Monin-Obukhov similarity theory.
3. If your domain spans kilometers and you would like to apply winds in terms of the pressure gradient and Coriolis forces, or you want to apply a geostrophic wind, see Sec. 16.3.
4. If you want to model the wind as one would do for a wind tunnel; that is, to create essentially a “wall of wind” where an entire side of the computational domain is turned into a giant fan blowing air laterally, see Sec. 16.4.

The temperature stratification of the atmosphere is discussed in Sec. 16.5. This is relevant unless you have chosen to apply wind and temperature profiles based on Monin-Obukhov theory. Finally, boundary conditions for outdoor simulations are discussed in Sec. 16.6.

16.1 Wind Method 1: Specified Wind Speed and Direction

The simplest way to introduce a wind in FDS is to specify its `SPEED` and `DIRECTION`. For example, for a southwesterly wind blowing 5 m/s, add the following line:

```
&WIND SPEED=5., DIRECTION=225. /
```

It is assumed here that the wind speed and direction do not change with height save for the effect of ground friction. The wind `DIRECTION` follows the usual meteorological convention—a northerly wind has direction of 0° and blows from north to south, or in the negative y direction in the FDS coordinate system. An easterly wind has a direction of 90° and blows from east to west, or in the negative x direction.

The wind `SPEED` and `DIRECTION` can be made functions of time and/or height using `RAMP` functions. With respect to time, consider the following lines of input:

```
&WIND SPEED=1., RAMP_SPEED_T='spd', RAMP_DIRECTION_T='dir' /
```

```

&RAMP ID='dir', T= 0, F=300 /
&RAMP ID='dir', T= 600, F=330 /
&RAMP ID='dir', T=1200, F=350 /
.
.
&RAMP ID='spd', T= 0, F=2.8 /
&RAMP ID='spd', T= 600, F=3.1 /
&RAMP ID='spd', T=1200, F=3.5 /
.
.

```

These lines direct FDS to vary the wind speed and direction over time, according to the `RAMP` functions 'spd' and 'dir', respectively. The parameter `T` is the time in s. For the direction ramp function, `F` represents the direction angle in degrees from the north (positive y direction). For the speed ramp function, `F` represents a multiplier of the base `SPEED`, which in this case is 1 m/s.

You can also vary the wind velocity components with height using the ramp functions `RAMP_SPEED_Z` and `RAMP_DIRECTION_Z`.

```

&WIND SPEED=1., RAMP_SPEED_Z='spd', RAMP_DIRECTION_Z='dir' /

&RAMP ID='dir', Z= 0, F=300 /
&RAMP ID='dir', Z= 200, F=330 /
&RAMP ID='dir', Z= 500, F=350 /
.
.
&RAMP ID='spd', Z= 0, F=1.0 /
&RAMP ID='spd', Z= 200, F=1.5 /
&RAMP ID='spd', Z= 500, F=1.8 /
.
.

```

The parameter `Z` represents the height (m). As with the time functions above, for the direction ramp function, `F` represents the direction angle in degrees from the north (positive y direction). This value is added to any time-varying value set by `RAMP_DIRECTION_T`. For the speed ramp function, `F` represents a multiplier of the base `SPEED`. This value is multiplied by any time-varying value set by `RAMP_SPEED_T`.

Often for longer simulations lasting hours, you may not know exactly how the wind `DIRECTION` varies, but you can provide an estimate based on the *stability class* of the atmosphere. Using the terminology of the atmospheric dispersion community [38], the perturbation to the wind `DIRECTION` can be modeled as:

$$\theta'(t + \delta t) = R^2 \theta'(t) + N\left(0, \sqrt{1 - R^2} \sigma_\theta\right) \quad ; \quad R = e^{-\delta t / \tau} \quad ; \quad \theta'(0) = 0 \quad (16.1)$$

where $N(\mu, \sigma)$ denotes a normal random variable with mean μ and standard deviation σ , σ_θ (`SIGMA_THETA` specified with units of degrees on the `WIND` line) depends on the stability class of the atmosphere and τ (`TAU_THETA`) is a time scale with a default value of 300 s.

Example

The input files called `wind_example_nn.fds` in the samples folder called `Atmospheric_Effects` demonstrate wind functionality. Each case considers a flat patch of terrain 1000 m by 1000 m by 100 m high. In the examples called `wind_example_5` and `wind_example_10`, a 5 m/s wind is turned 90° in 10 min. One case is run at 5 m resolution and other at 10 m. Figure 16.1 shows the delay in the mean velocity components due to the fact that the wind profile is imposed only at external boundaries.

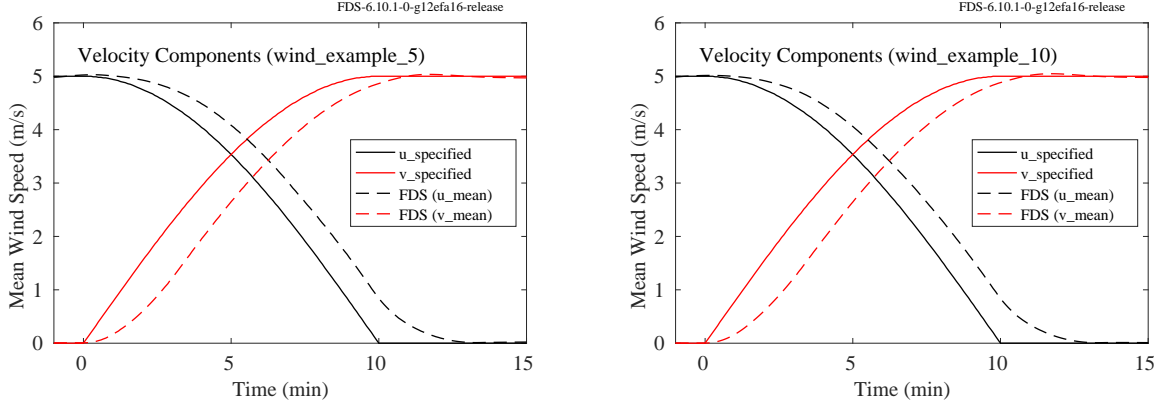


Figure 16.1: Mean velocity components of a 5 m/s wind turning 90° in 10 m. The grid resolutions are 5 m (left) and 10 m (right).

16.2 Wind Method 2: Monin-Obukhov Similarity

Monin-Obukhov similarity theory provides vertical wind and temperature profiles based on surface and atmospheric conditions. These profiles are applied at the exterior boundary of the computational domain; thus, the terrain must be relatively flat at the periphery. You might have a hill or valley within the domain, but the periphery should be flat to allow the wind field to develop naturally away from the boundary.

16.2.1 Basic Equations

It is assumed that the wind speed profile, u , and potential temperature¹, θ , vary with height, z , according to:

$$u(z) = \frac{u_*}{\kappa} \left[\ln \left(\frac{z}{z_0} \right) - \Psi_m \left(\frac{z}{L} \right) \right] \quad (16.2)$$

$$\theta(z) = \theta_0 + \frac{\theta_*}{\kappa} \left[\ln \left(\frac{z}{z_0} \right) - \Psi_h \left(\frac{z}{L} \right) \right] \quad (16.3)$$

where u_* is the friction velocity, $\kappa = 0.41$ is the Von Kármán constant, z_0 is the *aerodynamic* roughness length, θ_* is the scaling potential temperature, θ_0 is the ground level potential temperature, L is the Obukhov length, and the similarity functions are those proposed by Dyer [39, 40]:

$$\Psi_m \left(\frac{z}{L} \right) = \begin{cases} -5 \frac{z}{L} & : L \geq 0 \\ 2 \ln \left[\frac{1+\zeta}{2} \right] + \ln \left[\frac{1+\zeta^2}{2} \right] - 2 \tan^{-1}(\zeta) + \frac{\pi}{2} & : L < 0 \end{cases} \quad (16.4)$$

¹The potential temperature is given by

$$\theta = T \left(\frac{p_0}{p} \right)^{R/(W_{\text{air}} c_p)}$$

where p_0 is typically 100 kPa and $R/(W_{\text{air}} c_p) \approx 0.286$.

$$\Psi_h\left(\frac{z}{L}\right) = \begin{cases} -5\frac{z}{L} & : L \geq 0 \\ 2 \ln \left[\frac{1+\zeta^2}{2} \right] & : L < 0 \end{cases} \quad \zeta = \left(1 - \frac{16z}{L}\right)^{1/4} \quad (16.5)$$

The Obukhov length, L , characterizes the thermal stability of the atmosphere. When L is negative, the atmosphere is unstably stratified; when positive, the atmosphere is stably stratified. The stabilizing or destabilizing effects of stratification are strongest as L nears zero. Accordingly, a neutrally stratified atmosphere would have an infinite Obukhov length. Generally, an unstable atmosphere exhibits a decreasing temperature with height and relatively large fluctuations in wind direction/velocity. Unstable atmospheres are strongly affected by the buoyancy-generated turbulence, resulting in enhanced mixing. Conversely, highly stable atmospheric conditions suppress turbulent mixing.

In the event that these various parameters are not reported or known, they can be estimated from the basic meteorological conditions. From just a single measured mean wind velocity, u_{ref} , taken at a height, z_{ref} , the friction velocity can be calculated from:

$$u_* = \frac{\kappa u_{\text{ref}}}{\ln(z_{\text{ref}}/z_0) - \Psi_m(z_{\text{ref}}/L)} \quad (16.6)$$

The Obukhov length, L , can be chosen from Table 16.1. Suggested values of the aerodynamic roughness length², z_0 , are given in Table 16.2.

Table 16.1: Suggested values of the Obukhov length, L (m).

Stability	Value Range	Suggested Value
Very Unstable	$-200 \leq L < 0$	-100
Unstable	$-500 \leq L < -200$	-350
Neutral	$ L > 500$	1000000
Stable	$200 < L \leq 500$	350
Very Stable	$0 < L \leq 200$	100

Table 16.2: Davenport-Wieringa roughness length classification [41].

z_0 (m)	Classification	Landscape
0.0002	sea	sea, paved areas, snow-covered flat plain, tidal flats, smooth desert
0.005	smooth	beaches, pack ice, snow-covered fields
0.03	open	grass prairie, farm fields, tundra, airports, heather
0.1	roughly open	low crops and occasional obstacles (single bushes)
0.25	rough	high crops, scattered obstacles such as trees or hedgerows, vineyards
0.5	very rough	mixed farm fields and forest clumps, orchards, scattered buildings
1.0	closed	suburbs, villages, forests
>2	chaotic	large towns and cities, irregular forests

WARNING: It is not recommended to use MO theory in highly resolved LES where the grid resolution near the surface is smaller than the sand grain roughness [42]. In this situation, FDS will fall back on using a `NO_SLIP` boundary to compute the wall stress.

²Note that *aerodynamic* roughness, z_0 , is not the same as *sand grain* roughness, s , discussed in Sec. 10.1.7. For more information, see Sec. 16.3.4.

The scaling potential temperature, θ_* , can be obtained from the relation:

$$\theta_* = \frac{u_*^2 \theta_0}{g \kappa L} \quad ; \quad \theta_0 = \theta_{\text{ref}} / \left(1 + \frac{u_*^2}{g \kappa^2 L} [\ln(z_{\text{ref}}/z_0) - \Psi_h(z_{\text{ref}}/z_0)] \right) \quad (16.7)$$

Alternatively, the scaling potential temperature can be estimated from the expression for the sensible heat flux:

$$\dot{q}_c'' = -\rho c_p u_* \theta_* \quad (16.8)$$

Note that a positive value of the heat flux indicates that the ground is warmer than the air above.

Figure 16.2 displays a few sample wind profiles. Notice that the sign of L determines whether the atmosphere is stable or unstable; that is, whether the temperature increases or decreases relative to the ground temperature.

16.2.2 Applying Monin-Obukhov Profiles to FDS

The following input line specifies a southwesterly wind blowing 5 m/s over a relatively flat, rural landscape on a bright, sunny day:

```
&WIND SPEED=5., Z_REF=3, DIRECTION=225., L=-100., Z_0=0.03 /
```

Setting a non-zero value of L indicates that you want to use Monin-Obukhov profiles. The parameter `SPEED` specifies the wind speed at a distance `Z_REF` (m) above the ground. The default value of `Z_REF` is 10 m. You can change the height of the ground by setting `GROUND_LEVEL`, which is at $z = 0$ m by default. It is assumed that the temperature at `Z_REF` is the specified ambient, `TMPA`, unless you specify `TMP_REF` ($^{\circ}\text{C}$). The wind `DIRECTION` follows the usual meteorological convention—a northerly wind has direction of 0° and blows from north to south, or in the negative y direction in the FDS coordinate system. An easterly wind has a direction of 90° and blows from east to west, or in the negative x direction. You can vary the direction of the wind in time only, not in space:

```
&WIND ..., RAMP_DIRECTION_T='dir' /

&RAMP ID='dir', T= 0, F=300 /
&RAMP ID='dir', T= 600, F=330 /
&RAMP ID='dir', T=1200, F=350 /
```

Here, T is time in seconds and F is the wind direction in degrees. You may not vary the speed in time or space.

If you know the value of u_* or θ_* , you can input these on the `WIND` line as `U_STAR` (m/s) and `THETA_STAR` (K). Otherwise, they will be computed from L and z_0 . Do not input all four of these parameters together.

Also, do not specify parameters on the `SURF` line that defines the ground that might not be consistent with the specified M-O parameters. A fixed temperature is usually sufficient given that the atmospheric profile is being imposed at the boundaries.

Example

The input files called `wind_example_nn.fds` in the `samples` folder called `Atmospheric_Effects` demonstrate wind functionality. Each case considers a flat patch of terrain 1000 m by 1000 m by 100 m high. In the example called `wind_example_32` there are various rectangular blocks scattered about the domain to represent buildings and the wind is held steady. Figure 16.3 displays the specified wind speed and temperature profiles in the vertical direction compared to the computed time average profiles. The profiles should be similar, but not exactly the same.

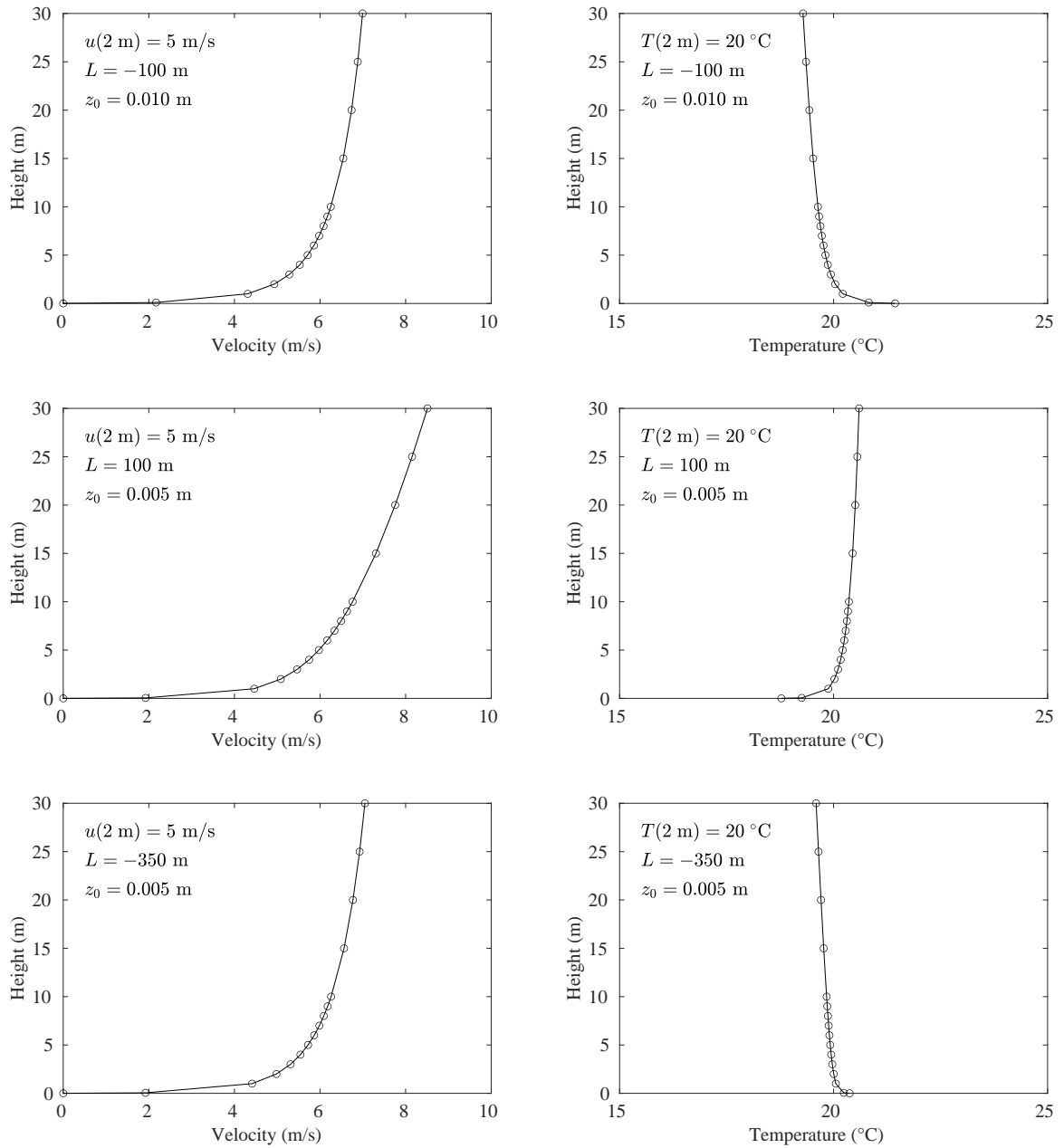


Figure 16.2: Sample vertical wind and temperature profiles.

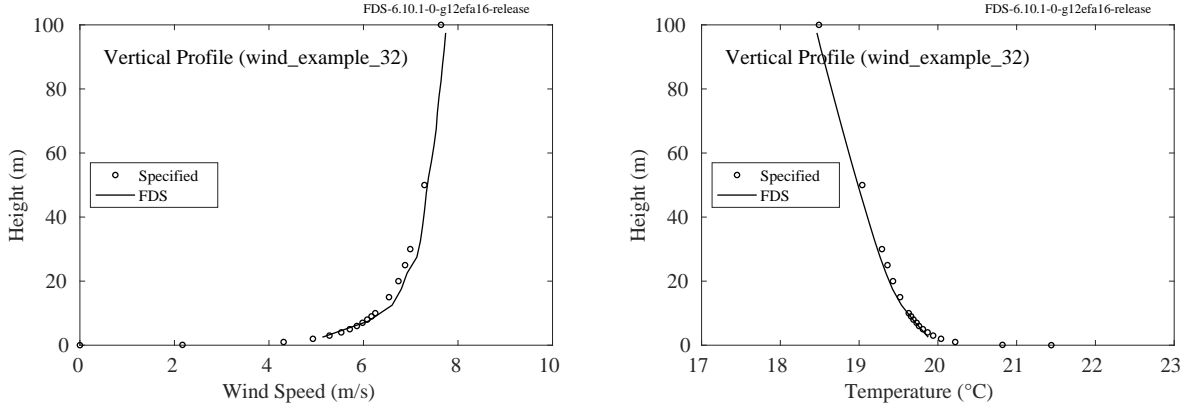


Figure 16.3: Simulated vs. specified vertical wind speed and temperature profiles.

16.3 Wind Method 3: Advanced Meteorological Concepts

If your computational domain spans kilometers, you might consider a more fundamental set of parameters to describe the wind field. This is a challenge because it is difficult to infer far-field meteorological conditions from a few near-field, near-surface wind or temperature readings that you might have from a few weather stations. If you pursue this course, you will most likely need to use some trial and error to ensure that your far-field conditions yield something close to the data obtained at the weather stations.

16.3.1 Pressure Gradient Force

Winds are driven by horizontal pressure gradients that push air masses from regions of high to low surface pressure. This optional *pressure gradient force*, \mathbf{F} , is an extra force term added to the momentum equation:

$$\frac{\partial \mathbf{u}}{\partial t} + \dots = \mathbf{F}/\rho \quad (16.9)$$

The magnitude of the pressure gradient force can be related to the average wind speed, U , over the height of the boundary layer, H , by [41]

$$|\mathbf{F}| \approx \rho C_D \frac{U^2}{H} \quad (16.10)$$

where C_D is a dimensionless drag coefficient ranging between 2×10^{-3} for relatively smooth surfaces to 2×10^{-2} for rough or forested surfaces. This relation is appropriate for neutrally stable conditions, i.e. windy and relatively low surface heating. For unstable conditions, i.e., light winds and strong surface heating, the relation becomes

$$|\mathbf{F}| \approx \rho b_D \sqrt{gH \frac{\Delta\theta}{\theta}} \frac{U}{H} \quad (16.11)$$

where $b_D = 1.83 \times 10^{-3}$ and $\Delta\theta/\theta$ is the relative change in potential temperature between the ground and the mid-height of the boundary layer. Note that these relationships are approximate. You need to conduct some numerical experiments if you want to match a particular wind measurement.

The `PRESSURE_GRADIENT_FORCE` is input on the `WIND` line in units of Pa/m, for example,

```
&WIND PRESSURE_GRADIENT_FORCE=0.01, DIRECTION=315 /
```

The wind `DIRECTION` follows the usual meteorological convention—a northerly wind has direction of 0° and blows from north to south, or in the negative y direction in the FDS coordinate system. An easterly wind has a direction of 90° and blows from east to west, or in the negative x direction. You can vary the direction of the wind in time only, not in space:

```
&WIND ..., RAMP_DIRECTION_T='dir' /
&RAMP ID='dir', T= 0, F=300 /
&RAMP ID='dir', T= 600, F=330 /
&RAMP ID='dir', T=1200, F=350 /
```

Here, T is time in seconds and F is the wind direction in degrees. You may vary the `PRESSURE_GRADIENT_FORCE` in time using `RAMP_PGF_T`.

There are some applications, like tunnels, where the pressure gradient force is a convenient way to introduce an air flow due to some external force. In such cases, it is sometimes more convenient to use the vector `FORCE_VECTOR(1:3)` along with the corresponding time ramps `RAMP_FVX_T`, `RAMP_FVY_T`, and `RAMP_FVZ_T` to control each individual component. The `PRESSURE_GRADIENT_FORCE` is simply the magnitude of the `FORCE_VECTOR` with the same units.

16.3.2 Coriolis Force

The Coriolis force accounts for a rotating reference frame [41]. The Coriolis acceleration augments the right-hand side of the Navier-Stokes equations as follows:

$$\frac{D\mathbf{u}}{Dt} = \dots - 2\boldsymbol{\Omega} \times \mathbf{u} \quad (16.12)$$

Where $\boldsymbol{\Omega}$ is the rotation vector (see Fig. 16.4). To apply the Coriolis force in FDS, you have two options: either specify the components of $\boldsymbol{\Omega}$ directly on the `WIND` lines via `CORIOLIS_VECTOR(3)`, or you can take the easy route and simply specify the `LATITUDE` (degrees) of the domain, provided you align the x direction with east. In this latter case, FDS will compute the rotation vector for you as

$$\boldsymbol{\Omega} = \omega \begin{bmatrix} 0 \\ \cos(\phi\pi/180) \\ \sin(\phi\pi/180) \end{bmatrix} \quad (16.13)$$

where ω is Earth's rotation rate, about $1.16 \times 10^{-5} 2\pi/\text{s}$.

The non-dimensional number that determines if the Coriolis force is significant is the Rossby number. Let U be the relevant velocity scale and L the length scale of the problem. The Coriolis frequency is given by $f_c = 2\omega \sin(\phi\pi/180)$. The Rossby number is then given by

$$\text{Ro} = \frac{U}{f_c L} \quad (16.14)$$

When the Rossby number is large, the Coriolis force is negligible. In building fires, this is typically the case, as U and L are of order unity and $f_c = \mathcal{O}(10^{-4} \text{ rad/s})$.

16.3.3 Geostrophic Wind

The *geostrophic wind* is a theoretical construct obtained from the lateral momentum equations when the Coriolis force is in balance with the horizontal pressure gradient at steady state [41]. The horizontal wind components can be written as

$$U_g = -\frac{1}{\rho f_c} \frac{\partial \bar{p}}{\partial y} \quad (16.15)$$

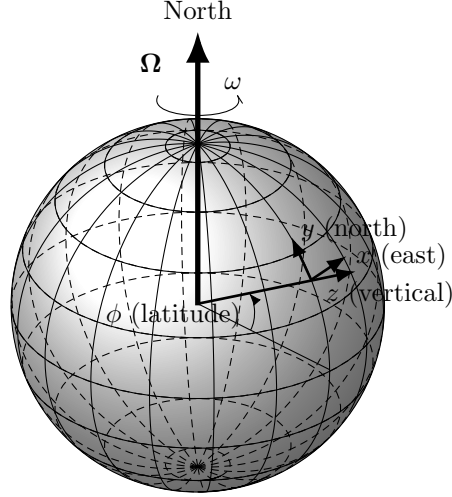


Figure 16.4: The rotation vector Ω points from the Earth's center to the North Pole. To apply the effect of rotation on our model, we need to know the components of the rotation vector in the reference frame of the computational domain. Typically, we align the computational domain so that x points east, y points north, and z represents elevation. The angle ϕ [degrees] is the latitude, positive $[0^\circ, 90^\circ]$ in the northern hemisphere, negative $[0^\circ, -90^\circ]$ in the southern hemisphere. The rotation rate ω is (obviously) 1 rotation per day or $2\pi/(24 \times 3600) \approx 1.16 \times 10^{-5} \text{ } 2\pi/\text{s}$.

$$V_g = + \frac{1}{\rho f_c} \frac{\partial \bar{p}}{\partial x} \quad (16.16)$$

If the geostrophic wind speed is known, it can be entered on the `WIND` line via `GEOSTROPHIC_WIND(1:2) = (Ug, Vg)`. Notice that specification of the wind in x implies a mean pressure gradient in y and vice versa. When a geostrophic wind is specified, FDS converts this value to the implied `FORCE_VECTOR` according to Eq. (16.15). To achieve the specified geostrophic wind aloft, you must also have the Coriolis force turned on (see Sec. 16.3.2). Since the geostrophic wind is equivalent to applying `FORCE_VECTOR`, you may also apply a time ramp (see Sec. 16.3.1).

16.3.4 Surface Roughness

The default wall model used in FDS, in the “fully rough” limit, is given by the following log law:

$$\frac{u(z)}{u_*} = \frac{1}{\kappa} \ln \left(\frac{z}{s} \right) + 8.5 \quad (16.17)$$

The *sand grain* roughness, s , is related to the *aerodynamic* roughness length, z_0 , discussed in Sec. 16.2. Substituting the velocity profile given by Monin-Obukhov similarity theory, Eq. (16.2), into Eq. (16.17), and noting that $\psi_m \approx 0$ near the ground, the effective translation of roughness factors is

$$s = z_0 e^{8.5\kappa} \approx 32.6 z_0 \quad (16.18)$$

The sand grain roughness, s , is specified on a `SURF` line via the parameter `ROUGHNESS` (m). This parameter has a more intuitive meaning than its counterpart in M-O similarity theory, z_0 , because it can be viewed as the characteristic height of ground level obstructions. Table 16.2 presents suggested values of z_0 , and Eq. 16.18 can be used to compute the `ROUGHNESS` length that is input on the `SURF` line that defines the ground.

16.3.5 Thermal Boundary Conditions at the Ground

The ground temperature relative to the air temperature is an important consideration in defining the stability of the atmospheric boundary layer. Cold air blowing over warm ground leads to greater dispersion than warm air blowing over cold ground. You can apply a variety of thermal boundary conditions on the `SURF` line that defines the ground: (1) a fixed or time-varying temperature using `WALL_TEMPERATURE` (°C), (2) a `CONVECTIVE_HEAT_FLUX` (kW/m²), or (3) parameters that define the ground as a thermally-thick solid. Setting the `CONVECTIVE_HEAT_FLUX` is convenient if you want to make contact with Monin-Obukhov similarity theory, where Eqs. (16.7) and (16.8) can be combined to provide an expression for the convective heat flux at the surface:

$$\dot{q}_c'' = -\frac{u_*^3 \theta_0 \rho c_p}{g \kappa L} \quad (16.19)$$

Note that ρ (kg/m³) is the density and c_p is the specific heat of the air near the ground.

16.3.6 Example

Figure 16.5 displays velocity and temperature profiles generated by FDS over a 1000 m square domain with periodic boundaries and a height of 200 m. The wind fields are generated using pressure gradient forces, F , of various values, and the ground is given several different values of `CONVECTIVE_HEAT_FLUX` (\dot{q}_c'') and surface `ROUGHNESS` (s). Eqs. (16.18) and (16.19) are used to convert the specified \dot{q}_c'' and s to L and z_0 that are then used to generate Monin-Obukhov velocity and temperature profiles with which to compare the simulations. Note that these simulations do not invoke the Monin-Obukhov profiles directly. Rather, the M-O profiles are used to test if the FDS simulations produce realistic vertical profiles using just a specified pressure gradient force, F , surface roughness, s , and surface heat flux, \dot{q}_c'' .

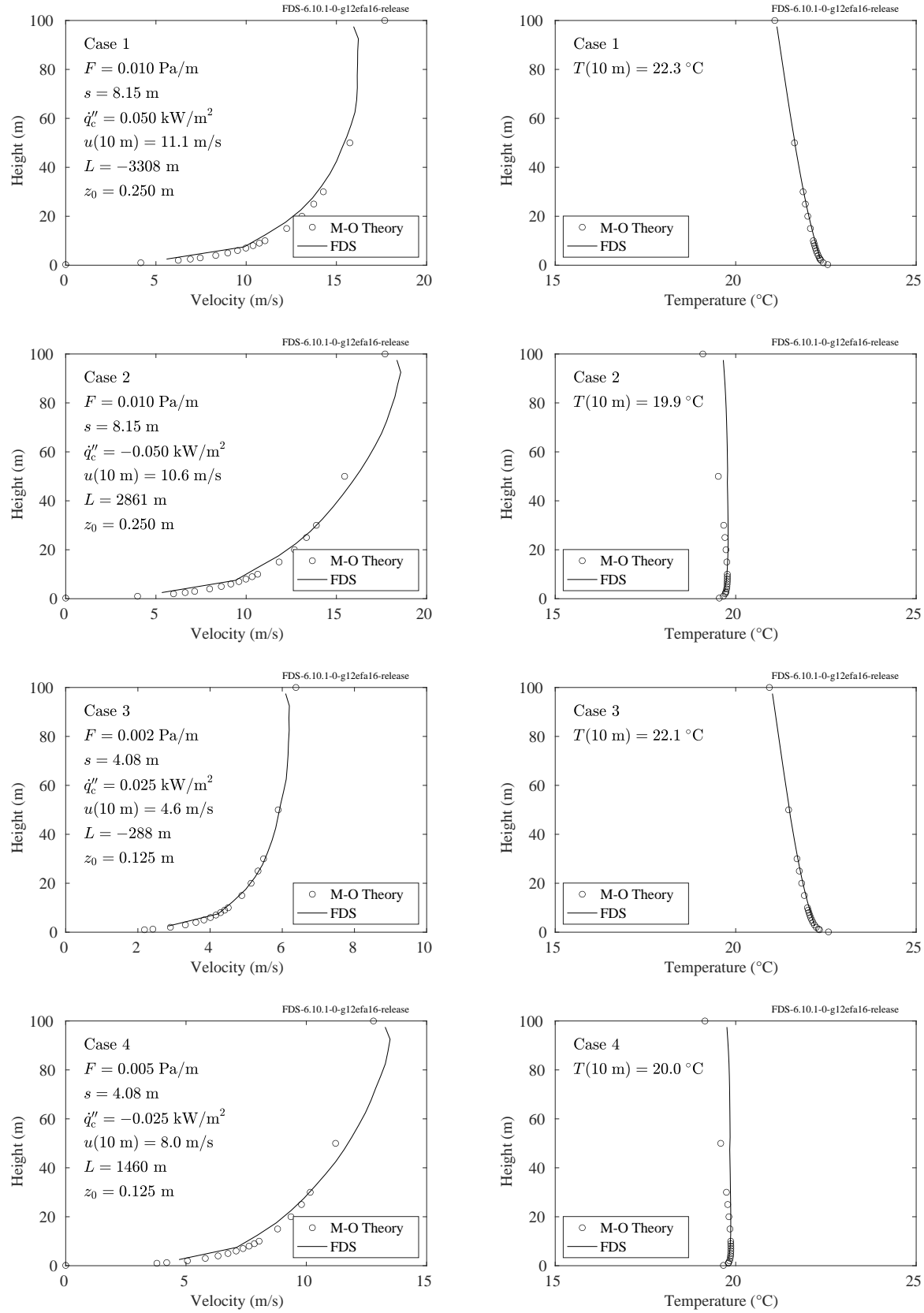


Figure 16.5: Comparison of FDS-predicted velocity and temperature profiles compared to those generated using Monin-Obukhov similarity theory.

16.4 Wind Method 4: The “Wall of Wind”

An alternative to using Monin-Obukhov similarity theory in specifying a wind is to specify a power law wind profile at an external boundary of the computational domain. This essentially creates a “wall of wind” and for early versions of FDS it was the recommended method for specifying a wind. However, the techniques described above are preferable because they handle the lateral boundaries of the computational domain in a more natural way.

To specify a “wall of wind”, set `PROFILE='ATMOSPHERIC'` on the `SURF` line that is assigned to an exterior lateral boundary of the computational domain. This generates a power law atmospheric wind profile of the form $u = u_0(z/z_0)^p$ where z is the height above the ground. If an atmospheric profile is prescribed, also prescribe `Z0` for z_0 and `PLE` for p . `VEL` specifies the reference velocity u_0 . Note that z_0 is not the ground, but rather the height above the ground where the wind speed is measured, like an elevated weather station. It is assumed that the ground is located at 0 m; to change this assumption, set `GROUND_LEVEL` on the `WIND` line to be the appropriate elevation. Be careful not to apply an atmospheric velocity profile (e.g. negative z) below `GROUND_LEVEL` or FDS will stop with an error.

When using the “wall of wind” approach, be aware that the wind field requires time to develop. To speed things along, you may want to initialize the flow to the chosen wind speed throughout the domain, but you may not want this initial wind field to persist indefinitely. In such cases, add the line:

```
&WIND U0=..., V0=..., W0=... /
```

where `U0`, `V0`, and `W0` are your desired initial wind field. Setting the velocity components directly will initialize the simulation with a constant flow field, but the flow field will die off and the `SURF` line with the atmospheric profile will govern the wind flow.

16.5 Temperature Stratification

Note: If you have chosen to apply Monin-Obukhov wind and temperature profiles, this section is not applicable.

Typically, in the first few hundred meters of the atmosphere, the temperature decreases several degrees Celsius per kilometer. This small temperature change is important when considering the rise of smoke since the temperature of the smoke decreases rapidly as it rises. The `LAPSE_RATE` of the atmosphere can be specified on the `WIND` line in units of °C/m. A negative sign indicates that the temperature *decreases* with height. This need only be set for outdoor calculations where the height of the domain is tens or hundreds of meters. The default value of the `LAPSE_RATE` is 0 °C/m.

Alternatively, you can specify `RAMP_TMP0_Z` on the `WIND` line if you want a non-linear change in temperature with height. For example,

```
&WIND ..., RAMP_TMP0_Z='T profile', ... /

&RAMP ID='T profile', Z= 0.001 , F= 1.0258 /
&RAMP ID='T profile', Z= 0.500 , F= 1.0019 /
&RAMP ID='T profile', Z= 1.000 , F= 1.0000 /
&RAMP ID='T profile', Z= 1.900 , F= 0.9986 /
&RAMP ID='T profile', Z= 3.000 , F= 0.9977 /
&RAMP ID='T profile', Z= 4.000 , F= 0.9972 /
```

The value of `z` is the height (m) above the ground. The value of `F` is the ratio of the temperature at the given height in degrees K to the ambient temperature in degrees K, `TMFA+273`.

By default, FDS assumes that the density and pressure decrease with height, regardless of the application or domain size. For most simulations, this effect is negligible, but it can be turned off completely by setting `STRATIFICATION=F` on the `WIND` line.

16.5.1 Stack Effect

Tall buildings often experience buoyancy-induced air movement due to temperature differences between the interior and exterior, known as *stack effect* [43]. These temperature differences create flows within vertical shafts (stairwells, atria, elevator shafts, etc.) due to leaks or openings at different levels. To simulate this phenomenon in FDS, you must include the entire building, or a substantial fraction of it, both inside and out, in the computational domain. It is important to capture the pressure and density decrease in the atmosphere based on the specified temperature profile that is entered on the `WIND` line.

For the case where the stack flow is through small leakage paths, divide the building into one or more pressure `ZONES`. The leakage paths can be defined in terms of HVAC components. Note that the leakage model combines all leaky surfaces over the entire height of the building and as a result averages out the pressure gradients. For doing stack effect calculations individual leakage paths should be defined. A simple example is described next.

Example Case: `Atmospheric_Effects/stack_effect`

This test case is a two-dimensional simulation of a 100 m tall building whose interior air temperature is slightly warmer than its exterior. The building has leakage paths at the top and ground floors only. Since the inside air temperature is slightly warmer, the inside air pressure is slightly higher as well, and it drives air out of the building and in turn draws air into the building at the ground level. The interior air temperature, T_b , is initially 20 °C (293 K), and the exterior air temperature, T_∞ , is 10 °C (283 K). The `LAPSE_RATE` is set to 0 °C/m; thus, $T_0(z) = T_\infty$ outside the building and $T_0(z) = T_b$ inside the building. Two small leakage

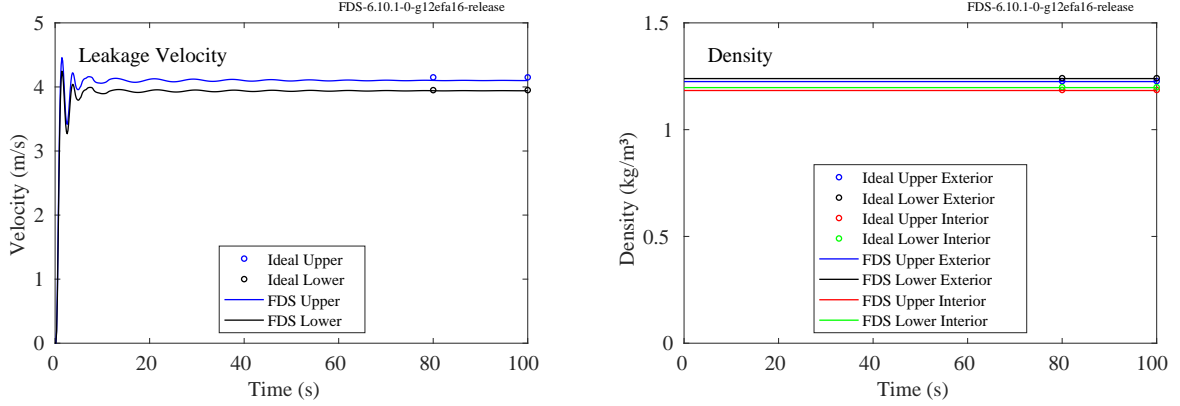


Figure 16.6: (Left) Velocity at the upper and lower vents for the `stack_effect` case. (Right) Upper and lower exterior and interior densities.

openings are defined 2.5 m above the ground floor and 2.5 m below the roof using the HVAC solver. Each opening is given an area of 0.01 m^2 and a loss coefficient of 2 (e.g., an entrance loss into the leak path of 1 and an exit loss out of the leak path of 1 both of which represent a sharp edge opening).

The initial density stratification inside and outside the building can be calculated using the relation:

$$\frac{\rho_0(z)}{\rho_\infty} = \exp\left(-\frac{g\bar{W}}{RT_0}z\right) \quad (16.20)$$

where R is the universal gas constant, g is the acceleration of gravity, and \bar{W} is the average molecular weight of the air, z is the height above the `GROUND_LEVEL`, and T_0 is the ambient temperature. Applying this formula to the external and internal locations at the lower and upper vents results in densities of 1.2412, 1.1989, 1.2272, and 1.1858 kg/m^3 , respectively.

Since the openings in the building are equally spaced over its height, the neutral plane should be close to its midpoint. This can be computed from:

$$\frac{\Delta H}{H_n} = 1 + \frac{T_b}{T_\infty} \quad (16.21)$$

where H_n is the neutral plane height above the bottom vent and $\Delta H = 95 \text{ m}$ is the distance between the leak points. This gives a neutral plane of 46.68 m above the lower vent or 49.18 m above the bottom of the building. Note that this is close to the midpoint value of 50 m above the bottom of the building. The pressure difference across the building's wall is computed from

$$\Delta p = \frac{\bar{W} p_0(z) g \Delta z}{R} \left(\frac{1}{T_\infty} - \frac{1}{T_b} \right) \quad (16.22)$$

where Δz is the distance from the leak point to the neutral plane. Using the neutral plane location, the Δz values are -46.68 m for the lower vent and +48.32 m for the upper vent which respectively result in lower and upper vent pressure differences of -19.4 Pa and +20.4 Pa. Using the loss of 2 and the pressure difference in the HVAC momentum equation results in a steady-state inflow velocity at the bottom of 3.95 m/s and an outflow velocity at the top of 4.15 m/s. Results for velocity and density are shown in Fig. 16.6.

16.6 External Boundary Conditions

For typical outdoor simulations, the lateral exterior and upper boundaries of the computational domain are either open or periodic, and the lower boundary represents the earth. Do not change the gravity vector when doing outdoor simulations, as there are many assumptions made that assume that the ground is down and the sky is up.

FDS assumes the ground to be ambient temperature, `TMPA`, initially, and the temperature of the atmosphere either increases or decreases with height according to the stability condition. You can set the ground temperature to a different temperature using `TMP_FRONT`, or you provide material properties and allow the ground to heat up or cool down naturally. You may also set a `CONVECTIVE_HEAT_FLUX`, where a positive value means that the ground is warmer than the air above.

Usually, the top of the domain is set to be `'OPEN'`, meaning that even though the wind field is more or less parallel to it, there might be, for example, a hot plume that can rise out of the computational domain.

The lateral boundaries of the domain can be set either to `SURF_ID='OPEN'`, `'PERIODIC'`, or `'PERIODIC FLOW ONLY'`. `SURF_ID='OPEN'` is the usual inflow/outflow condition for any case where the computational domain is open to an infinite ambient void. `SURF_ID='PERIODIC'` assumes that the domain repeats itself endlessly in each direction. This is an important assumption because the turbulent atmospheric boundary layer that is modeled using Monin-Obukhov similarity theory is assumed to develop over distances far beyond the computational domain. `SURF_ID='PERIODIC FLOW ONLY'` is the same as `'PERIODIC'` except that only the velocity field is periodic, but scalar quantities are not. That is, you do not want a smoke plume that exits the eastern boundary of the domain to reappear instantly at the western boundary, even though you still might want the velocity field to be periodic.

Chapter 17

Wildland Fire Spread

There are three ways of simulating wildland fire spread in FDS:

1. **Particle Model:** The vegetation is represented by a collection of Lagrangian particles that are heated via convection and radiation (Sec. 17.2).
2. **Boundary Fuel Model:** Ground vegetation is modeled like a porous solid with a thickness equal to the height of the vegetation (Sec. 17.3).
3. **Level Set Model:** The fire front propagates using purely empirical rules (Sec. 17.5).

The Particle Model and Boundary Fuel Model use the same basic pyrolysis model (Sec. 17.1), and the fire spread rate is *predicted* by the model. The Level Set Model relies on a set of experimentally-determined spread rates for different types of vegetation and wind speeds.

17.1 Thermal Degradation Model for Vegetation

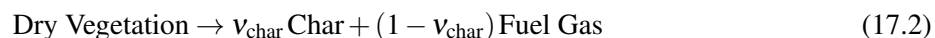
17.1.1 Solid Phase

The solid-phase thermal degradation process for generic vegetation are typically modeled with three reactions (note that the stoichiometric coefficients are mass-based) [44, 45, 46]:

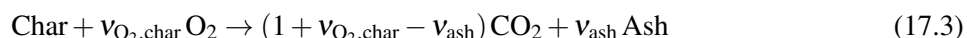
1. Endothermic moisture evaporation



2. Endothermic pyrolysis of Dry Vegetation



3. Exothermic char oxidation

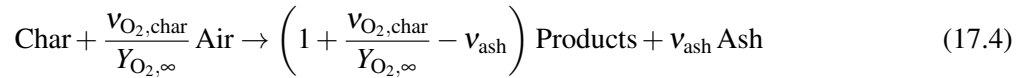


M is the vegetation *moisture content* or *moisture fraction* determined on a dry weight basis, specified with `MOISTURE_FRACTION` on the `SURF` line. v_{char} is the mass fraction of Dry Vegetation that is converted to char during pyrolysis, specified with the parameter `NU_MATL` on the `MATL` line that describes the Dry Vegetation. The character string `MATL_ID` on the same `MATL` line indicates the name of the char. $v_{\text{O}_2, \text{char}}$ is the mass of

oxygen consumed per unit mass of char oxidized. v_{ash} is the mass fraction of char that is converted to ash during char oxidation, specified by `NU_MATL` on the `MATL` line describing the char.

It is assumed that the Dry Vegetation in Eq. (17.2) is 47 % (by mass) carbon [47] with an effective organic component $\text{C}_{3.4}\text{H}_{6.2}\text{O}_{2.5}$ [48]. In general, it is assumed that char may be comprised of more than pure carbon and is defined as $\text{C}_x\text{O}_z\text{A}$ in Eq. 17.23. In the specific case where char is composed of pure carbon which reacts completely with O_2 to form CO_2 then $v_{\text{O}_2,\text{char}} = 2.67$ and $v_{\text{ash}} = 0$. A full discussion of the composition of the Char and Fuel Gas is given in Sec. 17.1.2.

The char reaction, Eq. (17.3), is usually modified in FDS when the default “simple chemistry” model is used; that is, the one-step, mixing-controlled reaction for the lumped species Fuel, Air, and Products. In this case, O_2 and CO_2 are not explicitly calculated, but rather are implicitly defined in terms of the lumped species, `AIR` and `PRODUCTS`, which are defined based on the stoichiometry of the reaction. Equation (17.3) cannot be written in terms of the lumped species, but a close approximation is as follows:



In the input file, this reaction is specified by the following parameters on the `MATL` line that defines the char:

```
&MATL ID          = 'char'
...
MATL_ID           = 'ash'
NU_MATL           = 0.02
SPEC_ID           = 'PRODUCTS', 'AIR'
NU_SPEC           = 8.15, -7.17 /
```

What this means is that the decomposition of 1 g of char requires 7.17 g of Air and produces 8.15 g of Products and 0.02 g of Ash. The Products in this case are the products of the single step gas phase reaction, not just CO_2 . The discrepancy is minor, and the alternative to using this approximation is to explicitly track all gas species and apply Eq. (17.3) directly. If O_2 is not explicitly tracked FDS will infer $v_{\text{O}_2,\text{char}}$ in Eq. 17.4 from the mass fraction of O_2 in the reactant (e.g. Air). Assuming $Y_{\text{O}_2,\infty} = 0.23$, this example implies $v_{\text{O}_2,\text{char}} = 1.65$, which is a common value used for vegetative material [44]. An oxygen mass fraction of 0.23 is usually a good approximation for Air when converting between the lumped species and oxygen stoichiometric coefficients. The actual mass fractions of the components of a lumped species can always be confirmed by checking the `CHID.out` file.

In the referenced papers [44, 45, 46], the reaction rates are written in terms of “bulk” quantities. For example, in the paper by Mell et al. [49], the mass of dry vegetation per unit volume is denoted $\langle m_{\text{dry}}''' \rangle_{V_b}$, where the angled brackets denote the explicit LES filtering over the grid cell volume V_b . However, in FDS the reaction rates are written in terms of the component densities of the composite solid:

$$\rho_s = \rho_{s,\text{dry}} + \rho_{s,\text{H}_2\text{O}} + \rho_{s,\text{char}} + \rho_{s,\text{ash}} \quad (17.5)$$

The reaction rates for evaporation of H_2O and pyrolysis of the dry vegetation are:

$$r_{\text{H}_2\text{O}} = \rho_{s,\text{H}_2\text{O}} A_{\text{H}_2\text{O}} T^{-\frac{1}{2}} \exp\left(-\frac{E_{\text{H}_2\text{O}}}{RT}\right) \quad (17.6)$$

$$r_{\text{pyr}} = \rho_{s,\text{dry}} A_{\text{pyr}} \exp\left(-\frac{E_{\text{pyr}}}{RT}\right) \quad (17.7)$$

where T is the temperature of the vegetation. Note that the exponent $-1/2$ in Eq. (17.6) is specified using `N_T=-0.5` on the `MATL` line for water, and the exponent 0 in Eq. (17.8) is specified using `N_S=0` on the `MATL` line for the char species. The default values of the kinetic constants are given in Table 17.1.

In the case of char the surface oxidation rate is determined by [50]:

$$r_{\text{char}} = Y_{\text{O}_2, \text{surf}} \sigma A_{\text{char}} \exp\left(-\frac{E_{\text{char}}}{RT}\right) \quad (17.8)$$

This model is invoked by setting `SURFACE_OXIDATION_MODEL` to `T` on the `MATL` line. Use of this model automatically assumes the reaction is zeroth order for the solid phase density ($N_{\text{S}}=0$) and first order for the oxygen mass fraction ($N_{\text{O}_2}=1$), and FDS will override any values provided by the user. Suggested kinetic constants are given in Table 17.1. The reaction rate depends on the surface area to volume ratio of the solid, σ . It also depends on the oxygen mass fraction at the material surface, $Y_{\text{O}_2, \text{surf}}$. This is determined from a mass conservation approach which assumes the oxygen consumed by the reaction must be supplied by transport to the surface from the oxygen in the surrounding gas ($Y_{\text{O}_2, \infty}$). This produces the following balance between transport and kinetics [51]:

$$h_m \ln(B + 1) = Y_{\text{O}_2, \text{surf}} A_{\text{char}} \exp\left(-\frac{E_{\text{char}}}{RT}\right) \quad (17.9)$$

$$B = \frac{Y_{\text{O}_2, \infty} - Y_{\text{O}_2, \text{surf}}}{v_{\text{O}_2, \text{char}} + Y_{\text{O}_2, \text{surf}}} \quad (17.10)$$

The mass transfer coefficient is $h_m = h/c_p$, using the heat and mass transfer analogy (h is obtained from Eq. (8.2)). A linear approximation of $\ln(B + 1) \approx B$ is applied, allowing $Y_{\text{O}_2, \text{surf}}$ to be calculated from a quadratic equation and substituted into Eq. (17.8). It is important to note that the transport equation assumes a single reaction for char that produces CO_2 , as in Eq. (17.4). This approach also assumes the oxidation of char is a surface reaction where the details of temperature and oxygen gradients through the particle depth can be neglected and is therefore most applicable for thermally thin solids. These assumptions are not enforced by FDS in any way, however, so consideration should be given when using the `SURFACE_OXIDATION_MODEL`. Also note that because this model depends on full-scale parameters σ and h_m it is not necessarily appropriate to compare directly with micro-scale TGA data via `TGA_ANALYSIS`.

The equation governing the temperature of the thermally-thick solid is

$$\rho_s c_s \frac{\partial T_s}{\partial t} = \frac{1}{r^I} \frac{\partial}{\partial r} \left(r^I k_s \frac{\partial T_s}{\partial r} \right) + \dot{q}_s''' \quad (17.11)$$

where I is 1 for cylindrical and 2 for spherical coordinates. The source term is given by

$$\dot{q}_s''' = -(\Delta h_{\text{H}_2\text{O}} r_{\text{H}_2\text{O}} + \Delta h_{\text{pyr}} r_{\text{pyr}} + \Delta h_{\text{char}} r_{\text{char}}) \quad (17.12)$$

and the boundary condition

$$k \frac{\partial T_s}{\partial r} = \dot{q}_c'' + \dot{q}_r'' \quad (17.13)$$

The specific heat of the composite solid is given by

$$c_s = \frac{\sum \rho_{s, \alpha} c_{\alpha}}{\rho_s} \quad (17.14)$$

Suggested values [54] for the specific heats, in units of $\text{kJ}/(\text{kg} \cdot \text{K})$, as functions of temperature ($^{\circ}\text{C}$) are listed here:

$$c_{\text{dry}}, c_{\text{char}} = \begin{cases} 1.1 + 0.01 T & T < 200^{\circ}\text{C} \\ 2 & T \geq 200^{\circ}\text{C} \end{cases} \quad (17.15)$$

$$c_{\text{ash}} = 2 \quad (17.16)$$

Table 17.1: Default vegetation pyrolysis constants.

Parameter	Value	Reference
Dry Vegetation	$C_{3.4}H_{6.2}O_{2.5}A$	Ritchie et al. [48]
Fuel Gas	$C_{2.1}H_{6.2}O_{2.2}$	Sec. 17.1.2
Char	$C_{1.3}O_{0.3}A$	Sec. 17.1.2
A_{H_2O}	$600,000 \sqrt{K}/s$	Porterie et al. [44]
E_{H_2O}	$48,200 \text{ J/mol}$	Porterie et al. [44]
A_{pyr}	1040 s^{-1}	Grishin [52]
E_{pyr}	$61,041 \text{ J/mol}$	Grishin [52]
A_{char}	$465 \text{ kg/m}^2/s$	Boonmee and Quintiere [50]
E_{char}	$68,000 \text{ J/mol}$	Boonmee and Quintiere [50]
$v_{O_2, char}$	1.65	Kashiwagi and Nambu [53]
v_{char}	0.25	Assumption
v_{ash}	0.04	Assumption
Δh_c	$17,400 \text{ kJ/kg}$	Sec. 17.1.2
Δh_{pyr}	418 kJ/kg	Porterie et al. [44]
Δh_{char}	$-25,000 \text{ kJ/kg}$	Kashiwagi and Nambu [53]
Δh_{H_2O}	$2,259 \text{ kJ/kg}$	Porterie et al. [44]

$$c_{H_2O} = 4.19 \quad (17.17)$$

The rate of change of the total mass is

$$\frac{\partial \rho_s}{\partial t} = -r_{H_2O} - (1 - v_{char}) r_{pyr} - (1 - v_{ash}) r_{char} \quad (17.18)$$

The rate of change for the components of the vegetative mass during thermal degradation are:

$$\frac{\partial \rho_{s, H_2O}}{\partial t} = -r_{H_2O} \quad (17.19)$$

$$\frac{\partial \rho_{s, dry}}{\partial t} = -r_{pyr} \quad (17.20)$$

$$\frac{\partial \rho_{s, char}}{\partial t} = -v_{char} r_{pyr} - r_{char} \quad (17.21)$$

Gaseous products created during the thermal degradation process are water vapor (during evaporation), fuel vapor (during pyrolysis), and CO_2 (during char oxidation). The source terms ($\text{kg m}^{-3} \text{ s}^{-1}$) for these species are, respectively, r_{H_2O} , $(1 - v_{char}) r_{pyr}$, and $(1 + v_{O_2, char} - v_{ash}) r_{char}$. In addition, the char oxidation process consumes oxygen in the gas phase at a rate of $-v_{O_2, char} r_{char}$.

Figure 17.1 includes sample input lines demonstrating how the vegetation model can be applied to particles that represent pine needles that are 1 mm in diameter and 5 cm long. Note that the special surface oxidation model is invoked by setting `SURFACE_OXIDATION_MODEL` to `T` on the `MATL` line. All parameters described above must be explicitly specified in the input file—there are no hard-wired default values.

```

&MATL ID = 'MOISTURE'
  DENSITY = 1000.
  CONDUCTIVITY = 0.6
  SPECIFIC_HEAT = 4.190
  A = 600000.
  E = 48200.
  N_T = -0.5
  NU_SPEC = 1.
  SPEC_ID = 'WATER VAPOR'
  HEAT_OF_REACTION = 2259. /

&MATL ID = 'DRY VEGETATION'
  DENSITY = 400.
  CONDUCTIVITY = 0.1
  SPECIFIC_HEAT_RAMP = '...'
  A = 1040.
  E = 61,041.
  NU_MATL = 0.25
  MATL_ID = 'CHAR'
  NU_SPEC = 0.74
  SPEC_ID = 'Fuel Gas'
  HEAT_OF_REACTION = 418. /

&MATL ID = 'CHAR'
  DENSITY = 300.
  CONDUCTIVITY = 0.05
  SPECIFIC_HEAT_RAMP = '...'
  SURFACE_OXIDATION_MODEL = T
  A = 465.
  E = 68000.
  MATL_ID = 'ASH'
  NU_MATL = 0.10
  SPEC_ID = 'PRODUCTS', 'AIR'
  NU_SPEC = 8.07, -7.17
  HEAT_OF_REACTION = -25000. /

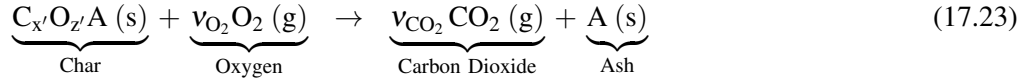
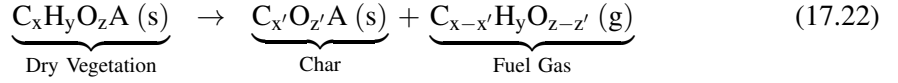
&MATL ID = 'ASH'
  DENSITY = 67.
  CONDUCTIVITY = 0.1
  SPECIFIC_HEAT_RAMP = '...' /

```

Figure 17.1: Input parameters describing vegetation.

17.1.2 Gas Phase

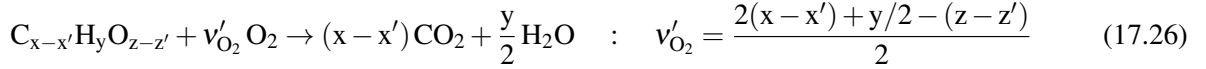
To estimate the heat of combustion of the gas phase reaction of fuel vapors generated by the pyrolysis of vegetation, the decomposition reaction given in Eqs. (17.2) and (17.3) is written as an equivalent set of reactions assuming that the Dry Vegetation is given by the effective formula $C_xH_yO_zA$, where A represents the inorganic components that eventually form the Ash.



$$v_{O_2} = \frac{v_{O_2, \text{char}} v_{\text{char}} W_{\text{veg}}}{W_{O_2}} ; \quad v_{CO_2} = \frac{(1 + v_{O_2, \text{char}} - v_{\text{ash}}) v_{\text{char}} W_{\text{veg}}}{W_{CO_2}} ; \quad W_{\text{veg}} = \frac{12x + y + 16z}{1 - v_{\text{ash}} v_{\text{char}}} \quad (17.24)$$

$$x' = v_{CO_2} ; \quad z' = 2(v_{CO_2} - v_{O_2}) \quad (17.25)$$

The ideal one step gas phase reaction of the Fuel Gas is as follows:



The heat of combustion of the gas phase reaction (heat release per unit mass fuel gas consumed) can be estimated from the heat release per unit mass of oxygen consumed, $E = 13.98$ MJ/kg, measured by Tihay et al. [55]:

$$\Delta h_c \approx \frac{W_{O_2} v'_{O_2}}{12(x - x') + y + 16(z - z')} E \quad (17.27)$$

The total heat release rate of burning vegetation is the sum of both the gas phase combustion of pyrolyzed plant matter and the solid phase exothermic char oxidation reaction. The effective heat of combustion, $\Delta h_{c, \text{eff}}$, is a weighted average of the two reactions. The heat of combustion of the pyrolyzed plant matter, Δh_c , is approximately 17400 kJ/kg according to Eq. (17.27). The heat of reaction for the char oxidation, Δh_{char} is approximately -25000 kJ/kg (the minus sign in this instance refers to an *exothermic* solid phase reaction). The effective heat of combustion is found from:

$$(1 - v_{\text{char}} v_{\text{ash}}) \Delta h_{c, \text{eff}} = (1 - v_{\text{char}}) \Delta h_c + v_{\text{char}} (1 - v_{\text{ash}}) (-\Delta h_{\text{char}}) \quad (17.28)$$

17.1.3 Examples

Consider the sample case called `WUI/char_oxidation_1.fds` where $m = 0.0531$ kg of dry vegetation is completely consumed by fire except for a small amount of ash. The char and ash yields are $v_{\text{char}} = 0.25$ and $v_{\text{ash}} = 0.04$. The heat of combustion of the pyrolyzed fuel vapor is $\Delta h_c = 17400$ kJ/kg; thus the effective heat of combustion for the mass of vegetation consumed is $\Delta h_{c, \text{eff}} = 19146$ kJ/kg according to Eq. (17.28). The area under the heat release rate curve in the left hand plot of Fig. 17.2 should be $m(1 - v_{\text{char}} v_{\text{ash}}) \Delta h_{c, \text{eff}} \approx 1012$ kJ and the mass of ash remaining should be $v_{\text{char}} v_{\text{ash}} m \approx 0.000531$ kg, as shown in the right hand plot of Fig. 17.2.

The sample case called `Verification/WUI/char_oxidation_2.fds` considers two pines needle both with an initial temperature $T_{s,i} = 293.15$ K introduced into an adiabatic box with a volume $V = 0.125$ m³ and initial temperature $T_{g,i} = 1073.15$ K. The needles are a cylinder with length $L = 0.1$ m and radius $r = 0.0005$ m. They are composed of plant matter, or 'pulp', with a density of $\rho_s = 500$ kg/m³, specific heat $c_s = 1$ kJ/(kg·K), and mass $m_p = \pi r^2 L \rho_s$ with one needle dry and the other needle having 20 % moisture.

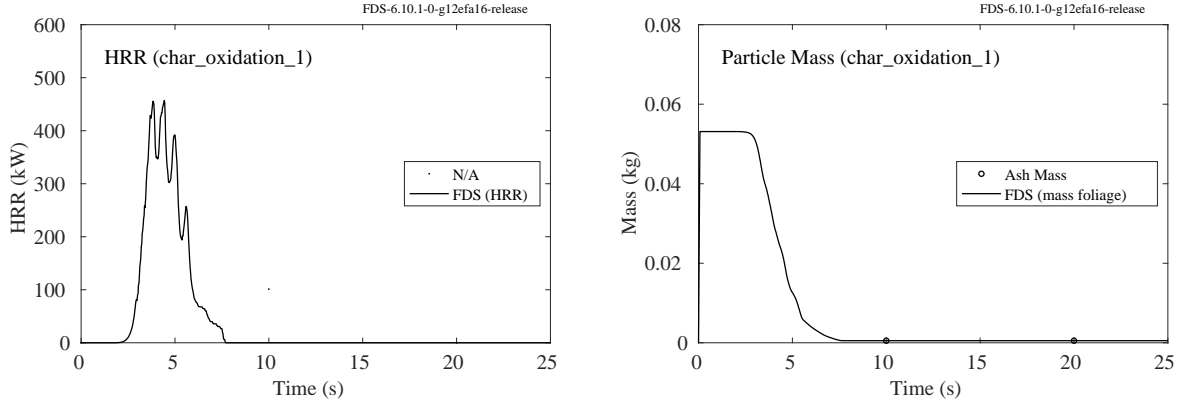


Figure 17.2: Heat release rate (left) and total mass (right) as functions of time in the `char_oxidation_1` test case.

The box is initially filled with nitrogen and oxygen, the latter with an initial mass fraction of 0.05. The vegetation undergoes three reactions, one in which $(1 - v_{\text{char}})m_p$ is converted to fuel gas and $v_{\text{char}}m_p$ to solid char, and the second in which the char oxidizes to form CO_2 gas and solid ash, and the third is the evaporation of moisture. The char yield $v_{\text{char}} = 0.25$ and the ash yield $v_{\text{ash}} = 0.02$. Note that the ash yield is the fraction of char that is converted to ash, not the fraction of the original pine needle.

To simplify calculation of the exact answer, all species and material c values, except for the moisture, are set to 1 kJ/(kg·K). Energy is conserved if the enthalpy change of the particle is equal and opposite to the internal energy change of the gas. Note that it is internal energy and not enthalpy since the box is sealed and any pressure rise must be accounted for in the energy balance. For this case, since the gas species have a specified specific heat, their heat of formation are all 298.15 kJ/kg. Since there are four materials and three material reactions, following the discussion in Sec. 9.4, FDS will perform a least squares solution to determine the material heats of formation (the enthalpy at 298.15 K). The result of this calculation is a pulp heat of formation of 6145 kJ/kg, a char heat of formation of 25,290 kJ/kg, an ash heat of formation of -231 kJ/kg, and a moisture heat of formation of -2202 kJ/kg.

At the start of the simulation the particle enthalpy is:

$$H_{\text{part}} = 0.1 \text{ m} \times \pi \times (0.0005 \text{ m})^2 \times 500 \text{ kg/m}^3 \times (6145 - 5) \text{ kJ/kg} = 0.2411 \text{ kJ} \quad (17.29)$$

At the end of the simulation only ash is left which is 2 % of the char mass which was 25 % of the initial particle mass. The ash will end up at the gas temperature.

$$H_{\text{part}} = 0.1 \text{ m} \times \pi \times (0.0005 \text{ m})^2 \times 500 \text{ kg/m}^3 \times 0.02 \times 0.25 \times (775 - 231) \text{ kJ/kg} = 0.0001 \text{ kJ} \quad (17.30)$$

The particle will therefore lose 0.2410 kJ.

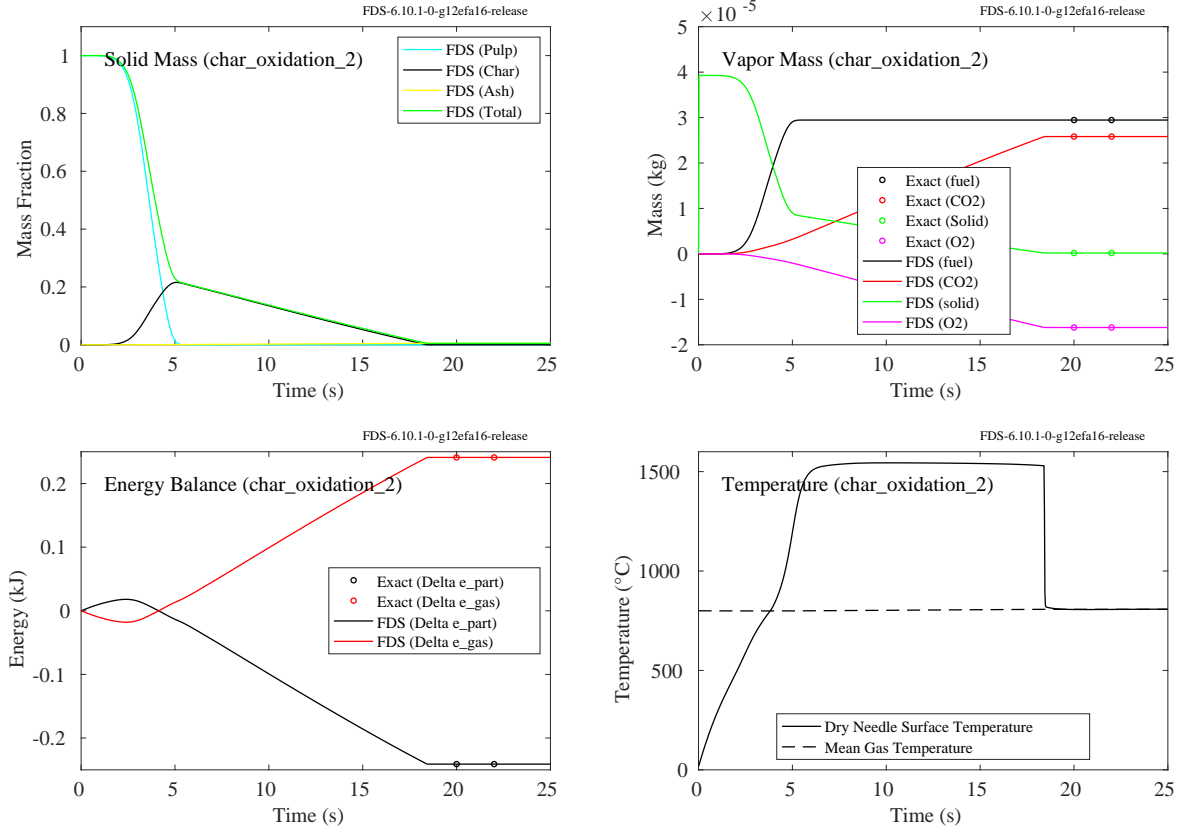


Figure 17.3: (Top left) Mass of virgin foliage, char and ash. (Top right) Total mass of solid and fuel vapor, CO₂, and O₂. (Lower left) Energy lost by the pine needle (black) and gained by the gas (red). (Lower right) Surface temperature of the pine needle and mean gas temperature of the enclosure.

17.2 Lagrangian Particle Model

Lagrangian particles can be used to represent different types of vegetation, like leaves, grass, and so on. The best way to explain how to use this feature is by way of example. The sample input file in the `WUI` (Wildland-Urban Interface) folder called `pine_needles.fds`¹ describes a collection of pine needles that occupy a unit cube. The input parameters for this case are spread among a number of namelist groups.

First, the `PART` line defines a class of solid particles that represent pine needles:

```
&PART ID='pine needles', SAMPLING_FACTOR=1, SURF_ID='wet vegetation',
      DRAG_COEFFICIENT=2.8, PROP_ID='needle image', STATIC=T /
```

`STATIC=T` specifies that the needles do not move, and `SAMPLING_FACTOR=1` specifies that each needle is to be shown in Smokeview. More importantly, this line specifies a drag coefficient for the needles based on wind tunnel measurements made at NIST [56]. The force per unit volume exerted by the vegetation is given by:

$$\mathbf{f}_b = \frac{\rho}{2} C_d C_s \beta \sigma \mathbf{u} |\mathbf{u}| \quad (17.31)$$

where ρ is the air density, C_d is the drag coefficient, C_s is the `SHAPE_FACTOR` (0.25 by default), β is the

¹The parameter values in this example have been chosen simply to demonstrate the technique. These values should not be used for a real calculation.

packing ratio (mass per unit volume divided by the material density), σ is the surface area to volume ratio (see below), and u is the air velocity.

The `PART` line also tells Smokeview to visualize each needle as a tube that is 0.1 m long and 0.5 mm in diameter:

```
&PROP ID='needle image', SMOKEVIEW_ID='TUBE',
      SMOKEVIEW_PARAMETERS='L=0.1','D=0.0005' /
```

The geometry and composition of the needle is described on the `SURF` line that is referenced by the `PART` line:

```
&SURF ID = 'wet vegetation'
      MATL_ID = 'dry pine'
      MOISTURE_FRACTION = 0.25
      SURFACE_VOLUME_RATIO = 8000.
      LENGTH = 0.1
      GEOMETRY = 'CYLINDRICAL' /
```

The needle is composed of two materials—'dry pine' and 'MOISTURE'. Following the convention used in forestry, the moisture content is expressed via the `MOISTURE_FRACTION`, which is the mass of moisture divided by the mass of *dry* vegetation. Do not confuse this with the mass fraction of moisture, Y_m , which is related to the moisture fraction, M , via

$$Y_m = \frac{M}{1+M} \quad (17.32)$$

The length of each needle is specified directly, but its diameter is specified via the surface area to volume ratio, $\sigma = 2/r$ for a cylinder. For a sphere, $\sigma = 3/r$, where r is the radius of the cylinder or sphere. The `GEOMETRY` can be 'CARTESIAN', i.e. a plate, 'CYLINDRICAL', or 'SPHERICAL'. For a plate, $\sigma = 1/\delta$ where δ is the half-thickness of the plate. The `MATL_ID` specifies the thermo-physical properties of the pine needle.

```
&MATL ID = 'dry pine'
      DENSITY = 500.
      CONDUCTIVITY = 0.1
      SPECIFIC_HEAT = 1.0
      REFERENCE_TEMPERATURE = 300.
      NU_MATL = 0.2
      NU_SPEC = 0.8
      SPEC_ID = 'CELLULOSE'
      HEAT_OF_REACTION = 1000
      MATL_ID = 'CHAR' /

&MATL ID = 'CHAR'
      DENSITY = 200.
      CONDUCTIVITY = 1.0
      SPECIFIC_HEAT = 1.6 /
```

Note that if you specify a `MOISTURE_FRACTION` on the `SURF` line, FDS will automatically add a `MATL` line for 'MOISTURE' as it is written in Fig. 17.1. FDS will also alter the `DENSITY` of the dry vegetation, in this case 'dry pine', so that the size and wood content of the particle do not change when moisture is added. The modified density of the “dry” vegetation, $\tilde{\rho}_d$, is given by:

$$\tilde{\rho}_d = \frac{\rho_d}{1 - \frac{\rho_d}{\rho_m} M} \quad (17.33)$$

where ρ_d is the user-specified density of the dry vegetation and ρ_m is the density of the moisture, typically

assumed to be 1000 kg/m^3 . The rationale behind this adjustment is that dry vegetation contains small pores that trap moisture. Thus, when moisture is added, the size and vegetation content of the particle do not change.

Each `MATL` line must include the `DENSITY`, `CONDUCTIVITY` and `SPECIFIC_HEAT` of that component. If the material component undergoes a reaction, there needs to be either a `REACTION_TEMPERATURE` or more detailed kinetic parameters like the ones listed in Fig. 17.1. Material components that react need stoichiometric coefficients `NU_SPEC` and `NU_MATL` to designate how much of the material mass is converted to a gas `SPECIES` and how much to a solid `MATERIAL`. In this case, 'CELLULOSE' is the gas-phase fuel specified on the `REAC` line. The gas species 'WATER VAPOR' is defined by a `SPEC` line. Even though water vapor is included in the combustion products, you still need to explicitly specify 'WATER VAPOR' as the gasified form of the moisture in the pine needles.

Finally, the pine needles are introduced into the simulation via the `INIT` line:

```
&INIT PART_ID='pine needles', XB=0.,1.,0.,1.,0.,1., N_PARTICLES=1000,
      MASS_PER_VOLUME=0.8, DRY=T /
```

This line inserts 1000 Lagrangian particles representing pine needles randomly within a unit cube. The `MASS_PER_VOLUME` is the mass (kg) of solid needles divided by the volume (m^3) they occupy, sometimes called the “bulk density.” The parameter `DRY=T` means that if you have specified a `MOISTURE_FRACTION` on the `SURF` line that describes the vegetation, then the actual mass per volume of wet vegetation is

$$m_w''' = m_d''' (1 + M) \quad (17.34)$$

In the example, 1 kg of wet pine needles occupy 1 m^3 . The number of particles used to represent the pine needles is somewhat arbitrary. FDS will automatically weight the specified number so that the total mass per volume is 1 kg. The vegetation is heated until all of the water and fuel evaporate. The fuel is not allowed to burn by setting the ambient oxygen concentration to 1 %. Figure 17.4 shows the evolution of the fuel, water and char mass. Agreement with the expected values means that mass is conserved.

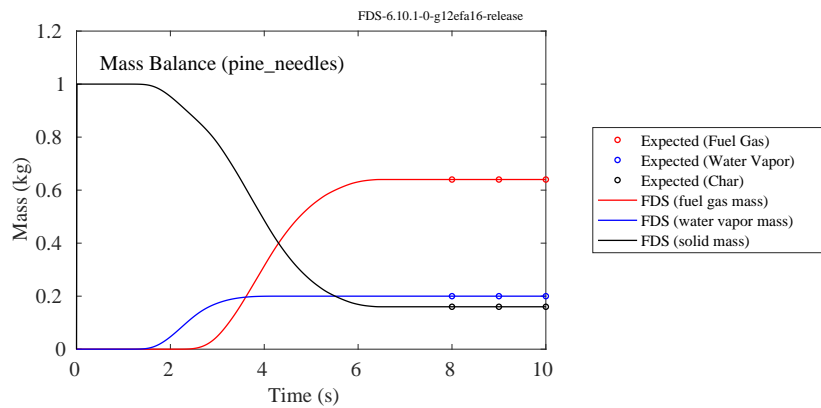


Figure 17.4: Evolution of vegetation mass in the `pine_needles` test case.

17.2.1 Trees

A coniferous tree (e.g. pine, spruce, fir) can be represented in FDS as a collection of Lagrangian particles as follows:


```
&PART ID='needles', DRAG_COEFFICIENT=2.8, SURF_ID='needle surface', STATIC=T,
      COLOR='FOREST GREEN' /
&INIT ID='needles', PART_ID='needles', XYZ=..., CROWN_BASE_WIDTH=3.2,
      CROWN_BASE_HEIGHT=0.2, TREE_HEIGHT=2, SHAPE='CONE', N_PARTICLES_PER_CELL=1,
      MASS_PER_VOLUME=10., DRY=T /
```

These lines create a collection of particles in the shape of a cone. The dimensions of the cone are specified via the parameters `CROWN_BASE_WIDTH`, the diameter of the base of the cone (m), `CROWN_BASE_HEIGHT`, the distance (m) from the ground to the base of the cone, `TREE_HEIGHT`, the height (m) of the tree from ground to tip, and the position triplet, `XYZ`, of the trunk at the ground. Note that in this example, exactly one particle is specified per grid cell, positioned randomly within the cell. The number of actual pine needles this single particle represents depends on the specified (dry) `MASS_PER_VOLUME`. Additional details on specifying different standard tree crown geometries can be found in Section 15.5.3.

A handy way to introduce many trees into a simulation is by way of a `SURF` line as follows:

```
&SURF ID='forest', ..., INIT_IDS(1:2)='needles','branches', INIT_PER_AREA=0.1 /
```

This line indicates that solid boundaries with the surface properties given by 'forest' shall have 1 tree per 10 m², as indicated by the parameter `INIT_PER_AREA`. The trees may consist of more than one type of particle, as indicated by the array `INIT_IDS`. Note that this construct is designed for trees and is only invoked on upward-facing horizontal boundaries. Also note that the coordinates `XYZ` on the `INIT` line should be 0, 0, 0 so that it can be repositioned accordingly.

17.2.2 Bulk Density Input Files

In some cases, the user may have access to detailed 3-dimensional spatial information on the distribution of vegetation bulk density, or `MASS_PER_VOLUME`, within a tree crown or a forest canopy. These data may be available in a voxelized (gridded) format and are increasingly common with the advancement of both remote sensing techniques (e.g. lidar) and tools for modeling forest canopy structure. It is possible to directly import such bulk density data by specifying a `BULK_DENSITY_FILE` on the `INIT` line. Each file should specify the bulk density values for the type of vegetation defined by the `PART_ID` and `SURF_ID` associated with the particular `INIT` line. For example, one file may specify the spatial distribution of foliage, another for branches, and so on:

```
&INIT ID='insert', PART_ID='foliage part', BULK_DENSITY_FILE='canopy_foliage.bdf' /
```

For a full example, see the `bulk_density_file.fds` test cases in the `WUI` directory of the verification suite. Bulk density data are assumed to be provided on a dry-mass basis. `DRY=T` is set automatically and moisture mass is added following the specification of the associated `SURF` (see Sec. 17.2 for detail).

The `BULK_DENSITY_FILE` must be a FORTRAN unformatted (binary) file and follows the following convention:

```
WRITE(LU_VEG_IN) VXMIN, VXMAX, VYMIN, VYMAX, VZMIN, VZMAX
WRITE(LU_VEG_IN) VDX, VDY, VDZ
WRITE(LU_VEG_IN) NVOX
WRITE(LU_VEG_IN) VCX, VCY, VCZ
WRITE(LU_VEG_IN) MASS_PER_VOLUME
.
.
WRITE(LU_VEG_IN) VCX, VCY, VCZ
WRITE(LU_VEG_IN) MASS_PER_VOLUME
```

The sextuplet `VXMIN, VXMAX, VYMIN, VYMAX, VZMIN, VZMAX` gives the bounds (upper and lower voxel faces) of the entire vegetation containing volume. This is used to determine whether a specific mesh contains any of the vegetation. The triplet `VDX, VDY, VDZ` gives the spatial resolution of the input data and `NVOX` is the total number of voxels which actually contain vegetation. The coordinates of each voxel center are given by `VCX, VCY, VCZ` and the bulk density by `MASS_PER_VOLUME`. Third-party tools to create vegetation inputs are in development. However, an example python script for creating a custom `BULK_DENSITY_FILE` can be found in the cad GitHub repository, `firemodels/cad`.

FDS effectively creates a `BLOCK` shaped `INIT` line for each voxel. In this way, it is possible to change the FDS resolution and meshing without needing to create a new `BULK_DENSITY_FILE`. By default, if a `BULK_DENSITY_FILE` is specified FDS will input vegetation with `N_PARTICLES_PER_CELL = 1` to minimize memory usage. However, if an FDS grid cell intersects multiple vegetation voxels (for example, due to misalignment between FDS and voxel resolution), more than one particle will be added to the cell. The input mass will be conserved; this will simply increase memory usage, and it is generally recommended to provide a `BULK_DENSITY_FILE` with the same resolution as your simulation if possible.

Additionally, the `BULK_DENSITY_FILE` can be scaled using a `BULK_DENSITY_FACTOR`. The value of this input, also provided on the `INIT` line, will be multiplied by all values in the `BULK_DENSITY_FILE` when they are read in. This allows quick adjustment of the overall canopy density without having to regenerate new binary files. It can also be used to re-apply the same `BULK_DENSITY_FILE` for different classes of particles, such as foliage and branches, which may have the same spatial distribution of material but different proportions of bulk density.

17.2.3 Firebrands

Firebrands, or embers, are small pieces of burning wood and vegetation that can be lofted into the air and blown by the wind ahead of a wildland fire front. Manzello et al. [57] have developed a variety of experimental apparatus designed to generate firebrands in a laboratory setting. The example input file called `dragon_5a` in the `WUI` (Wildland-Urban Interface) folder is a very simple mock-up of one of these experiments. The word “dragon” is based on the nickname of the apparatus; 5a is the figure number in Ref. [57] on which this example case is loosely based. In the experiment, 700 g of small dowels (length 50 mm, diameter 8 mm) made of Ponderosa Pine wood were poured into a small steel chamber equipped with several propane burners. The dowels were left to burn for roughly a minute subject to a slow induced air flow after which time the air flow was increased and firebrands were propelled horizontally out of a 15 cm duct 2.25 m above the lab floor. It is reported that after several replicate experiments, the average mass of the firebrands collected from pans on the floor was 57 g. The average diameter of the collected dowels was 5.6 mm, and the average length was 13.5 mm.

It is not possible to simulate the experiment in FDS exactly as it was performed. The reason is that in the experiment, all 700 g of the wooden dowels were poured into the heating chamber at once. FDS cannot handle such a dense packing of Lagrangian particles. Instead, the simulated dowels are introduced at a rate of 10 per second. FDS also does not have a mechanism to break-up the dowels, reducing their length from 50 mm to 13.5 mm. Thus, the initial cylindrical particles are 13.5 mm and remain that length throughout the simulation. The diameter of the cylindrical particles is reduced, however, from 8 mm to 5.6 mm, which takes the initial density of 440 kg/m^3 down to 71 kg/m^3 because the mass of the firebrands is assumed to be 8 % of the original. The plot in Fig. 17.5 shows the increasing mass of firebrands thrown to the floor in the simulation after 100 s of particle insertion. The total mass of particles inserted into the apparatus is:

$$\pi (0.004 \text{ m})^2 \times (0.0135 \text{ m}) \times (440 \text{ kg/m}^3) \times (10 \text{ part/s}) \times (100 \text{ s}) \approx 0.3 \text{ kg} \quad (17.35)$$

The amount expected on the floor is approximately 0.024 kg.

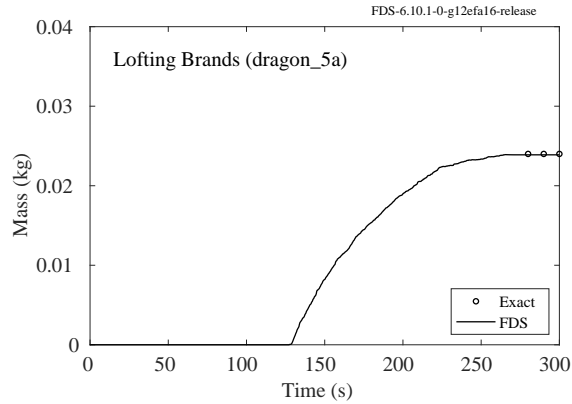


Figure 17.5: Mass generation of firebrands in the `dragon_5a` test case.

17.2.4 Ember Generation From Particles

Although in its early stages of development, a rudimentary ember generation algorithm is available for use in FDS. Lagrangian particle classes may be tagged as “ember particles” and thus subject to lofting by a drag force. Two parameters control the point at which a given particle becomes an ember. First is the density threshold, controlled by the parameter `EMBER_DENSITY_THRESHOLD` on the `PART` line. As a vegetative particle burns and converts to char its density decreases from that of a woody material with a density of 500-1000 kg/m³ to char with a density of 50-100 kg/m³. As the wood turns to char its structural integrity diminishes and the drag forces may rip the vegetative element apart. While the phenomenon of course depends on the force exerted by the gas flow around the particle, we use a velocity threshold as a surrogate to the drag force, since this is more intuitive. The threshold is controlled by `EMBER_VELOCITY_THRESHOLD` on `PART`. The `EMBER_PARTICLE` tag should only be applied to particle classes that are initially `STATIC`. A typical example is given below.

```
&PART ID='small branches', STATIC=T, EMBER_PARTICLE=T, EMBER_DENSITY_THRESHOLD=75.,
      EMBER_VELOCITY_THRESHOLD=3.0
```

Ember Removal In some cases, tracking embers may be prohibitively expensive or irrelevant to the problem and it may be sufficient simply to remove the particles from the calculation once they reach the ember thresholds. If this is desired, add `TRACK_EMBERS=F` to the `PART` line.

17.3 Boundary Fuel Model

In many simulations of wildland fire, the ground vegetation layer is too thin to be resolved explicitly, as is done when using Lagrangian particles to represent the vegetation. In such cases, the ground vegetation can be modeled as a porous boundary consisting of a layer of dry vegetation, moisture, and air, underneath of which is hard ground [58]. The drag exerted by the vegetation is modeled using a special velocity boundary condition, and convective heat transfer is modeled via a source term in the one-dimensional heat conduction equation that is solved through the layer of vegetation and solid ground. Thermal radiation penetrates the vegetation layer via a 1-D radiative transport equation that is used for semi-transparent solids.

To invoke this *Boundary Fuel Model*, create a `SURF` line similar to the following:

```
&SURF ID = 'Ground Vegetation'
      MATL_ID(1,1) = 'Dry Vegetation'
      MATL_ID(2,1) = 'Soil'
      MOISTURE_FRACTION(1) = 0.218
      SURFACE_VOLUME_RATIO(1) = 3092.
      MASS_PER_VOLUME(1) = 5.
      THICKNESS(1:2) = 0.076,0.1 /
```

The presence of the parameter `MASS_PER_VOLUME` automatically triggers the Boundary Fuel Model. Note that its argument of 1 refers to the first layer; the second layer being `Soil`. If you specify `MOISTURE_FRACTION`, FDS will automatically add a `MATL` line for 'MOISTURE'.

The drag exerted on the wind flowing through the vegetation is imposed as a force term in the gas phase grid cell adjacent to the boundary:

$$\mathbf{f}_b = \frac{\rho}{2} C_s \sigma \beta C_d \frac{h_b}{\delta z} \mathbf{u} |\mathbf{u}| \quad (17.36)$$

where ρ is the density of the gas, C_s is the `SHAPE_FACTOR` of the subgrid-scale vegetation (0.25 by default), σ is the surface area to volume ratio (`SURFACE_VOLUME_RATIO`), β is the packing ratio (mass per volume divided by the solid density), C_d is the `DRAG_COEFFICIENT` (2.8 by default), h_b is the depth of the vegetation (`THICKNESS(1)`), δz is the height of the grid cell, and \mathbf{u} is the gas velocity in the first grid cell.

Thermal radiation is absorbed in depth according to a 1-D radiative transport solver. The absorption coefficient is given by:

$$\kappa = C_s \sigma \beta \quad (17.37)$$

Thermal convection is not imposed at the interface between the gas phase and the vegetation layer, but rather is imposed via a source term in the 1-D heat conduction solver:

$$\langle \dot{q}_{c,b}''' \rangle = \sigma \beta \dot{q}_c'' \quad (17.38)$$

where \dot{q}_c'' is given in Eq. (8.1) under the assumption that the subgrid-scale vegetation is cylindrical and the gas velocity and temperature are taken from the first gas phase grid cell adjacent to the boundary.

17.3.1 Burnout Time

For scenarios where the fire's rate of spread (ROS) is relatively slow and the numerical grid is relatively coarse, the spatial resolution may be inadequate. Consider the simple relationship between the ROS, the width of the fire front, W , and the average burn duration of a particular patch of vegetation, Δt_b :

$$W = \text{ROS} \times \Delta t_b \quad (17.39)$$

The more grid cells spanning the front width, the better resolved the simulation. In badly resolved simulations, the time required for the fire to spread the width of a grid cell may be comparable or longer than the

burnout time, Δt_b . In such cases, it is necessary to specify an effective “burnout time” that accounts for the unresolved burning occurring at subgrid scales. This parameter, `MINIMUM_BURNOUT_TIME` (s), is specified on the `SURF` line that provides parameters for the vegetation. By default, it has a very large value, meaning that it is ignored unless specified.

17.4 Comparing the Particle and Boundary Fuel Models of Vegetation

For the purpose of comparing the two methods of representing ground vegetation, either as particles or a porous boundary, consider the simple test cases in the sample folder named `WUI` (Wildland-Urban Interface) under the heading `ground_vegetation`.

17.4.1 Combustible Load

In the first case, `WUI/ground_vegetation_load.fds`, two 3 m long channels contain $A = 1 \text{ m}^2$ of generic ground vegetation with a dry mass per unit volume, $m_d''' = 4 \text{ kg/m}^3$, dry density, $\rho_d = 400 \text{ kg/m}^3$, height, $h_b = 0.075 \text{ m}$, char yield, $y_{\text{char}} = 0.25$, ash yield, $y_{\text{ash}} = 0.02$, moisture fraction, $M = 0.1$. The ceiling of the channel is set to 700°C and the lateral walls are periodic, mimicking an infinitely wide fire front. No wind is imposed in this example. The total amount of mass evaporated is

$$(1 + M) m_d''' h_b (1 - y_{\text{char}} y_{\text{ash}}) A = 0.32835 \text{ kg} \quad (17.40)$$

Figure 17.6 displays the heat release rate (left) and cumulative mass consumption (right) of the two calculations.

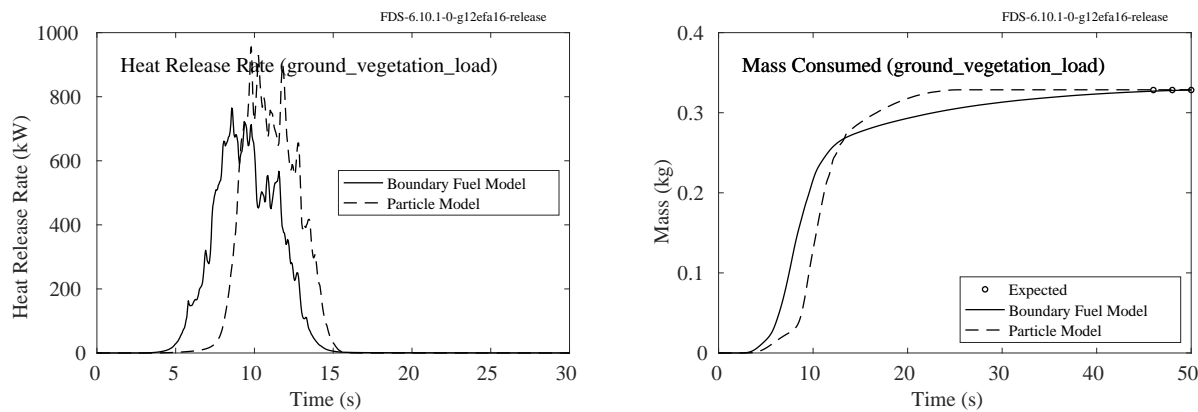


Figure 17.6: (Left) Heat release rate of the two ground vegetation simulations. (Right) Cumulative mass consumption of the two cases.

17.4.2 Vegetation Drag

In this next example, ground vegetation with the same properties as the previous example lines the floor of two wind tunnels that are 8 m long, 1 m wide, and 0.5 m tall. The height of the vegetation is 2.5 cm, and the grid size in the simulation is 5 cm. The tunnel walls and ceiling are free-slip. Particles model the vegetation in one tunnel; the Boundary Fuel Model is applied in the other. A 2 m/s wind is imposed for 30 s, followed by 4 m/s for another 30 s, and finally 6 m/s for another 30 s. The pressure upstream of the vegetation is recorded. The tunnel is open downstream, where the pressure is ambient. The upstream pressure ought to be similar, as shown in Fig. 17.7.

17.4.3 Vegetation Radiation Absorption

In this next example, ground vegetation lines the floors of two unit cubes with hot ceilings and cold floors. The walls are `MIRROR` boundaries so that the ceiling and floor can be taken as infinitely wide parallel plates.

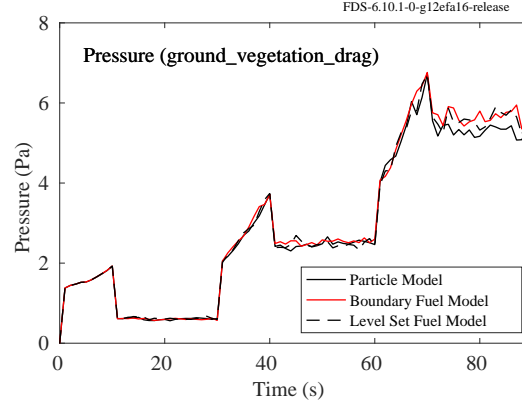


Figure 17.7: Pressure upwind of a 6 m wide strip of ground vegetation.

The height of the vegetation is $\delta = 0.05$ m, and the grid size in the simulation is 5 cm. Particles model the vegetation in one cube; the Boundary Fuel Model is applied in the other. The surface area to volume ratio of the vegetation, $\sigma_v = 3092 \text{ m}^{-1}$, the packing ratio, $\beta = 0.02$, and the shape factor, $C_s=0.25$. The ceiling temperature, $T_c = 993$ K; the floor temperature, $T_f = 293$ K. The emissivity of both surfaces is 1. The radiation absorption coefficient is $\kappa \equiv C_s \sigma_v \beta = 15.46 \text{ m}^{-1}$. Using the third-order exponential integral function, E_3 , the expected heat flux to the floor is [59]:

$$\dot{q}'' = 2 \sigma T_c^4 E_3(\kappa \delta) - \sigma T_f^4 (1 - 2 E_3(\kappa \delta)) = 15.52 \text{ kW/m}^2 \quad (17.41)$$

The results are shown in Fig. 17.8. The Lagrangian particles that represent the vegetation only occupy two layers of grid cells; thus, the radiation absorption calculation has some error. In the case of the Boundary Fuel Model the error originates from the use of the "two-flux" model for internal radiation in the fuel (see the FDS Technical Reference Guide [3]).

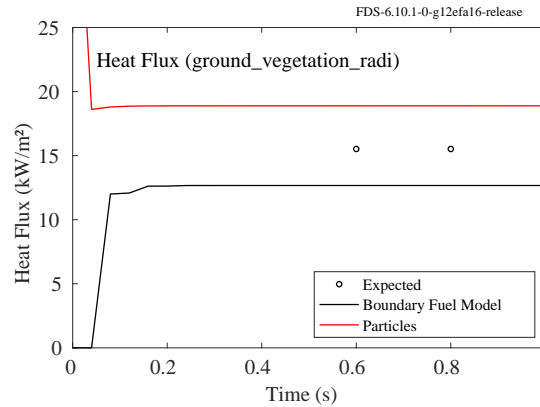


Figure 17.8: Heat flux penetrating a patch of ground vegetation.

17.4.4 Vegetation Convective Heating

In this next example, two small wind tunnels blowing hot air at $U = 2$ m/s and $T_{\text{air}} = 300$ °C contain small amounts of vegetation modeled using either Lagrangian particles or the Boundary Fuel Model. The tunnel walls are `ADIABATIC` so that heat is not lost to the walls. Two cylindrical particles of different diameters

($\sigma_{v,A} = 3000 \text{ m}^{-1}$ and $\sigma_{v,B} = 1500 \text{ m}^{-1}$) occupy adjacent grid cells that abut the solid floor. Each particle represents all the vegetation within the grid cell. In the other tunnel, two adjacent patches on the floor represent the equivalent amount of vegetation via the Boundary Fuel Model. The initial vegetation temperature is $T_0 = 20^\circ\text{C}$. Radiation heat transfer is turned off. The expected temperature increase with time is found by solving:

$$\rho_s c_s \frac{dT}{dt} = \sigma_v h (T_{\text{air}} - T) \quad ; \quad h = \frac{k_{\text{air}}}{D} 0.683 \text{Re}^{0.466} \text{Pr}^{0.333} \quad ; \quad \text{Re} = \frac{\rho_{\text{air}} U D}{\mu_{\text{air}}} \quad ; \quad \text{Pr} = 0.7 \quad (17.42)$$

where the solid density $\rho_s = 400 \text{ kg/m}^3$, specific heat $c_s = 1.5 \text{ kJ/(kg}\cdot\text{K)}$, $\rho_{\text{air}} \approx 0.60 \text{ kg/m}^3$, $\mu_{\text{air}} \approx 3 \times 10^{-5} \text{ kg/(m}\cdot\text{s)}$, $k_{\text{air}} \approx 4 \times 10^{-5} \text{ kW/(m}\cdot\text{K)}$, $D = 4/\sigma_v$. The solution is:

$$T(t) = T_{\text{air}} - (T_{\text{air}} - T_0) \exp\left(-\frac{h \sigma_v t}{\rho_s c_s}\right)^\circ\text{C} \quad (17.43)$$

The resulting vegetation temperatures are shown in Fig. 17.9.

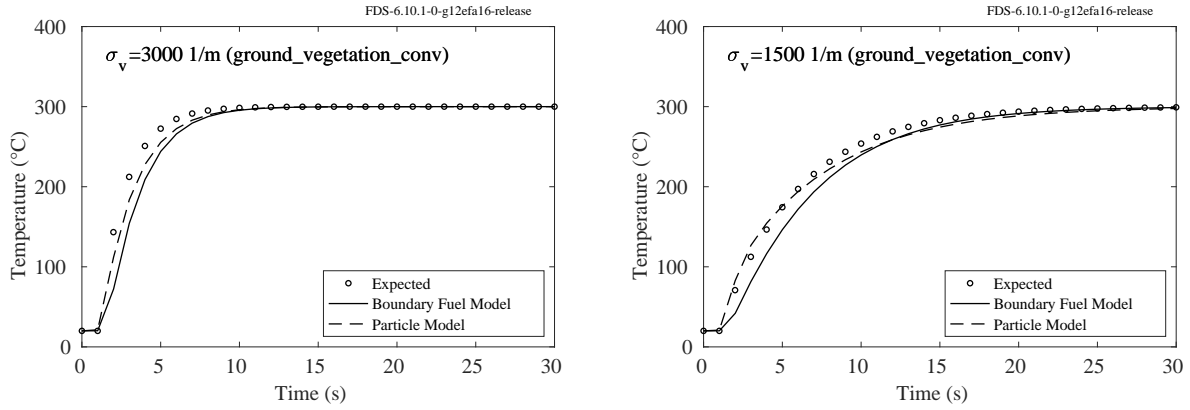


Figure 17.9: Temperature of convectively heated vegetation of two different sizes. (Left) $\sigma_v = 3000 \text{ m}^{-1}$. (Right) $\sigma_v = 1500 \text{ m}^{-1}$.

17.5 Level Set Model for Wildland Fire Spread

For simulations of wildland fires spanning large areas that cannot be gridded finely enough to predict fire spread using a physics-based model, there is an empirical model built into FDS based on level sets [60]. The methodology reproduces in an Eulerian framework the approach used in the Lagrangian-based fire front-tracking model FARSITE [61]. This approach makes use of the Rothermel-Albini [62, 63] surface fire spread rate formulae and the assumption that a surface fire spreading from a point under certain wind, slope and vegetation conditions does so with an ellipse-shaped² fire front with, for a given wind speed, a fixed length-to-breadth ratio [64].

To invoke the level set feature, you *must* set `LEVEL_SET_MODE` on the `MISC` line to be either 1, 2, 3 or 4:

- `LEVEL_SET_MODE=1` Only the level set simulation is performed. The wind is not affected by the terrain, and there is no fire.
- `LEVEL_SET_MODE=2` The wind field is established over the terrain, but it is “frozen” when the fire ignites.
- `LEVEL_SET_MODE=3` The wind field follows the terrain, but there is no actual fire in the simulation. It is just front-tracking.
- `LEVEL_SET_MODE=4` The wind and fire are fully-coupled. When the fire front arrives at a given surface cell, it burns for a finite duration and with a heat release per unit area provided as part of the fuel model.

For Modes 2 through 4, the wind speed and direction that are used in the level set equation are taken from the first grid cell above ground level.

Regardless of the mode, the front propagation is initiated by specifying a small area to be ignited at a particular time:

```
&SURF ID='Ignited Area', VEG_LSET_IGNITE_TIME=0., COLOR='RED' /  
&VENT XB=..., SURF_ID='Ignited Area' /
```

These lines indicate that a patch of terrain specified using `XB` on the `VENT` line ignites at the start of the simulation. If the terrain is constructed using the immersed boundary framework (i.e. a `GEOM` line), add the logical parameter `GEOM=T` to the `VENT` line to indicate that the terrain geometry is an immersed boundary and does not conform exactly with the numerical grid. In such a case, the coordinates given by `XB` designate a horizontal patch at the lowest level of the computational domain, and this plane is then projected upward until it cuts through the terrain surface.

There are two options for specifying surface properties for the terrain:

Surface Option 1 The parameters for the 13 original Rothermel-Albini fuel models are listed in Table 17.2. To use fuel model 5, for example, add the following lines:

```
&SURF ID='Wet Brush', VEG_LSET_FUEL_INDEX=5, VEG_LSET_M1=0.06 /
```

This set of inputs invokes fuel model 5, but it also provides you with a way to change the default moisture content of the dead and live fuel components. `M1`, `M10`, and `M100` set the moisture fraction of the 1 h, 10 h, and 100 h dead fuels, and `MLW` and `MLH` set the moisture fraction of the live woody and herbaceous fuels. The default values of these parameters are listed at the bottom of Table 17.2.

²There are other formulations of the level set method besides an elliptical fire front. The parameter `LEVEL_SET_ELLIPSE=T` on the `MISC` line is a placeholder until other methods are implemented.

The 13 fuel models can be modified by specifying a different fuel layer height (VEG_LSET_HT), fuel load (VEG_LSET_SURF_LOAD), and/or fuel packing ratio (VEG_LSET_BETA). The modified properties will replace those in Table 17.2 in the calculation of spread rate following Rothermel-Albini [62, 63]. Note that setting VEG_LSET_SURF_LOAD gives the total fuel load and the amount in different size or dead/live categories will be adjusted to maintain the distribution in the original fuel model.

Surface Option 2 In this second option, you create your own custom fuel type as follows:

```
&SURF ID = 'My Fuel'
    VEG_LSET_ROS_00 = 0.007
    VEG_LSET_SIGMA = 3344.
    VEG_LSET_BETA = 0.0041
    VEG_LSET_HT = 0.91
    VEG_LSET_SURF_LOAD = 0.5
    VEG_LSET_CHAR_FRACTION = 0.25
    VEG_LSET_FIREBASE_TIME = 20 /
```

VEG_LSET_ROS_00 is the no-wind, no-slope rate of spread for the given fuel model. VEG_LSET_SIGMA (m^{-1}) and VEG_LSET_BETA are the average surface-to-volume ratio (σ) and packing ratio (β) of the underlying vegetation. VEG_LSET_HT (m) is the height of the vegetation. When the fire front arrives at a given surface cell, fuel vapors are generated at a rate given by

$$\dot{m}_f'' = (1 - v_{\text{char}}) \frac{m_f''}{\delta t} \text{ kg}/(\text{m}^2 \cdot \text{s}) \quad ; \quad \delta t = \frac{75600}{\sigma} \text{ s} \quad (17.44)$$

Here, m_f'' is the dry fuel loading in units of kg/m^2 (VEG_LSET_SURF_LOAD, default $1.0 \text{ kg}/\text{m}^2$); v_{char} is the char fraction (VEG_LSET_CHAR_FRACTION, default 0.2); and δt is the duration of the fire at a given location. The expression for the burn duration is given by Albini [63], but you can enter your own burn duration using VEG_LSET_FIREBASE_TIME (s).

It is possible to set a custom fuel for which the spread rate does not vary with slope or wind speed. This is done by specifying VEG_LSET_ROS_FIXED=T on the SURF line. The spread rate is then fixed at the no-wind, no-slope value. The main purpose of this feature is for reproduction of manual ignition patterns (e.g. ignition with a drip torch at a fixed walking speed).

17.5.1 Level Set Vegetation Drag

When a surface is defined using level set fuel parameters, a drag force is applied in the gas phase grid cell adjacent to the boundary following the same approach described for the Boundary Fuel Model in Section 17.3. In this case, the fuel surface-area to volume ratio (VEG_LSET_SIGMA), packing ratio (VEG_LSET_BETA), and depth (VEG_LSET_HT) are either obtained from the default fuel models or specified by the user for a custom fuel. The shape factor (SHAPE_FACTOR) and drag coefficient (DRAG_COEFFICIENT) can also be modified from their default values of 0.25 and 2.8, respectively. A comparison of the consistency of the level set drag with the particle and boundary models is included in Figure 17.7.

17.5.2 Simple Test Cases

Examples of Uncoupled Level Set Calculations

Bova et al. [60] compare the level set calculation that has been implemented in FDS with the identical algorithm in the U.S. Forest Service FARSITE model. The three contour plots in Fig. 17.10 match those in Figs. 1(a), 1(b), and 4(a) in the paper. The FDS input files, WUI/Bova_1a.fds, WUI/Bova_1b.fds, and

Table 17.2: Parameters corresponding to the 13 Rothermel-Albini fuel models [62, 63].

	Fuel Type	Dead Fuels						Live Fuels				$M_{x,dead}$	No-Wind, No-Slope RoS (m/s)
		Fine		Medium		Large		Woody		Herbaceous			
		σ m ⁻¹	m'' (kg/m ²)	σ m ⁻¹	m'' (kg/m ²)	σ m ⁻¹	m'' (kg/m ²)	σ m ⁻¹	m'' (kg/m ²)	σ m ⁻¹	m'' (kg/m ²)		
1	Short Grass	11500	0.17	—	—	—	—	—	—	—	—	0.30	0.030
2	Timbergrass	9840	0.45	358	0.22	98	0.11	4920	0.70	4920	0.70	0.30	0.017
3	Tall Grass	4920	0.68	—	—	—	—	—	—	—	—	0.76	0.034
4	Chaparral	6560	1.12	358	0.90	98	0.45	—	—	4920	1.12	1.83	0.035
5	Brush	6560	0.22	358	0.11	—	—	4920	0.45	—	—	0.61	0.010
6	Dormant Brush	5740	0.34	358	0.56	98	0.45	—	—	—	—	0.76	0.013
7	Southern Rough	5740	0.26	358	0.42	98	0.34	4920	0.08	—	—	0.76	0.010
8	Closed Timber Litter	6560	0.34	358	0.22	98	0.56	—	—	—	—	0.06	0.002
9	Hardwood Litter	8200	0.66	358	0.09	98	0.03	—	—	—	—	0.06	0.006
10	Timber	6560	0.68	358	0.45	98	1.12	4920	0.45	—	—	0.30	0.007
11	Light Slash	4920	0.34	358	1.01	98	1.24	—	—	—	—	0.30	0.004
12	Medium Slash	4920	0.90	358	3.15	98	3.71	—	—	—	—	0.70	0.010
13	Heavy Slash	4920	1.57	358	5.17	98	6.29	—	—	—	—	0.91	0.014

For all models, the total mineral content $S_t = 0.056$, the effective mineral content $S_e = 0.01$, the heat of combustion $\Delta H = 18600$ kW/kg, density $\rho_p = 510$ kg/m³, moisture content of fine, medium, and large dead vegetation, $M_{d,1} = 0.03$, $M_{d,2} = 0.04$, $M_{d,3} = 0.05$, moisture content of live woody and herbaceous vegetation $M_{l,w} = M_{l,h} = 0.70$.

WUI/Bova_4a.fds all have the same SURF line that defines the vegetation. Note that in these examples, there is no actual fire nor does the level set calculation depend on the local wind field (LEVEL_SET_MODE=1 on the MISC line). The purpose of the exercise is simply to test the level set algorithm.

```
&MISC ..., LEVEL_SET_MODE = 1 /  
&SURF ID = 'Custom Grass'  
      VEG_LSET_ROS_00 = 0.04  
      VEG_LSET_SIGMA = 11400.  
      VEG_LSET_BETA = 0.0012  
      VEG_LSET_HT = 0.51 /
```

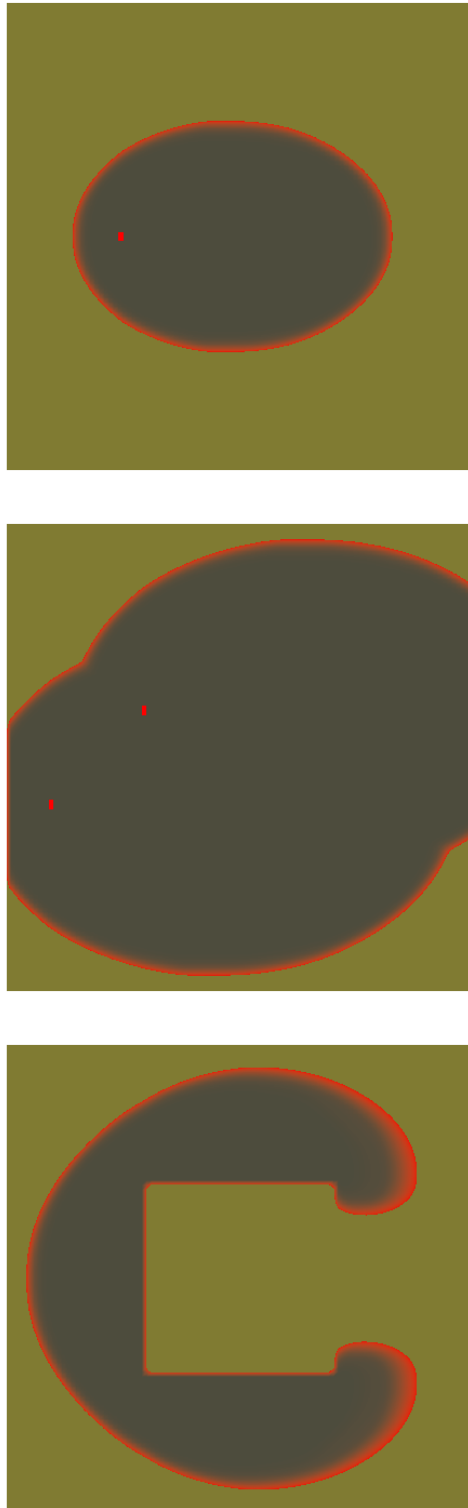


Figure 17.10: Level set test cases from Bova et al. [60], Figs. 1(a), 1(b), and 4(a).

Examples of Coupled Level Set Calculations

In this example, `WUI/level_set_fuel_model_1.fds`, a fire is ignited directly in the center of a square patch of flat terrain, $L = 1000$ m on a side, with no wind. `VEG_LSET_FUEL_INDEX=1` (Short Grass) is selected as the vegetation type. The simulation is performed on a uniform mesh with $\delta x = 25$ m grid cells. For this type of vegetation, the mass loss rate is calculated to be $\dot{m}'' = 0.00955$ kg/m²/s, the heat of combustion is assumed to be $\Delta H = 18607$ kJ/kg, and the local duration of burning is calculated to be $\Delta t = 6.584$ s. These values are all reported in the file `level_set_fuel_model_1.out`. After 10 h of simulation, the fire sweeps over the entire domain, consuming $\dot{m}'' L^2 \Delta t \approx 62877$ kg of vegetation. The left hand side of Fig. 17.11 displays the fire intensity (burning rate) after 5 h, and the right hand side displays the integrated mass loss. Note that in this case the rate of spread is $ROS \approx 0.03$ m/s, which means that the fire front crosses a single grid cell in approximately $\delta x / ROS \approx 830$ s, a much greater time period than the vegetation burning time, Δt . To ensure a relatively smooth fire front and burning rate, the mass loss rate and burning duration are adjusted by dividing and multiplying these values, respectively, by $\delta x / ROS$.

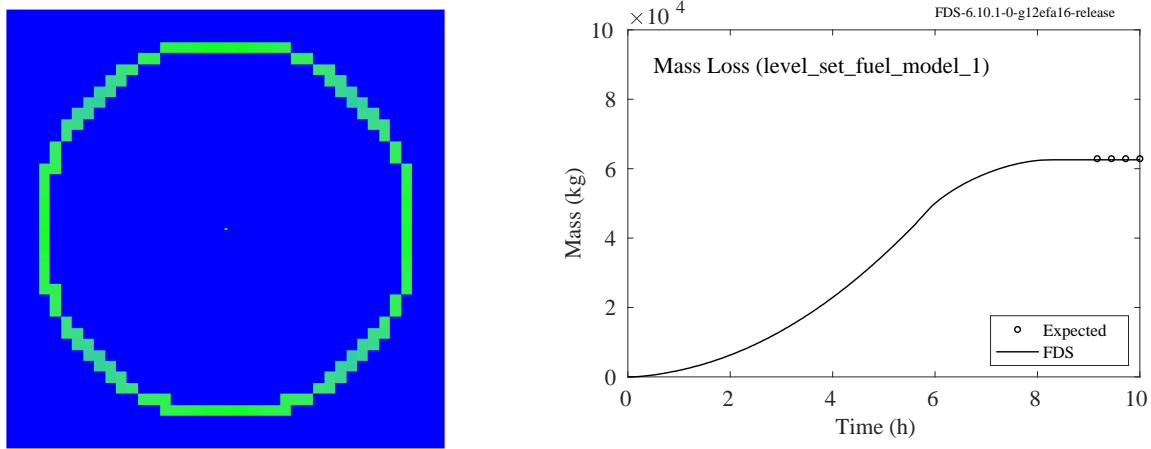


Figure 17.11: Level set test case for Fuel Model 1.

17.5.3 Ember Generation From Surfaces

It is possible to make a burning surface create Lagrangian particles to represent embers by specifying an `EMBER_YIELD` (kg/kg) on the `SURF` line. As this yield requires a burning surface it was developed to be used with `LEVEL_SET_MODE=4`, however, it can also be applied to the Boundary Fuel Model. A `PART_ID` also needs to be specified to provide the properties of the generated embers. Currently, the ember yield is associated with the mass available to burn in the gas phase and does not account for char yield. For example, a fuel load of 1 kg/m² with a char yield of 0.2 and an ember yield of 0.05 would produce 0.2 kg/m² of char, 0.04 kg/m² of embers, and 0.76 kg/m² of fuel vapor.

In many cases, it is not necessary or even practical to track every single ember and a single Lagrangian particle can be used to represent the mass of many embers. This can be accomplished by setting `EMBER_TRACKING_RATIO` on the `SURF` line. If not specified, the default value is 100. You can also specify an `EMBER_GENERATION_HEIGHT` (m) to dictate the vertical offset distance from the burning surface at which particles are produced. This is important in cases where the vertical fuel structure may not be resolved but the height of ember production plays a role in the lofting potential. If a single number is provided for the `EMBER_GENERATION_HEIGHT`, all particles are generated at that offset distance. If two numbers are provided, particles are randomly generated within the specified offset range following a uniform distribution. For

example, if `EMBER_GENERATION_HEIGHT=0.0,1.0` then particles will be produced anywhere up to 1.0 m above the burning surface. Finally, it is recommended to set `VEL_PART=0.0` on the `SURF` line when modeling ember generation so that particles start from rest and are naturally lofted by the surrounding flow.

17.5.4 Ember Ignition

When running a Level Set case, if an airborne ember lands on the ground, there is a chance that it might ignite a new fire. To model this behavior if you are using the level set approach for fire spread, specify the parameter `EMBER_IGNITION_POWER_MEAN` on the `SURF` line describing the ground level vegetation. The ember power, in units of kW, is the net radiative and convective energy generated by the ember. The `EMBER_IGNITION_POWER_MEAN` is the mean value of a normal distribution of ignition propensity; that is, there is a 50 % likelihood that an ember with this power will ignite the surface vegetation. A typical value for the mean is on the order of 0.01 kW (10 W), depending on the ground vegetation. You can also specify the standard deviation of the distribution, `EMBER_IGNITION_POWER_SIGMA`, whose default value is 0.001 kW. The best functional form for ignition probability, and ember ignition itself, is an ongoing topic of research. Therefore, it may be useful to treat this as more of a binary threshold (i.e. use a small standard deviation). Further, a Lagrangian particle may represent more than one ember. Therefore, the particle weighting factor is used to convert the probability of ignition from a single ember to the probability of at least one ignition from the total number of embers represented. The occurrence of ignition will be tested for each time step that a particle is lying on the ground. If ignition occurs, the particle will be removed from the simulation.

17.5.5 Wildfire Spread over Realistic Terrain using QGIS

Wildfire spread over realistic terrain can be modeled in FDS using a plugin to QGIS called `qgis2fds`:

<https://github.com/firetools/qgis2fds>

QGIS is an open source GIS (Geographic Information System) and `qgis2fds` is a Python script developed by Emanuele Gissi and Ruggero Poletto that embeds topography and land use data in an FDS input file.

A set of sample input files created by `qgis2fds` reside within the GitHub repository

<https://github.com/firemodels/cad>

in the folder called `Wildland_Fire_Spread_Example`. There are four cases that demonstrate the four level set fire spread options over a 2 km by 2 km patch of mountainous terrain, as shown in Fig. 17.12. The first image displays a satellite photograph of the area superimposed on the terrain. The second image displays the same terrain covered by 30 m square tiles, each tile representing one of thirteen different land use categories. This data is provided by LANDFIRE (Landscape Fire and Resource Management Planning Tools), a program managed by the U.S. Forest Service and the U.S. Department of the Interior that provides land use and elevation data for the continental United States.

For each case, a hypothetical fire spreads from its point of origin in the lower portion of the chosen area over a ridge in the course of 24 h. The results of the four variants of the level set fire spread method are shown in Fig. 17.13.

A typical input file created with `qgis2fds` includes the latitude and longitude of the origin of the computational domain via `ORIGIN_LAT` and `ORIGIN_LON` on the `MISC` line, in units of decimal degree. Another useful parameter is `NORTH_BEARING` (default 0°) which indicates the direction of true north.

```
&MISC ORIGIN_LAT=35.7207093
      ORIGIN_LON=-83.5212387
      NORTH_BEARING=0.
      TERRAIN_IMAGE='casename.png'
      LEVEL_SET_MODE=1 /
```

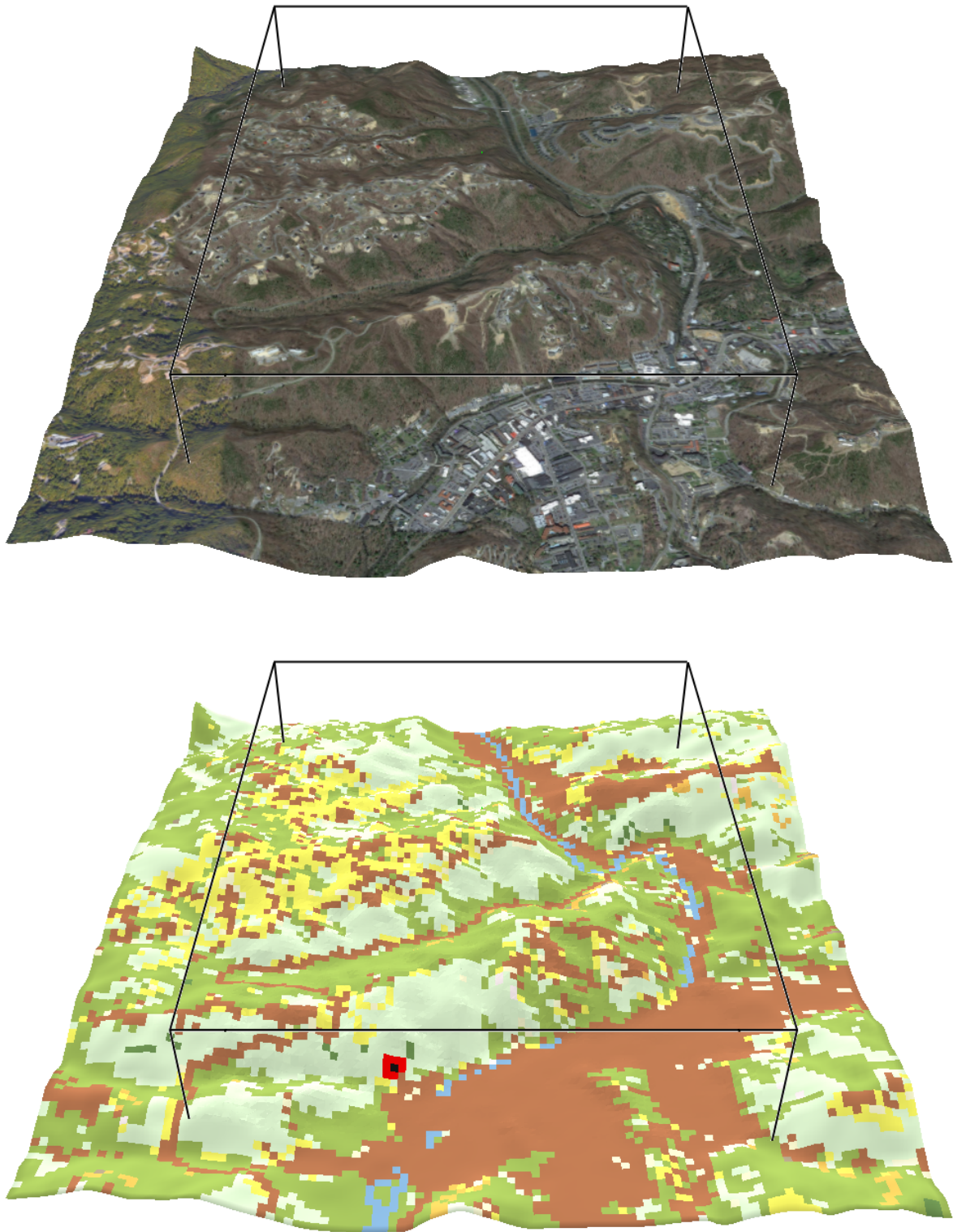



Figure 17.12: (Top) Satellite image of a 2 km by 2 km patch of terrain. (Bottom) 30 m by 30 m tiles indicating the land use of this same area.

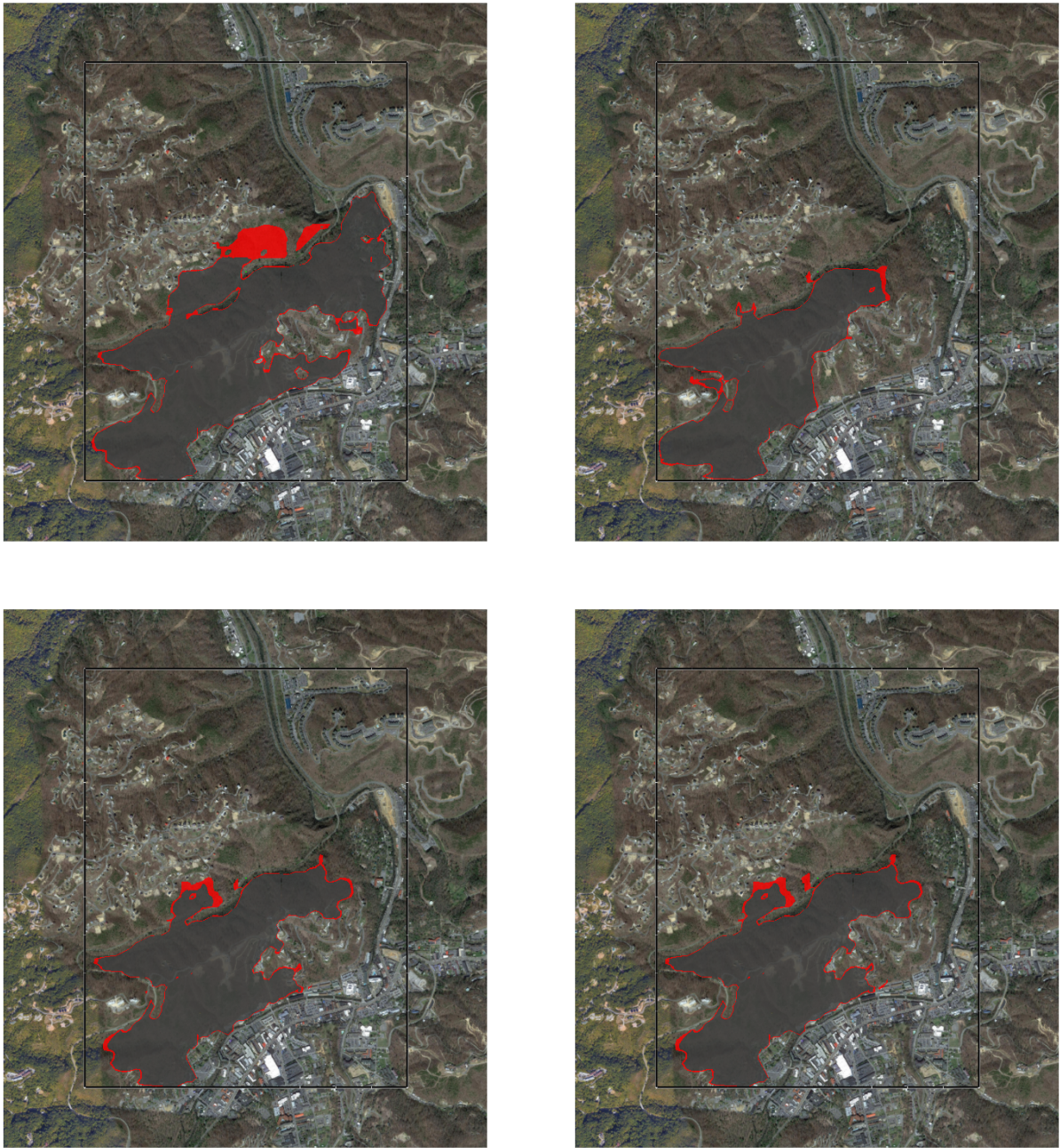


Figure 17.13: Sample wildland fire spread showing the extent of the fire front after 24 h. The plot in the upper left uses the mode 1 of the level set approach, where the wind field does not conform to the terrain but does vary in time. The plot in the upper right uses mode 2, where the terrain-conforming wind field is frozen at the time of ignition. The plot in the lower left uses mode 3, where the wind varies spatially and temporally over the entire course of the fire. The plot in the lower right uses mode 4, where the wind varies with the terrain, but also is influenced by the fire plume.

Chapter 18

Devices and Control Logic

Sprinklers, smoke detectors, heat flux gauges, and thermocouples may seem to be completely unrelated, but from the point of view of FDS, they are simply devices that operate in specific ways depending on the properties assigned to them. They can be used to record some quantity of the simulated environment, like a thermocouple, or they can represent a mathematical model of a complex sensor, like a smoke detector, and in some cases they can trigger events to happen, like a timer.

All devices, in the broadest sense of the word, are designated via the namelist group `DEVC`. In addition, advanced functionality and properties are accommodated via additional namelist groups called `CTRL` (Control) and `PROP` (Properties).

18.1 Device Location and Orientation

Regardless of the specific properties, each device needs to be sited either at a point within the computational domain, or over a span of the domain, like a beam smoke detector. For example, a sprinkler is sited within the domain with a line like:

```
&DEVC XYZ=3.0,5.6,2.3, PROP_ID='Acme Sprinkler 123', ID='Spk_39' /
```

The physical coordinates of the device are given by a triplet of real numbers, `XYZ`. FDS uses these coordinates to determine in which gas or wall cell the device is located. Devices are evaluated using cell centered or face centered values of the cell the device is located in; no interpolation is done. The properties of the device are contained on the `PROP` line designated by `PROP_ID`, which will be explained below for each of the special devices included in FDS. The character string `ID` is merely a descriptor to identify the device in the output files, and if any action is tied to its activation.

Not all devices need to be associated with a particular set of properties via the `PROP_ID`. For example, pointwise output quantities are specified with a single `DEVC` line, like

```
&DEVC ID='TC-23', XYZ=3.0,5.6,2.3, QUANTITY='TEMPERATURE' /
```

which tells FDS to record the temperature at the given point as a function of time. The `ID` is a label in the output file whose name is `CHID_devic.csv`. Note that FDS outputs the data stored for that cell without performing any interpolation with surrounding cells.

Some devices have a particular orientation. The parameter `IOR` (Index of Orientation) is required for any device that is placed on the surface of a solid. The values ± 1 or ± 2 or ± 3 indicate the direction that the device “points.” For example, `IOR=-1` means that the device is mounted on a wall that faces in the negative x direction. `ORIENTATION` is used for devices that are not on a surface and require a directional

specification, like a sprinkler. `ORIENTATION` is specified with a triplet of real number values that indicate the components of the direction vector. The default value of `ORIENTATION` is (0,0,-1). For example, a default downward-directed sprinkler spray can be redirected in other direction. If you were to specify

```
&DEVC XYZ=3.0,5.6,2.3, PROP_ID='...', ID='...', ORIENTATION=0.707,0.707,0.0 /
```

the sprinkler would point in the direction halfway between the positive x and y directions. For other devices, the `ORIENTATION` would only change the way the device is drawn by Smokeview.

The delivered density to the floor from a sprinkler depends upon where the sprinkler arms are located. Rather than redefining the spray pattern for every possible direction that the sprinkler can be attached to the pipe, the `DEVC` can be given the parameter `ROTATION`. The default `ROTATION` is 0 degrees, which for a downwards pointing sprinkler is the positive x -axis. Positive `ROTATION` will rotate the 0 degree point towards the positive y -axis.

18.2 Device Output

Each device has a `QUANTITY` associated with it. The time history of each `DEVC` quantity is output to a comma-delimited ASCII file called `CHID_devc.csv` (see Sec. 27.3 for output file format). This file can be imported into most spreadsheet software packages. Most spreadsheet programs limit the number of columns to some number (for example the 2003 version Microsoft Excel had a 256 column limit). As a default, FDS places no limit on the amount of columns in a comma-separated value (.csv) file. If your spreadsheet application allows fewer columns than the number of `DEVC` or `CTRL` in your input file then set `COLUMN_DUMP_LIMIT` equal to `T` on the `DUMP` line. Use `DEVC_COLUMN_LIMIT` and `CTRL_COLUMN_LIMIT` to indicate the limit of columns in the device and control output files. Their default values are 254. If more devices or controls are present than the limit, then multiple output files will be written. The file name will have a number appended to it. For example, if two device file are required, they will be named `CHID_1_devc.csv` and `CHID_2_devc.csv`.

The `DEVC` output is written to a file every `DT_DEVC` seconds or at discrete times indicated by `RAMP_DEVC`, both of which are specified on the `DUMP` line. By default, the output `QUANTITY` is time-averaged between printouts. To prevent this, add `TIME_AVERAGED=F` to the `DEVC` line.

A useful option for the `DEVC` line is to add `RELATIVE=T`, which will indicate that only the change in the initial value of the `QUANTITY` is to be output. This can be useful for verification and validation studies. You can also output the absolute value of the device quantity by setting `ABSOLUTE_VALUE` to `T` on the `DEVC` line.

You can change the values of the output `QUANTITY` by multiplying by `CONVERSION_FACTOR` and/or adding `CONVERSION_ADDEND`. For example, to change temperature output from the default $^{\circ}\text{C}$ to $^{\circ}\text{F}$, the `CONVERSION_FACTOR` should be set to 1.8 and the `CONVERSION_ADDEND` to 32. You can then change the units that appear in the output files by setting `UNITS='F'` or whatever the case may be. If you do not convert the output `QUANTITY` from its default value, you need not set the `UNITS`.

If you do not want the `DEVC QUANTITY` to be included in the output file, set `OUTPUT=F` on the `DEVC` line. Sometimes, devices are just used as clocks or control devices. In these cases, you might want to prevent its output from cluttering the output file. If the `DEVC QUANTITY='TIME'`, then `OUTPUT` is set to `F` automatically.

All devices must have a specified `QUANTITY`. Some special devices (Sec. 18.3) have their `QUANTITY` specified on a `PROP` line. A `QUANTITY` specified on a `PROP` line associated with a `DEVC` line will override a `QUANTITY` specified on the `DEVC` line.

18.3 Special Device Properties

Many devices are fairly easy to describe, like a point measurement, with only a few parameters which can be included on the `DEVC` line. However, for more complicated devices, it is inconvenient to list all of the properties on each and every `DEVC` line. For example, a simulation might include hundreds of sprinklers, but it is tedious to list the properties of the sprinkler each time the sprinkler is sited. For these devices, use a separate namelist group called `PROP` to store the relevant parameters. Each `PROP` line is identified by a unique `ID`, and invoked by a `DEVC` line by the string `PROP_ID`. The best way to describe the `PROP` group is to list the various special devices and their properties.

18.3.1 Sprinklers

To specify one or more sprinklers, you need to specify several different groups of parameters that fall into a variety of namelist groups. For example, here is a basic sprinkler description:

```
&SPEC ID='WATER VAPOR' /
&PART ID='my droplets', DIAMETER=1000., SPEC_ID='WATER VAPOR' /
&PROP ID='K-11', QUANTITY='SPRINKLER LINK TEMPERATURE', RTI=148., C_FACTOR=0.7,
      ACTIVATION_TEMPERATURE=74., OFFSET=0.10, PART_ID='my droplets', FLOW_RATE=189.3,
      PARTICLE_VELOCITY=10., SPRAY_ANGLE=30.,80., SMOKEVIEW_ID='sprinkler_upright' /
&DEVC ID='Spr-1', XYZ=22.8,19.7,7.4, PROP_ID='K-11' /
&DEVC ID='Spr-2', XYZ=22.8,22.7,7.4, PROP_ID='K-11' /
```

A sprinkler, known as 'Spr-1', is located at a point in space given by `XYZ`. It is a 'K-11' type sprinkler, whose properties are given on the `PROP` line. Note that the various names (`IDS`) mean nothing to FDS, except as a means of associating one thing with another, so try to use `IDS` that are meaningful. The parameter `QUANTITY='SPRINKLER LINK TEMPERATURE'` *does* have a specific meaning to FDS, directing it to compute the activation of the device using the standard RTI (Response Time Index [65]) algorithm. Properties associated with sprinklers included in the `PROP` group are:

- `RTI` Response Time Index in units of $(\text{m}\cdot\text{s})^{1/2}$. (Default 100.)
- `C_FACTOR` Conduction Factor in units of $(\text{m}/\text{s})^{1/2}$. (Default 0.)
- `ACTIVATION_TEMPERATURE` in units of $^{\circ}\text{C}$. (Default 74°C)
- `INITIAL_TEMPERATURE` of the link in units of $^{\circ}\text{C}$. (Default `TMPA`)
- `FLOW_RATE` in units of L/min. This parameter is only appropriate for liquid droplets. An alternative is to provide the `K_FACTOR` in units of $\text{L}/(\text{min}\cdot\text{bar}^{1/2})$ and the `OPERATING_PRESSURE`, the gauge pressure at the sprinkler, in units of bar. The flow rate is then given by $K\sqrt{P}$. Note that 1 bar is equivalent to 14.5 psi, 1 gpm (gal/min) is equivalent to 3.785 L/min, $1\text{ gpm}/\text{psi}^{1/2}$ is equivalent to $14.41\text{ L}/\text{min}/\text{bar}^{1/2}$.
- `MASS_FLOW_RATE` in units of kg/s. This parameter can be used instead of `FLOW_RATE` for liquid droplets, but it must be used for solid particles. If `MASS_FLOW_RATE` is specified, the `PARTICLE_VELOCITY` must also be specified.
- `OFFSET` Radius (m) of a sphere surrounding the sprinkler where the water droplets are initially placed in the simulation (Default 0.05 m). It is assumed that beyond the `OFFSET` the droplets have completely broken up and are transported independently of each other. Do not locate a sprinkler or nozzle within `OFFSET` meters of a mesh boundary. If you do, the droplets introduced into the adjacent mesh will be rejected. The entire spray pattern of the sprinkler need not lie within one mesh, but the volume immediately surrounding the sprinkler itself must lie in one mesh.

- **PARTICLE_VELOCITY** Initial droplet velocity. (Default 0 m/s)
- **ORIFICE_DIAMETER** Diameter of the nozzle orifice in m (default 0 m). This input provides an alternative way to set droplet velocity by giving values for **FLOW_RATE** and **ORIFICE_DIAMETER**, in which case the droplet velocity is computed by dividing the flow rate by the orifice area. Use this method if you do not have any information about droplet velocity. However, quite often you must fine-tune the **PARTICLE_VELOCITY** in order to reproduce a particular spray profile. The **ORIFICE_DIAMETER** is not used if either **PARTICLE_VELOCITY** or **SPRAY_PATTERN_TABLE** is specified.
- **SPRAY_ANGLE** A pair of angles (in degrees) through which the droplets are sprayed. The angles outline a conical spray pattern relative to the south pole of the sphere centered at the sprinkler with radius **OFFSET**. For example, **SPRAY_ANGLE=30., 80.** directs the water spray through a band between 30° and 80° of the **ORIENTATION** vector, which is (0,0,-1) by default (see Figure 18.1). Elliptical spray patterns can be specified via a pair of spray angles. For example, **SPRAY_ANGLE(1:2,1)=0., 60.** and **SPRAY_ANGLE(1:2,2)=0., 30.**, defines a spray pattern with 60 degree angle in the direction of the *x* axis and a 30 degree angle in the direction of the *y* axis. From above, the spray pattern resembles an ellipse. **SPRAY_PATTERN_SHAPE** determines how the droplets are distributed within the specified **SPRAY_ANGLE**. Choices are 'UNIFORM' and 'GAUSSIAN'. The default distribution is 'GAUSSIAN'. The parameter **SPRAY_PATTERN_MU** controls the latitude of the maximum density of droplets for the 'GAUSSIAN' distribution. The width of the distribution is controlled by the parameter **SPRAY_PATTERN_BETA**.
- **SPRAY_PATTERN_TABLE** Name of a set of **TABL** lines containing the description of the spray pattern.
- **PART_ID** The name of the **PART** line containing properties of the droplets. See Chapter 15 for additional details.
- **PRESSURE_RAMP** The name of the **RAMP** lines specifying the dependence of pipe pressure on the number of active sprinklers and nozzles.
- **SMOKEVIEW_ID** The name of a drawing of a sprinkler to include in the Smokeview animation.

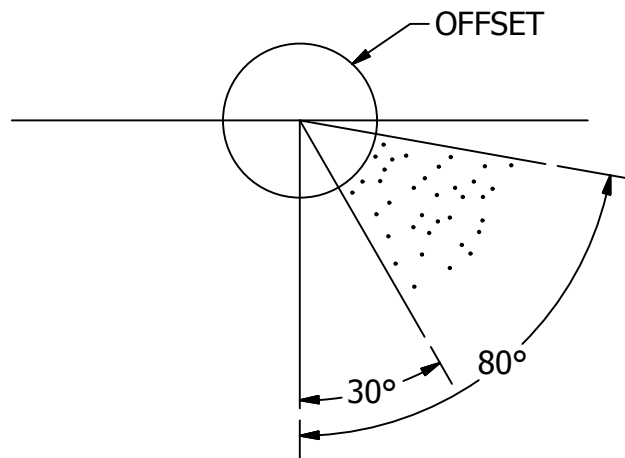


Figure 18.1: Sketch showing the role of **OFFSET** and **SPRAY_ANGLE**.

Specifying Complex Spray Patterns

As an example of the more advanced sprinkler options, a sprinkler with an elliptical spray pattern and uniform mass flux distribution within the spray angle is given by:

```
&PROP ... SPRAY_ANGLE(1:2,1)=0.,60., SPRAY_ANGLE(1:2,2)=0.,30.,  
        SPRAY_PATTERN_SHAPE='UNIFORM' /
```

For full-cone sprays, the parameter `SPRAY_PATTERN_MU` is set to zero by default. For hollow-cone sprays it is set to the average of `SPRAY_ANGLE(1:2,1)`, the spray angle in the x direction. The following example uses `SPRAY_PATTERN_MU` to define a spray that is somewhere between a full-cone and a hollow-cone spray:

```
&PROP ... SPRAY_ANGLE=0.,30., SPRAY_PATTERN_MU=15. /
```

Figure 18.2 gives a sense of how the parameters `SPRAY_PATTERN_MU` and `SPRAY_PATTERN_BETA` affect the distribution. In each row of the figure, the maximum angle, `SPRAY_ANGLE(2,1)`, is varied from 22.5 to 45 degrees. Each plot varies the spread parameter, `SPRAY_PATTERN_BETA`, from 0 (uniform) to 1000 (narrow distribution). The peak angle of the distribution, `SPRAY_PATTERN_MU`, is varied from 0 to 10 to 20 as you go from the top to the bottom row.

If a more complex spray pattern is desired than one characterized by a `SPRAY_ANGLE`, then a `SPRAY_PATTERN_TABLE` can be specified using the `TABL` namelist group. Specify the total flow using `FLOW_RATE` on the `PROP` line, the name of the spray pattern using `SPRAY_PATTERN_TABLE` and then one or more `TABL` lines of the form:

```
&TABL ID='table_id', TABLE_DATA=LAT1,LAT2,LON1,LON2,VELO,FRAC /
```

where each `TABL` line for a given 'table_id' provides information about the spherical distribution of the spray pattern for a specified solid angle. `LAT1` and `LAT2` are the bounds of the solid angle measured in degrees from the south pole (0 is the south pole and 90 is the equator, 180 is the north pole). Note that this is not the conventional way of specifying a latitude, but rather a convenient system based on the fact that a typical sprinkler sprays water downward, which is why 0 degrees is assigned to the “south pole,” or the $-z$ direction. The parameters `LON1` and `LON2` are the bounds of the solid angle (also in degrees), where 0 (or 360) is aligned with the $-x$ axis and 90 is aligned with the $-y$ axis. `VELO` is the velocity (m/s) of the droplets at their point of insertion. `FRAC` the fraction of the total flow rate of liquid that should emerge from that particular solid angle.

In the test case called `bucket_test_2`, the spray consists of two jets, each with a velocity of 5 m/s and a combined flow rate of 60 L/min. The sprinkler is set to operate for only 5 s. The first jet contains 0.2 of the total flow, the second, 0.8 of the total. The jets are centered at points 60° below the “equator,” and are separated by 180°.

```
&PROP ... FLOW_RATE=60., SPRAY_PATTERN_TABLE='TABLE1' /  
&TABL ID='TABLE1', TABLE_DATA=30,31, 0, 1,5,0.2 /  
&TABL ID='TABLE1', TABLE_DATA=30,31,179,180,5,0.8 /
```

Note that each set of `TABL` lines must have a unique `ID`. Also note that the `TABL` lines can be specified in any order. Figure 18.3 verifies that the sprinkler releases 5 kg of water (1 kg/s for 5 s).

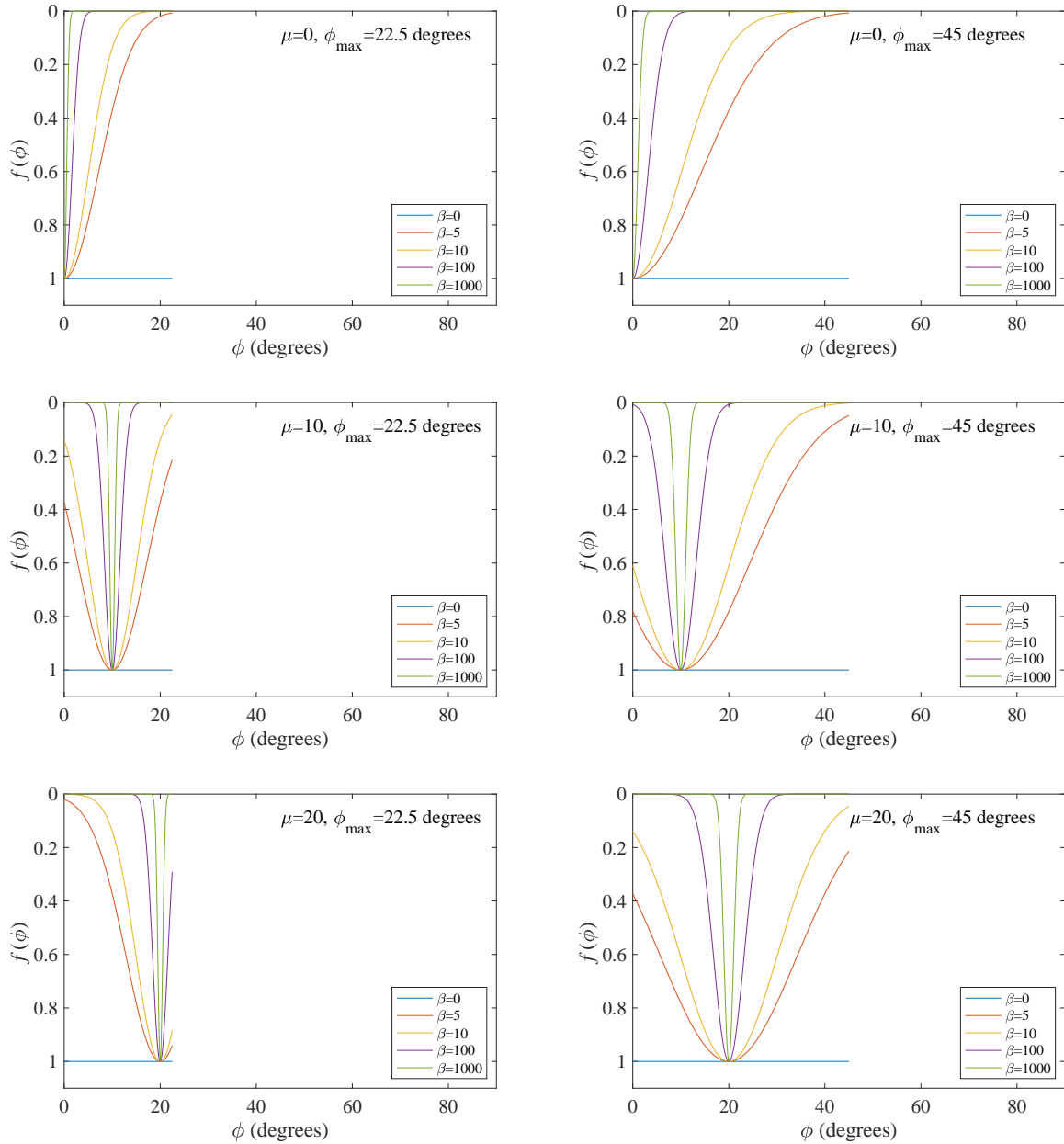


Figure 18.2: Spray pattern parameters, peak and spread.

Varying Pipe Pressure

In real sprinkler systems, the pipe pressure is affected by the number of actuated sprinklers. Typically, the pressure is higher than the design value when the first sprinkler activates, and decreases as more and more sprinklers are activated. The pipe pressure has an effect on flow rate, droplet velocity and droplet size distribution. In FDS, the varying pipe pressure can be specified on a `PROP` line using `PRESSURE_RAMP`. On each `RAMP` line, the number of open sprinklers or nozzles is associated with certain pipe pressure (bar). For example:

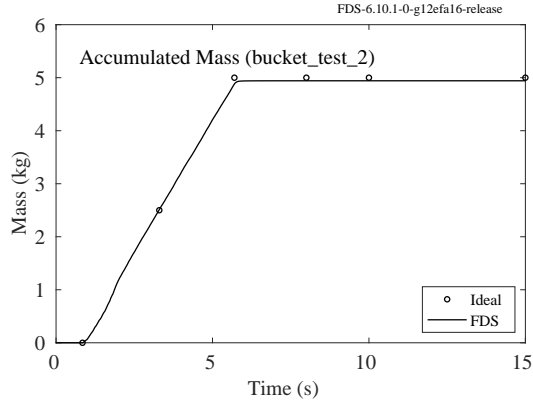


Figure 18.3: Accumulated water collected at the floor in the `bucket_test_2` case.

```
&PROP ID='My nozzle'
  OFFSET=0.1
  PART_ID='water drops'
  FLOW_RATE=0.9
  OPERATING_PRESSURE = 10.0
  PARTICLE_VELOCITY=15.0
  SPRAY_ANGLE=0.0,80.0
  PRESSURE_RAMP = 'PR1' /

&RAMP ID = 'PR1' T = 1, F = 16. /
&RAMP ID = 'PR1' T = 2, F = 10. /
&RAMP ID = 'PR1' T = 3, F = 8. /
```

These lines would indicate that the pressure is 16 bar when the first sprinkler activates, 10 bar when two sprinklers are active, and 8 bar when three or more sprinklers are active. When counting the number of active sprinklers, FDS accounts for all active sprinklers or nozzles with a given `PART_ID`.

When pressure ramps are used, both `FLOW_RATE` and `PARTICLE_VELOCITY` are dependent on the `OPERATING_PRESSURE`. Specify either the `FLOW_RATE`, or the `K_FACTOR` and `OPERATING_PRESSURE`. In the latter case, the flow rate is given by $K\sqrt{p}$ and the droplet velocity by using the liquid density and the `ORIFICE_DIAMETER`. If spray pattern table is used, the droplet velocity is determined separately for each line of the table by applying $K\sqrt{p}$ and the `ORIFICE_DIAMETER`. The median diameter of the particle size distribution is scaled as $d_m(p) = d_m(p_o)(p_o/p)^{1/3}$, where p_o is the `OPERATING_PRESSURE` and $d_m(p_o)$ is specified by parameter `DIAMETER` on the corresponding `PART` line.

For some simulations there may be groups of independent sprinklers or nozzles. For example one might have one set of nozzles for a fuel spray and a second set for water spray. In this case the flow of water would not be impacted by how many fuel spray nozzles are open. To have the `PRESSURE_RAMP` only count a subset of sprinklers or nozzles, the keyword `PIPE_INDEX` can be used on the `DEVC` line. For example:

```
&DEVC ID='Spr_1', XYZ=2.00,2.00,8.00, PROP_ID='My nozzle', PIPE_INDEX=1 /
&DEVC ID='Spr_2', XYZ=1.00,1.00,8.00, PROP_ID='My nozzle', PIPE_INDEX=1 /
&DEVC ID='Fuel_1', XYZ=2.00,2.00,1.00, PROP_ID='Fuel Spray', PIPE_INDEX=2 /
&DEVC ID='Fuel_2', XYZ=1.00,1.00,1.00, PROP_ID='Fuel Spray', PIPE_INDEX=2 /
```

These lines indicate that the fuel spray nozzles are a separate pipe network from the water sprinklers. With these inputs, a `PRESSURE_RAMP` for the water sprinklers would not count any active fuel spray nozzles. See the example case `flow_rate_2` in the Verification Guide for further details on the use of `PIPE_INDEX`.

18.3.2 Nozzles

Nozzles are very much like sprinklers, only they do not activate based on the standard RTI (Response Time Index) model. To simulate a nozzle that activates at a given time, specify a `QUANTITY` and `SETPOINT` directly on the `DEVC` line. The following lines:

```
&DEVC XYZ=23.91,21.28,0.50, PROP_ID='nozzle', ORIENTATION=0,0,1, QUANTITY='TIME',  
      SETPOINT=0., ID='noz_1' /  
&DEVC XYZ=26.91,21.28,0.50, PROP_ID='nozzle', ORIENTATION=0,0,1, QUANTITY='TIME',  
      SETPOINT=5., ID='noz_2' /  
&PROP ID='nozzle', PART_ID='heptane drops', FLOW_RATE=2.132,  
      FLOW_TAU=-50., PARTICLE_VELOCITY=5., SPRAY_ANGLE=0.,45. /
```

designate two nozzles of the same type, one which activates at time zero, the other at 5 s. Note that nozzles must have a designated `PROP_ID`, and the `PROP` line must have a designated `PART_ID` to describe the liquid droplets.

Example Case: Setting the Flow Rate of a Nozzle

This example demonstrates the use of pressure ramps and control logic for opening and closing nozzles. It also serves as a verification test for the water flow rate. There are four nozzles that open at designated times: 0 s, 15 s, 30 s and 45 s. At time 60 s, all the nozzles are closed. The number of open nozzles is measured using a device with quantity `'OPEN NOZZLES'`. A comparison of the FDS result and the exact, intended values is shown in Fig. 18.4. Note that `'OPEN NOZZLES'` counts only nozzles belonging to the specified `PIPE_INDEX`. The pressure ramp has been designed to deliver a total flow rate of 10 L/min at all values of open nozzles. Mathematically this means that

$$nK\sqrt{p} = 10 \text{ L/min} \quad \Rightarrow \quad p = \left(\frac{10 \text{ L/min}}{nK} \right)^2 \quad (18.1)$$

where n is the number of open nozzles. The corresponding nozzle and pressure ramp definitions are

```
&PROP ID='Head', OFFSET=0.10, PART_ID='water drops', K_FACTOR=2.5, OPERATING_  
PRESSURE=1.,  
      PRESSURE_RAMP='PR', PARTICLE_VELOCITY=2., SPRAY_ANGLE= 0.,60. /  
  
&RAMP ID='PR', T= 1., F=16. /  
&RAMP ID='PR', T= 2., F=4. /  
&RAMP ID='PR', T= 3., F=1.778 /  
&RAMP ID='PR', T= 4., F=1. /
```

The water is tracked using a device measuring the accumulated mass per unit area, integrated over the total floor area. The total mass of water should increase from zero to 10 kg in 60 s. A comparison of the FDS prediction and this analytical result is shown in Fig. 18.4. The slight delay of the FDS result is caused by the time it takes from the droplets to fall down on the floor.

18.3.3 Specified Entrainment (Velocity Patch)

The details of the sprinkler head geometry and spray atomization are practically impossible to resolve in a fire calculation. As a result, the local gas phase entrainment by the sprinkler is difficult to predict. As an alternative, it is possible to specify the local gas velocity in the vicinity of the sprinkler nozzle. The `PROP` line may be used to specify a polynomial function for a specific velocity component and this function may

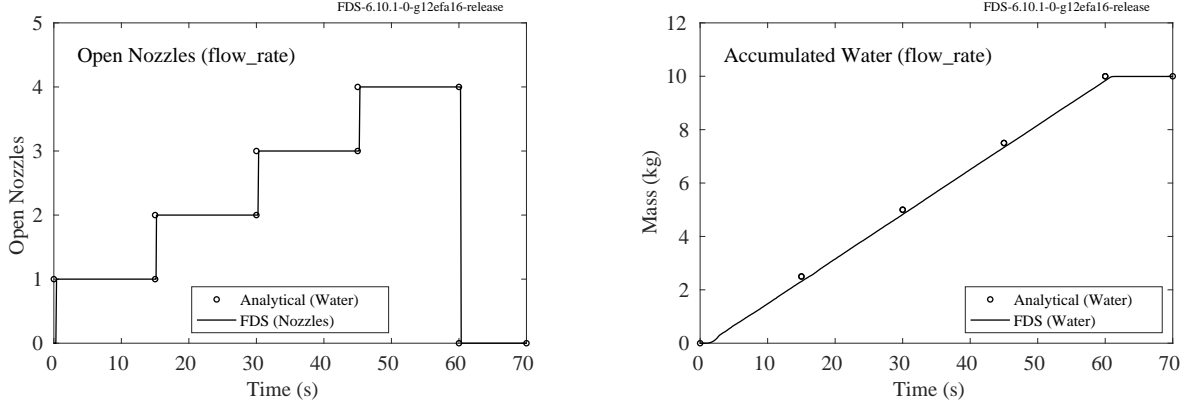


Figure 18.4: Output of the `flow_rate` test case.

be “patched” into the flow field using a device. This device is given the quantity `'VELOCITY PATCH'` and is initially inactive. The velocity patch must be activated with a separate control device, as discussed in Section 18.4. You specify the local region for the velocity patch using `XB` for the device. The polynomial is defined as a second-order Taylor expansion about the point `XYZ` (the default value of `XYZ` is the center of `XB`). FDS then uses an immersed boundary method to force the local velocity component to satisfy the polynomial. The polynomial is specified by the coefficients `P0`, `PX(1:3)`, and `PXX(1:3,1:3)`, which represent, respectively, the value of the k th velocity component, the first derivatives, and the second derivatives at point `XYZ`. Note that the first derivatives are represented by a three component array and the second derivatives are represented by a symmetric 3×3 array—only the upper triangular part needs to be specified. The polynomial is given by (note that summation of repeated indices is implied):

$$u_k(\mathbf{r}) = \underbrace{(u_k)_0}_{P0} + \underbrace{r_i \left(\frac{\partial u_k}{\partial x_i} \right)_0}_{PX(1:3)} + \frac{r_i r_j}{2} \underbrace{\left(\frac{\partial^2 u_k}{\partial x_i \partial x_j} \right)_0}_{PXX(1:3,1:3)} \quad (18.2)$$

The vector \mathbf{r} is the position of the velocity storage location relative to the point `XYZ`. The specific velocity component is specified on `PROP` by the integer `VELOCITY_COMPONENT`. Below we provide an example set of `PROP` and `DEVC` lines to specify a parabolic profile for the vertical component of velocity.

```
&PROP ID='p1', VELOCITY_COMPONENT=3, P0=-1,PXX(1,1)=5,PXX(2,2)=5 /
&DEVC XB=-.1,.1,-.1,.1,.9,.95, QUANTITY='VELOCITY PATCH',PROP_ID='p1', DEVC_ID='t1' /
&DEVC ID='t1', XYZ=0,0,.9, QUANTITY='TIME', SETPOINT=10/
```

In this example, a velocity patch is activated at 10 s in the simulation. Any w components of velocity with staggered storage locations within the box `XB=-.1,.1,-.1,.1,.9,.95` will be driven toward the value specified by the polynomial profile `'p1'`. You must ensure that the device box encompasses the staggered storage locations (see the theory manual [3] for a discussion on the face-centered velocity storage locations).

18.3.4 Heat Detectors

`QUANTITY='LINK TEMPERATURE'` defines a heat detector, which uses essentially the same activation algorithm as a sprinkler, without the water spray.

```
&DEVC ID='HD_66', PROP_ID='Acme Heat', XYZ=2.3,4.6,3.4 /
```

```
&PROP ID='Acme Heat', QUANTITY='LINK TEMPERATURE', RTI=132., ACTIVATION_
TEMPERATURE=74. /
```

Like a sprinkler, RTI is the Response Time Index in units of $\sqrt{m \cdot s}$. $ACTIVATION_TEMPERATURE$ is the link activation temperature in degrees Celsius (Default 74 °C). $INITIAL_TEMPERATURE$ is the initial temperature of the link in units of °C (Default T_{MFA}).

18.3.5 Smoke Detectors

A smoke detector is defined in the input file with an entry similar to:

```
&DEVC ID='SD_29', PROP_ID='Acme Smoke Detector', XYZ=2.3,4.6,3.4 /
&PROP ID='Acme Smoke Detector', QUANTITY='CHAMBER OBSCURATION', LENGTH=1.8,
ACTIVATION_OBSCURATION=3.24 /
```

for the single parameter Heskestad model. Note that a `PROP` line is mandatory for a smoke detector, in which case the `DEVC QUANTITY` can be specified on the `PROP` line. For the four parameter Cleary model, use a `PROP` line like:

```
&PROP ID='Acme Smoke Detector I2', QUANTITY='CHAMBER OBSCURATION',
ALPHA_E=1.8, BETA_E=-1.1, ALPHA_C=1.0, BETA_C=-0.8,
ACTIVATION_OBSCURATION=3.24 /
```

where the two characteristic filling or “lag” times are of the form:

$$\delta t_e = \alpha_e u^{\beta_e} \quad ; \quad \delta t_c = \alpha_c u^{\beta_c} \quad (18.3)$$

The default detector parameters are for the Heskestad model with a characteristic `LENGTH` of 1.8 m. For the Cleary model, the `ALPHAS` and `BETAS` must all be listed explicitly. Suggested constants for unidentified ionization and photoelectric detectors presented in Table 18.1. `ACTIVATION_OBSCURATION` is the threshold value in units of %/m. The threshold can be set according to the setting commonly provided by the manufacturer. The default setting¹ is 3.24 %/m (1 %/ft).

Table 18.1: Suggested values for smoke detector model [66]. See Ref. [67] for others.

Detector	α_e	β_e	α_c, L	β_c
Cleary Ionization I1	2.5	-0.7	0.8	-0.9
Cleary Ionization I2	1.8	-1.1	1.0	-0.8
Cleary Photoelectric P1	1.8	-1.0	1.0	-0.8
Cleary Photoelectric P2	1.8	-0.8	0.8	-0.8
Heskestad Ionization	—	—	1.8	—

¹Note that the conversion of obscuration from units of %/ft to %/m is given by:

$$O[\%/m] = \left[1 - \left(1 - \frac{O[\%/ft]}{100} \right)^{3.28} \right] \times 100 \quad (18.4)$$

Defining Smoke

By default, FDS assumes that the smoke from a fire is generated in direct proportion to the heat release rate. A value of `SOOT_YIELD=0.01` on the `REAC` line means that the smoke generation rate is 0.01 of the fuel burning rate. The “smoke” in this case is not explicitly tracked by FDS, but rather is assumed to be a function of the combustion products lumped species.

Suppose, however, that you want to define your own “smoke” and that you want to specify its production rate independently of the HRR (or even in lieu of an actual fire, like a smoldering source). You might also want to define a mass extinction coefficient for the smoke and an assumed molecular weight (as it will be tracked like a gas species). Finally, you also want to visualize the smoke using the `SMOKE3D` feature in Smokeview (Sec. 22.8). Use the following lines:

```
&SPEC ID='MY SMOKE', MW=29., MASS_EXTINCTION_COEFFICIENT=8700. /
&SURF ID='SMOLDER', TMP_FRONT=1000., MASS_FLUX(1)=0.0001, SPEC_ID='MY SMOKE',
    COLOR='RED' /
&VENT XB=0.6,1.0,0.3,0.7,0.0,0.0, SURF_ID='SMOLDER' /

&PROP ID='Acme Smoke', QUANTITY='CHAMBER OBSCURATION', SPEC_ID='MY SMOKE' /
&DEVC XYZ=1.00,0.50,0.95, PROP_ID='Acme Smoke', ID='smoke_1' /

&SM3D QUANTITY='DENSITY', SPEC_ID='MY SMOKE' /
```

The same smoke detector model is used that was described above, but now, the species 'MY SMOKE' is used in the algorithm, rather than that associated with the lumped species. Note that your species will not participate in the radiation calculation. It will merely serve as a surrogate for smoke.

18.3.6 Beam Detection Systems

A beam detector can be defined by specifying the endpoints, $(x1, y1, z1)$ and $(x2, y2, z2)$, of the beam and the total percent obscuration at which the detector activates. FDS determines which mesh cells lie along the linear path defined by the two endpoints. The beam detector response is evaluated as

$$\text{Obscuration} = \left(1 - \exp \left(-K_m \sum_{i=1}^N \rho_{s,i} \Delta x_i \right) \right) \times 100 \% \quad (18.5)$$

where i is a mesh cell along the path of the beam, $\rho_{s,i}$ is the soot density of the mesh cell, Δx_i is the distance within the mesh cell that is traversed by the beam, and K_m is the mass extinction coefficient. The line in the input file has the form:

```
&DEVC XB=x1,x2,y1,y2,z1,z2, QUANTITY='PATH OBSCURATION', ID='beam1', SETPOINT=33.0 /
```

A similar `QUANTITY` is 'TRANSMISSION' which is given by the following expression:

$$\text{Transmission} = \exp \left(-K_m \frac{L_0}{L} \sum_{i=1}^N \rho_{s,i} \Delta x_i \right) \times 100 \% / \text{m} \quad (18.6)$$

Note that the transmission is given in units of %/m rather than % like obscuration. L is the total path length of the beam, and L_0 is the reference dimension of 1 m.

Example Case: A Beam Detector

A 10 m by 10 m by 4 m compartment is filled with air at 20 °C with a density of $\rho = 1.195 \text{ kg/m}^3$. A uniformly distributed gas species with a mass fraction of $Y_s = 3 \times 10^{-5}$ represents smoke. The “smoke” density

is $\rho_s \equiv \rho Y_s = 3.585 \times 10^{-5} \text{ kg/m}^3$. Using the default mass extinction coefficient of $K_m = 8700 \text{ m}^2/\text{kg}$, the optical depth is calculated to be $K \equiv K_m \rho_s = 0.3119 \text{ m}^{-1}$. Three beam detectors are located in the compartment, all with a path length of $L = 10 \text{ m}$ but with different orientations. The expected path obscuration is $100(1 - \exp(-KL)) = 95.58 \%$. Note that this case uses 8 meshes to span the computational domain, and the beams are permitted to pass from one mesh to another.

This case also provides a check on the Smokeview rendering of the smoke-filled compartment. Nine columns are located at increasing distance from the front wall in increments of 1 m. Equation (22.23) predicts a visibility distance of $S \equiv 3/K = 9.6 \text{ m}$. Referring to Fig. 18.5, you should just barely see the ninth column at a distance of 8.9 m from the front wall. The plot at the bottom of Fig. 18.5 compares the pixel values used by Smokeview to color the columns versus the expected values. The pixel values are integers between 0 and 255, where 255 is white and 0 is black.

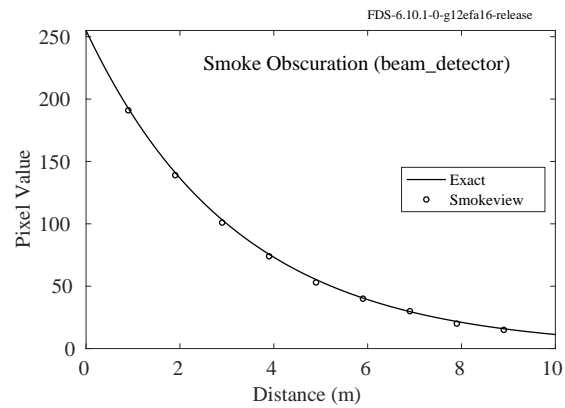
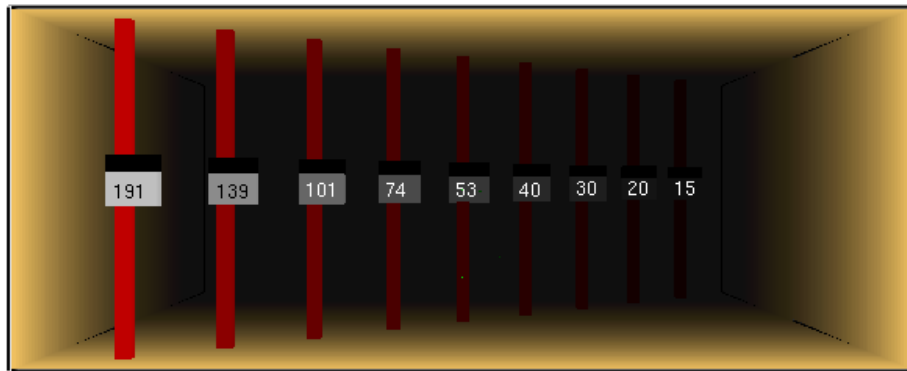
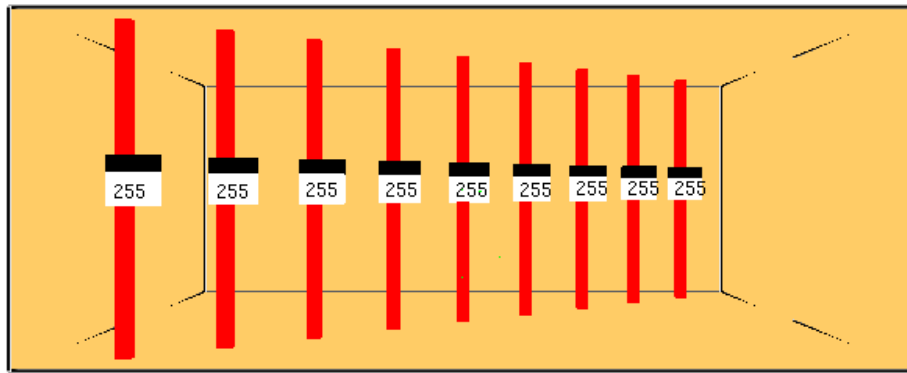


Figure 18.5: Smokeview rendering of a compartment filled with no smoke (top), smoke (center), and a comparison of the Smokeview pixel values with the expected values.

18.3.7 Aspiration Detection Systems

An aspiration detection system groups together a series of smoke measurement devices. An aspiration system consists of a sampling pipe network that draws air from a series of locations to a central point where an obscuration measurement is made. To define such a system in FDS, you must provide the sampling locations, sampling flow rates, the transport time from each sampling location, and if an alarm output is desired, the overall obscuration “setpoint.” One or more `DEVC` inputs are used to specify details of the sampling locations, and one additional input is used to specify the central detector:

```
&DEVC XYZ=..., QUANTITY='DENSITY', SPEC_ID='SOOT', ID='soot1', DEVC_ID='asp1',
      FLOWRATE=0.1, DELAY=20 /
&DEVC XYZ=..., QUANTITY='DENSITY', SPEC_ID='SOOT', ID='soot2', DEVC_ID='asp1',
      FLOWRATE=0.2, DELAY=10 /
...
&DEVC XYZ=..., QUANTITY='DENSITY', SPEC_ID='SOOT', ID='sootN', DEVC_ID='asp1',
      FLOWRATE=0.3, DELAY=30 /
&DEVC XYZ=..., QUANTITY='ASPIRATION', ID='asp1', BYPASS_FLOWRATE=0.4,
      SETPOINT=0.02 /
```

where the `DEVC_ID` is used at each sampling point to reference the central detector, `FLOWRATE` is the gas flow rate in kg/s, `DELAY` is the transport time (in seconds) from the sampling location to the central detector, `BYPASS_FLOWRATE` is the flow rate in kg/s of any air drawn into the system from outside the computational domain (accounts for portions of the sampling network lying outside the domain defined by the `MESH` inputs), and `SETPOINT` is the alarm threshold obscuration in units of %/m. The output of the aspiration system is computed as

$$\text{Obscuration} = \left(1 - \exp \left(-K_m \frac{\sum_{i=1}^N \rho_{s,i}(t - t_{d,i}) \dot{m}_i}{\sum_{i=1}^N \dot{m}_i} \right) \right) \times 100 \text{ \%}/\text{m} \quad (18.7)$$

where \dot{m}_i is the mass `FLOWRATE` at sampling location i , $\rho_{s,i}(t - t_{d,i})$ is the soot density at sampling location i , $t_{d,i}$ s prior (`DELAY`) to the current time t , and K_m is the `MASS_EXTINCTION_COEFFICIENT` associated with visible light.

Note that FDS doesn’t actually remove gas from the computational domain based upon the `FLOWRATE`; the `FLOWRATE` is just a weighting factor.

Example Case: aspiration_detector

A cubical compartment, 2 m on a side has a three sampling location aspiration system. The three locations have equal flow rates of 0.3 kg/s, and transport times of 50, 100, and 150 s, respectively. No bypass flow rate is specified for the aspiration detector. Combustion products are forced into the bottom of the compartment at a rate of 1 kg/s. The `SOOT_YIELD`=0.001. Mass is removed from the top of the compartment at a rate of 1 kg/s. The aspiration detector shows an increasing obscuration over time. There is a delay of slightly over 50 s in the initial increase which results from the 50 s transport time for the first sampling location plus a short period of time to transport the combustion products to the sampling location. The detector response has three plateaus that result from the delay times of the sampling locations. The sampling points are co-located, so each plateau represents an additional one third of the soot being transported to the detector. The soot density at the sampling point is 7.1×10^{-5} kg/m³. Using this value the plateaus are computed as 18 %, 33.2 %, and 45.7 %, as seen in Fig. 18.6.

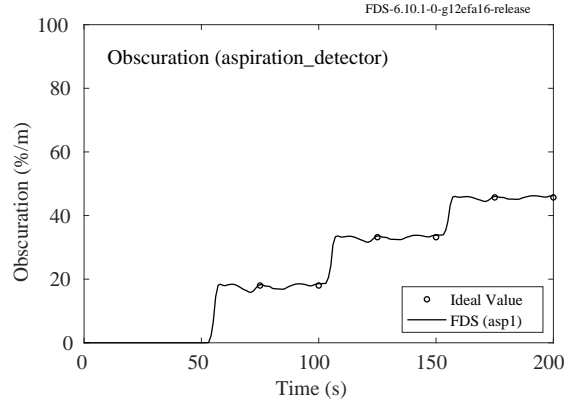


Figure 18.6: Output of `aspiration_detector` test case.

18.3.8 Fire Depth

In some cases it may be useful to output a characteristic depth of the fire along a certain path. For example, this parameter is important in the case of simulating spreading fires such as wildland fires. There a number of ways in which the depth of a fire might be defined, but a simple choice is to use logic similar to beam detectors and to calculate the length of the beam which traverses regions with a volumetric heat release greater than a threshold value. This quantity of 'FIRE DEPTH' can be written as:

$$\text{Fire Depth} = \sum_{i=1}^N C_i \Delta x_i ; C_i = \begin{cases} 0 & \dot{q}_i''' \leq \dot{q}_{\text{thresh}}''' \\ 1 & \dot{q}_i''' > \dot{q}_{\text{thresh}}''' \end{cases} \quad (18.8)$$

As with the beam detectors, i is a mesh cell along the path of the beam and Δx_i is the distance within the mesh cell that is traversed by the beam. The threshold value, $\dot{q}_{\text{thresh}}'''$, is applied by setting `SETPOINT` on the `DEVC` line in units of kW/m^3 . If this value is not provided, the default is the threshold for rendering `HRRPUV` as described in Section 22.8.

18.4 Basic Control Logic

Devices can be used to control various actions, like creating and removing obstructions, or activating and deactivating fans and vents. Every device has an associated `QUANTITY`, whether it is included directly on the `DEVC` line or indirectly on the optional `PROP` line. Using the `DEVC` parameter `SETPOINT`, you can trigger an action to occur when the `QUANTITY` value passes above, or below, the given `SETPOINT`. The following parameters dictate how a device will control something:

- **SETPOINT** The value of the device at which its state changes. For a detection type of device (e.g., heat or smoke) this value is taken from the device's `PROP` inputs and need not be specified on the `DEVC` line.
- **TRIP_DIRECTION** A positive integer means the device will change from its `INITIAL_STATE` when the value of the device is greater than the `SETPOINT` and be equal to the `INITIAL_STATE` when the value is less than the `SETPOINT`. A negative integer has the opposite behavior. The device will change from its `INITIAL_STATE` when the value of the device is less than the `SETPOINT` and be equal to the `INITIAL_STATE` when the value is greater than the `SETPOINT`. The default value is +1.
- **LATCH** If this logical value is set to `T` the device will only change state once. The default value is `F`.

- **INITIAL_STATE** This logical value is the initial state of the device. The default value is **F**. For example, if an obstruction associated with the device is to disappear, set **INITIAL_STATE=T**.

If you desire to control FDS using more complex logic than can be provided by the use of a single device and its setpoint, control functions can be specified using the **CTRL** input. See Section 18.5 for more on **CTRL** functions. The simplest example of a device is just a timer:

```
&DEVC XYZ=1.2,3.4,5.6, ID='my clock', QUANTITY='TIME', SETPOINT=30. /
```

Anything associated with the device via the parameter, **DEVC_ID='my clock'**, will change its state at 30 s. For example, if the text were added to an **OBST** line, that obstruction would change from its **INITIAL_STATE** of **F** to **T** after 30 s. In other words, it would be created at 30 s instead of at the start of the simulation. This is a simple way to open a door or window.

When using a **DEVC** output to control FDS, the instantaneous value of the **DEVC** is used. For some **QUANTITY** types, such as **TEMPERATURE**, this output can be very noisy. To prevent a spurious spike from causing a state change of the **DEVC** you can specify the parameter **SMOOTHING_FACTOR**. This is a parameter that can vary between 0 and 1. It performs an exponential smoothing of the **DEVC** output as follows:

$$\bar{x}^n = \bar{x}^{n-1} \text{ SMOOTHING_FACTOR} + x^n (1 - \text{SMOOTHING_FACTOR}) \quad (18.9)$$

where n is the time step, x is the instantaneous device output and \bar{x} is the smoothed output. The **SMOOTHING_FACTOR** defaults to 0 which means no smoothing is performed. Note that **SMOOTHING_FACTOR** only changes the value passed to control functions; it has no effect on the value of the **DEVC** written to the **CHID_devc.csv** file.

An alternative to **SMOOTHING_FACTOR** is to instead use a **SMOOTHING_TIME** (units of seconds [s]), which may provide a more intuitive time scale over which the function is to be smoothed. The two parameters are mathematically related by,

$$\text{SMOOTHING_FACTOR} = \left(1 - \frac{\delta t}{\text{SMOOTHING_TIME}} \right) \quad (18.10)$$

where δt is the local time step in the simulation. With a constant **SMOOTHING_TIME** used on the **DEVC** line, the **SMOOTHING_FACTOR** dynamically adjusts with the local time step.

Note that each state change of a device is recorded to a log file, see Sec. 27.5.

18.4.1 Creating and Removing Obstructions

In many fire scenarios, the opening or closing of a door or window can lead to dramatic changes in the course of the fire. Sometimes these actions are taken intentionally, sometimes as a result of the fire. Within the framework of an FDS calculation, these actions are represented by the creation or removal of solid obstacles, or the opening or closing of exterior vents.

Remove or create a solid obstruction by assigning the character string **DEVC_ID** to indicate the name of a **DEVC ID** on the **OBST** line that is to be created or removed. This will direct FDS to remove or create the obstruction when the device changes state to **F** or **T**, respectively. For example, the lines

```
&OBST XB=..., DEVC_ID='det2' /
&DEVC XYZ=..., ID='det2', INITIAL_STATE=T /
```

will cause the given obstruction to be removed when the specified **DEVC** changes state.

Creation or removal at a predetermined time can be performed using a **DEVC** that has **TIME** as its measured quantity. For example, the following instructions will cause the specified **HOLES** and **OBSTRUCTIONS** to

appear/disappear at the various designated times. These lines are part of the simple test case called `create_remove.fds`.

```
&OBST XB=0.3,0.4,0.1,0.9,0.1,0.9, COLOR='PURPLE' /
&HOLE XB=0.2,0.4,0.2,0.3,0.2,0.3, COLOR='RED', DEVC_ID='timer1' /
&HOLE XB=0.2,0.4,0.7,0.8,0.7,0.8, COLOR='GREEN', DEVC_ID='timer2' /
&OBST XB=0.7,0.8,0.2,0.3,0.2,0.3, COLOR='BLUE', DEVC_ID='timer3' /
&OBST XB=0.7,0.8,0.6,0.7,0.6,0.7, COLOR='PINK', DEVC_ID='timer4' /
&OBST XB=0.5,1.0,0.0,1.0,0.0,0.1, COLOR='YELLOW', DEVC_ID='timer5' /
&HOLE XB=0.7,0.8,0.7,0.8,0.0,0.1, COLOR='BLACK', DEVC_ID='timer6' /
&HOLE XB=0.7,0.8,0.2,0.3,0.0,0.1, COLOR='GRAY 50', DEVC_ID='timer7' /

&DEVC XYZ=..., ID='timer1', SETPOINT=1., QUANTITY='TIME', INITIAL_STATE=F /
&DEVC XYZ=..., ID='timer2', SETPOINT=2., QUANTITY='TIME', INITIAL_STATE=T /
&DEVC XYZ=..., ID='timer3', SETPOINT=3., QUANTITY='TIME', INITIAL_STATE=F /
&DEVC XYZ=..., ID='timer4', SETPOINT=4., QUANTITY='TIME', INITIAL_STATE=T /
&DEVC XYZ=..., ID='timer5', SETPOINT=5., QUANTITY='TIME', INITIAL_STATE=F /
&DEVC XYZ=..., ID='timer6', SETPOINT=6., QUANTITY='TIME', INITIAL_STATE=T /
&DEVC XYZ=..., ID='timer7', SETPOINT=6., QUANTITY='TIME', INITIAL_STATE=F /
```

At the start of the simulation, the purple obstruction is present with a red block embedded in it. This red block is actually a `HOLE` whose initial state is `F`, i.e., the hole is filled. Also at the start of the simulation, there is a pink obstruction that is visible. At 1 s the red block disappears. At 2 s the empty hole in the purple obstruction is filled with a green block. This hole was initially true, i.e. empty. The blue obstruction appears at 3 s because its initial state is false, meaning that it does not exist initially. The pink obstruction disappears at 4 s because its initial state is true and this state changes at 4 s. At 5 s a yellow obstruction appears with one empty hole and one embedded gray block. At 6 s the gray block disappears because it is a hole that was initially false and therefore was filled with the gray block when its parent obstruction (yellow) was created. Also at 6 s the hole originally present in the yellow obstruction is filled with a black block because it was a hole that was initially empty and then filled when its `DEVC` changed state. *You should always try a simple example first before embarking on a complicated creation/removal scheme for obstructions and holes.*

To create and remove obstructions multiple times, either set `LATCH=F` together with a cycling math function (see Sec. 18.5.1), as shown below, or use the '`CUSTOM`' control feature (see Sec. 18.5.5). The following uses `TIME` as the input to a `SIN` math function . The sign of the `DEVC` value toggles the state of the `OBST`.

```
&DEVC ID='t1', XYZ=..., QUANTITY='TIME', TIME_AVERAGED=F/
&CTRL ID='s1', FUNCTION_TYPE='SIN', INPUT_ID='t1'/
&DEVC ID='timer', XYZ=..., QUANTITY='CONTROL VALUE', CTRL_ID='s1', SETPOINT=0,
    LATCH=F/
&OBST XB=..., DEVC_ID='timer'/
```

18.4.2 Activating and Deactivating Vents

When a device or control function is applied to a `VENT`, the purpose is to either activate or deactivate any time ramp associated with the `VENT` via its `DEVC_ID`. For example, to control a fan, do the following:

```
&SURF ID='FAN', VOLUME_FLOW=5. /
&VENT XB=..., SURF_ID='FAN', DEVC_ID='det2' /
&DEVC ID='det2', XYZ=..., QUANTITY='TIME', SETPOINT=30., INITIAL_STATE=F /
```

Note that at 30 s, the “state” of the '`FAN`' changes from `F` to `T`, or more simply, the '`FAN`' turns on. Since

there is no explicit time function associated with the 'FAN', the default 1 s ramp-up will begin at 30 s instead of at 0 s. If `INITIAL_STATE=T`, then the fan should turn off at 30 s. Essentially, “activation” of a `VENT` causes all associated time functions to be delayed until the device `SETPOINT` is reached. “Deactivation” of a `VENT` turns off all time functions. Usually this means that the parameters on the `SURF` line are all nullified, so it is a good idea to check the functionality with a simple example.

A 'MIRROR' or 'OPEN' `VENT` should not be activated or deactivated. You can, however, place an obstruction in front of an 'OPEN' `VENT` and then create it or remove it to model the closing or opening of a door or window.

There are sometimes circumstances when you might want to create or remove obstructions that have `VENTS` attached. If the obstructions overlap, there can be confusion as to which `VENT` goes with which `OBST`. If you come across a situation like this, give the `OBST` an `ID` and assign this `OBST_ID` on the `VENT` line.

18.5 Advanced Control Functions: The `CTRL` Namelist Group

There are many systems whose functionality cannot be described by a simple device with a single “setpoint.” Consider for example, a typical HVAC system used for heat. It is controlled by a thermostat that is given a temperature setpoint. The system turns on when the temperature goes below the setpoint by some amount and then turns off when the temperature rises above that same setpoint by some amount. This behavior cannot be defined by merely specifying a single setpoint. You must also define the range or “deadband” around the setpoint, and whether an increasing or decreasing temperature activates the system. For the HVAC example, crossing the lower edge of the deadband activates heating; crossing the upper edge deactivates heating. These more complicated behaviors can be modeled in FDS using `CTRLS`. The following parameters dictate how a control function will behave:

- `ID` A name for the control function that is unique over all control functions.
- `FUNCTION_TYPE` The type of control function. The possible types are shown in Table 18.2.
- `INPUT_ID` A list of `DEVC` or `CTRL` `IDS` that are the inputs to the control function. Up to forty inputs can be specified. If a `DEVC` or `CTRL` is being used as an `INPUT_ID` for a control function, then it must have a unique `ID` over both devices and control functions. Additionally, a control function cannot be used as an input for itself.
- `SETPOINT` The value of the control function at which its state changes. This is only appropriate for functions that return numerical values.
- `TRIP_DIRECTION` A positive integer means the control function will change state when its value increases past the setpoint and a negative integer means the control function will change state when its value decreases past the setpoint. The default value is +1.
- `LATCH` If this logical value is set to `T` the control function will only change state once. The default is `LATCH=T`.
- `INITIAL_STATE` The initial state of the control function. Default `F`. For example, if an obstruction associated with the control function is to disappear, set `INITIAL_STATE=T` on the `DEVC` line.

For any object for which a `DEVC_ID` can be specified (such as `OBST` or `VENT`), a `CTRL_ID` can be specified instead.

If you want to design a system of controls and devices that involves multiple changes of state, include the attribute `LATCH=F` on the relevant `DEVC` or `CTRL` lines. By default, devices and controls may only change

Table 18.2: Control function types.

FUNCTION_TYPE	Purpose
ANY	Changes state if <u>any</u> INPUTS are T
ALL	Changes state if <u>all</u> INPUTS are T
ONLY	Changes state if and <u>only</u> if N INPUTS are T
AT_LEAST	Changes state if <u>at least</u> N INPUTS are T
TIME_DELAY	Changes state <u>DELAY</u> s after INPUT becomes T
CUSTOM	Changes state based on evaluating a RAMP of the function's input
DEADBAND	Behaves like a thermostat
KILL	Terminates code execution if INPUT is T
RESTART	Dumps restart files if INPUT is T
SUM	Sums the outputs of the INPUTS
SUBTRACT	Subtracts the second INPUT from the first
MULTIPLY	Multiplies the INPUTS
DIVIDE	Divides the first INPUT by the second
POWER	The first INPUT to the power of the second
EXP	The exponential of the INPUT
LOG	The natural logarithm of the INPUT
COS	The cosine of the INPUT
SIN	The sine of the INPUT
ACOS	The arccosine of the INPUT
ASIN	The arcsine of the INPUT
ATAN	The arctangent of the INPUT
MAX	Maximum value of the INPUTS
MIN	Minimum value of the INPUTS
PID	A Proportional-Integral-Derivative control for the INPUT
PERCENTILE	Calculate the user-specified percentile for a function
EXTERNAL	External file control, see Section 18.8

state once, like a sprinkler activating or smoke detector alarming. LATCH=T by default for both devices and controls.

If you want a DEVC to operate based on the logical state of a CTRL, set QUANTITY='CONTROL' and set the CTRL_ID to the ID of the control function. The numerical output in the CHID_devc.csv file will be zero when the logical state is F and 1 when the logical state is T.

The output value of a numerical control function is defined by a DEVC line with QUANTITY='CONTROL VALUE' and CTRL_ID set equal to the ID of the control function. You can then use SETPOINT to have the DEVC operate a particular output value of the control function. This output cannot be used for a purely logical control function whose only output is a logical state such as ANY or DEADBAND. For CUSTOM the output will be the value of the RAMP function, and for TIME_DELAY the value will be the evaluated delay.

The outputs QUANTITY='CONTROL' and QUANTITY='CONTROL VALUE' have the default of TEMPORAL_STATISTIC='INSTANT VALUE'.

The time dependent logical state of each control function is written to the CHID_ctrl.csv log file, see Sec. 27.4. Each state change of a control function or device with a SETPOINT is recorded to the CHID_devc_

ctrl_log.csv log file, see Sec. 27.5.

18.5.1 Control Functions: ANY, ALL, ONLY, and AT_LEAST

Suppose you want an obstruction to be removed (a door is opened, for example) after any of four smoke detectors in a room has activated. Use input lines of the form:

```
&OBST XB=..., SURF_ID='...', CTRL_ID='SD' /

&DEVC XYZ=1,1,3, PROP_ID='Acme Smoker', ID='SD_1' /
&DEVC XYZ=1,4,3, PROP_ID='Acme Smoker', ID='SD_2' /
&DEVC XYZ=4,1,3, PROP_ID='Acme Smoker', ID='SD_3' /
&DEVC XYZ=4,4,3, PROP_ID='Acme Smoker', ID='SD_4' /
&CTRL ID='SD', FUNCTION_TYPE='ANY', INPUT_ID='SD_1','SD_2','SD_3','SD_4',
      INITIAL_STATE=T /
```

The `INITIAL_STATE` of the control function `SD` is `T`, meaning that the obstruction exists initially. The “change of state” means that the obstruction is removed when any smoke detector alarms. By default, the `INITIAL_STATE` of the control function `SD` is `F`, meaning that the obstruction does not exist initially.

Suppose that now you want the obstruction to be created (a door is closed, for example) after all four smoke detectors in a room have activated. Use a control line of the form:

```
&CTRL ID='SD', FUNCTION_TYPE='ALL', INPUT_ID='SD_1','SD_2','SD_3','SD_4' /
```

The control functions `AT_LEAST` and `ONLY` are generalizations of `ANY` and `ALL`. For example,

```
&CTRL ID='SD', FUNCTION_TYPE='AT_LEAST', N=3, INPUT_ID='SD_1','SD_2','SD_3','SD_4' /
```

changes the state from `F` to `T` when at least 3 detectors activate. Note that in this example, and the example below, the parameter `N` is used to specify the number of activated devices required for the conditions of the control function to be satisfied. The control function,

```
&CTRL ID='SD', FUNCTION_TYPE='ONLY', N=3, INPUT_ID='SD_1','SD_2','SD_3','SD_4' /
```

changes the state from `F` to `T` when 3, and only 3, detectors activate.

18.5.2 Control Function: TIME_DELAY

The `TIME_DELAY` control function starts a timer of length `DELAY` when its input changes state. When the timer expires, the `TIME_DELAY` control function will change state. Note, that the timer starts at each change in state of the input; therefore, if the input changes state a second time before the first timer ends, the timer will get reset. This function enables FDS to model time delays between when a device activates and when some other action occurs, like in a dry pipe sprinkler system.

```
&DEVC XYZ=2,2,3, PROP_ID='Acme Sprinkler_link', QUANTITY='LINK TEMPERATURE',
      ID='Spk_29_link' /
&DEVC XYZ=2,2,3, PROP_ID='Acme Sprinkler', QUANTITY='CONTROL', ID='Spk_29',
      CTRL_ID='dry pipe' /
&CTRL ID='dry pipe', FUNCTION_TYPE='TIME_DELAY', INPUT_ID='Spk_29_link', DELAY=30. /
```

This relationship between a sprinkler and its pipes means that the sprinkler spray is controlled (in this case delayed) by the 'dry pipe', which adds 30 s to the activation time of `Spk_29`, measured by `Spk_29_link`,

before water can flow out of the head.

18.5.3 Control Function: DEADBAND

This control function behaves like an HVAC thermostat. It can operate in one of two modes analogous to heating or cooling. The function is provided with an `INPUT_ID` which is the `DEVC` whose value is used by the function, an upper and lower `SETPOINT`, and the mode of operation by `ON_BOUND`. If `ON_BOUND='LOWER'`, the function changes state from its `INITIAL_STATE` when the value of the `INPUT_ID` drops below the lower value in `SETPOINT` and reverts when it increases past the upper value, i.e., like a heating system. The reverse will occur if `ON_BOUND='UPPER'`, i.e., a cooling system.

For an HVAC system, the following lines of input would set up a simple thermostat:

```
&SURF ID='FAN', TMP_FRONT=40., VOLUME_FLOW=-1. /
&VENT XB=-0.3,0.3,-0.3,0.3,0.0,0.0, SURF_ID='FAN', CTRL_ID='thermostat' /
&DEVC ID='TC', XYZ=2.4,5.7,3.6, QUANTITY='TEMPERATURE' /
&CTRL ID='thermostat', FUNCTION_TYPE='DEADBAND', INPUT_ID='TC',
      ON_BOUND='LOWER', SETPOINT=23.,27., LATCH=F/
```

Here, we want to control the `VENT` that simulates the `FAN`, which blows hot air into the room. A `DEVC` called `TC` is positioned in the room to measure the `TEMPERATURE`. The `thermostat` uses a `SETPOINT` to turn on the `FAN` when the temperature falls below 23 °C (`ON_BOUND='LOWER'`) and it turns off when the temperature rises above 27 °C.

Note that a deadband controller needs to have `LATCH` set to `F`

18.5.4 Control Function: RESTART and KILL

There are times when you might only want to run a simulation until some goal is reached, or you might want to create some baseline condition and then run multiple permutations of that baseline. For example, you might want to run a series of simulations where different mitigation strategies are tested once a detector alarms. Using the `RESTART` control function, you can cause a restart file to be created once a desired condition is met. The simulation can continue and the restart files can be copied to have the job identifying string, `CHID`, of the various permutations (providing of course that the usual restrictions on the use of restart files are followed). Using the `KILL` control function stops FDS once a desired condition is met. If you want to both write restart files and stop FDS, then you would need to use both functions with the same `INPUT_ID`. For example, the lines

```
&DEVC ID='temp', QUANTITY='TEMPERATURE', SETPOINT=1000., XYZ=4.5,6.7,3.6 /
&DEVC ID='velo', QUANTITY='VELOCITY', SETPOINT=10., XYZ=4.5,6.7,3.6 /

&CTRL ID='kill', FUNCTION_TYPE='KILL', INPUT_ID='temp' /
&CTRL ID='restart', FUNCTION_TYPE='RESTART', INPUT_ID='velo' /
```

will kill the job when the temperature at the given point rises above 1000 °C; or just force restart files to be output when the velocity at a given point exceeds 10 m/s. The lines

```
&DEVC ID='temp', QUANTITY='TEMPERATURE', SETPOINT=1000., XYZ=4.5,6.7,3.6 /

&CTRL ID='kill', FUNCTION_TYPE='KILL', INPUT_ID='temp' /
&CTRL ID='restart', FUNCTION_TYPE='RESTART', INPUT_ID='temp' /
```

will kill the job when the temperature at the given point rises above 1000 °C and also write restart files.

18.5.5 Control Function: CUSTOM

For most of the control function types, the logical (true/false) output of the devices and control functions and the time they last changed state are taken as inputs. A `CUSTOM` function uses the numerical output of a `DEVC` along with a `RAMP` to determine the output of the function. When the `RAMP` output for the `DEVC` value is negative, the `CTRL` will have the value of its `INITIAL_STATE`. When the `RAMP` output for the `DEVC` value is positive, the `CTRL` will have the opposite value of its `INITIAL_STATE`. In the case below, the `CUSTOM` control function uses the numerical output of a timer device as its input. The function returns true (the default value for `INITIAL_STATE` is `F`) when the `F` parameter in the ramp specified with `RAMP_ID` is a positive value and false when the `RAMP F` value is negative. In this case, the control would start false and would switch to true when the timer reaches 60 s. It would then stay in a true state until the timer reaches 120 s and would then change back to false.

Note that when using control functions the `IDS` assigned to both the `CTRL` and the `DEVC` inputs must be unique across both sets of inputs, i.e., you cannot use the same `ID` for both a control function and a device. You can make a fan operate on a fixed cycle by using a `CUSTOM` control function based on time:

```
&SURF ID='FAN', TMP_FRONT=40., VOLUME_FLOW=-1. /
&VENT XB=-0.3,0.3,-0.3,0.3,0.0,0.0, SURF_ID='FAN', CTRL_ID='cycling timer' /
&DEVC ID='TIMER', XYZ=2.4,5.7,3.6, QUANTITY='TIME' /
&CTRL ID='cycling timer', FUNCTION_TYPE='CUSTOM', INPUT_ID='TIMER', RAMP_ID='cycle' /
&RAMP ID='cycle', T= 59, F=-1 /
&RAMP ID='cycle', T= 61, F= 1 /
&RAMP ID='cycle', T=119, F= 1 /
&RAMP ID='cycle', T=121, F=-1 /
```

In the above example the fan will be off initially, turn on at 60 s and then turn off at 120 s.

You can make an obstruction appear and disappear multiple times by using the following lines

```
&OBST XB=..., SURF_ID='whatever', CTRL_ID='cycling timer' /
&DEVC ID='TIMER', XYZ=..., QUANTITY='TIME' /
&CTRL ID='cycling timer', FUNCTION_TYPE='CUSTOM', INPUT_ID='TIMER', RAMP_ID='cycle' /
&RAMP ID='cycle', T= 0, F=-1 /
&RAMP ID='cycle', T= 59, F=-1 /
&RAMP ID='cycle', T= 61, F= 1 /
&RAMP ID='cycle', T=119, F= 1 /
&RAMP ID='cycle', T=121, F=-1 /
```

The above will have the obstacle initially removed, then added at 60 s, and removed again at 120 s.

Experiment with these combinations using a simple case before trying a case to make sure that FDS indeed is doing what is intended.

18.5.6 Control Function: Math Operations

The control functions that perform simple math operations (`SUM`, `SUBTRACT`, `MULTIPLY`, `DIVIDE`, `POWER`, etc.) can have a constant value specified as one of their inputs. This is done by specifying one of the `INPUT_IDS` as `'CONSTANT'` and providing the value using the input `CONSTANT`. For example, the inputs below represent a control function whose state changes when the square of the velocity exceeds 10 (see Sec. 18.4 for an explanation of `TRIP_DIRECTION`).

```
&DEVC ID='SPEED SENSOR', XYZ=..., QUANTITY='VELOCITY' /
&CTRL ID='multiplier', FUNCTION_TYPE='POWER',
      INPUT_ID='SPEED SENSOR','CONSTANT', CONSTANT=2., SETPOINT=10.,
```


TRIP_DIRECTION=1 /

18.5.7 Control Function: PID Control Function

A PID (Proportional Integral Derivative) control function is a commonly used feedback controller for controlling electrical and mechanical systems. The function computes an error between a process variable and a desired setpoint. The goal of the PID function is to minimize the error. A PID control function is computed as

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{de(t)}{dt} \quad (18.11)$$

where K_p , K_i , and K_d are respectively the `PROPORTIONAL_GAIN`, the `INTEGRAL_GAIN`, and the `DIFFERENTIAL_GAIN`; $e(t)$ is the error given by subtracting the `TARGET_VALUE` from the `INPUT_ID`; and $u(t)$ is the output.

Example Case: using PID for time integration of mass loss rate

The PID controller can be used to time integrate the mass loss from a pyrolyzing surface. First, use a `DEVC` with `SPATIAL_STATISTIC='SURFACE INTEGRAL'` as input to the PID controller. Omit the `TARGET_VALUE`, which then defaults to 0, and the input becomes the error function, $e(t)$. Set the `INTEGRAL_GAIN` to 1 and the other gains to 0. Then send the PID output to a `DEVC` for a 'CONTROL VALUE'. Here is example syntax:

```
&DEVC ID='MY MLR', XB=..., QUANTITY='BURNING RATE', SURF_ID='s1',
      SPATIAL_STATISTIC='SURFACE INTEGRAL'/
&CTRL ID='MY PID', FUNCTION_TYPE='PID', INPUT_ID='MY MLR',
      PROPORTIONAL_GAIN=0, DIFFERENTIAL_GAIN=0, INTEGRAL_GAIN=1/
&DEVC ID='PYROLYZATE', XYZ=..., QUANTITY='CONTROL VALUE', CTRL_ID='MY PID'/
```

Here, the 'PYROLYZATE' column of the `_devc.csv` file will have units of kg.

Special Case: using PID for controlling momentum

The PID controller can be used to adjust a mean momentum source term to achieve various output targets related to momentum. One simple example would be to control the bulk flow rate in a channel. In the following, we monitor the mean stream-wise velocity and set `CONTROL_FORCE(1)=T` to add a source term to the x momentum equation (use 2 and 3 for y and z , respectively). The result is similar to manually adjusting `FORCE_VECTOR(1)` to achieve the desired flow rate. In the case below, the PID controller will keep the mean bulk flow near the `TARGET_VALUE` of 10 m/s.

```
&DEVC ID='Ub', XB=..., QUANTITY='U-VELOCITY', SPATIAL_STATISTIC='MEAN'/
&CTRL ID='MY PID', FUNCTION_TYPE='PID', TARGET_VALUE=10, INPUT_ID='Ub', PROPORTIONAL_
      GAIN=0.1, CONTROL_FORCE(1)=T/
&DEVC ID='FORCE X', XYZ=..., QUANTITY='CONTROL VALUE', CTRL_ID='MY PID'/
```

18.5.8 Control Function: PERCENTILE

Consider the cumulative distribution function, $F(x)$:

$$F(x) = \int_{-\infty}^x f(x') dx' / \int_{-\infty}^{\infty} f(x') dx' \quad (18.12)$$

The `PERCENTILE` control function returns the value of x for which $F(x) = p$, where p is a value between 0 and 1. In discretized form, the function $f(x)$ is represented by the pairs (x_i, f_i) for $1 \leq i \leq N$. The p th percentile is then given by

$$x(p) = \bar{x}_{k-1} + (\bar{x}_k - \bar{x}_{k-1}) \frac{pF_N - F_{k-1}}{F_k - F_{k-1}} \quad (18.13)$$

where

$$F_k = \sum_{i=1}^k f_i \delta \bar{x}_i \quad ; \quad \bar{x}_i = \frac{x_{i+1} + x_i}{2} \quad ; \quad \delta \bar{x}_i = \bar{x}_i - \bar{x}_{i-1} \quad ; \quad k = \min_n (F_n > pF_N) \quad (18.14)$$

It is assumed that the values x_i are mid-points of the discretized function; that is, f_i is the average value of the function over the interval $(\bar{x}_{i-1}, \bar{x}_i)$.

The `PERCENTILE` function is useful for computing flame heights. Consider the following input lines:

```
&DEVC XB=0,0,0,0,0.05,4.95, QUANTITY='HRRPUV', SPATIAL_STATISTIC='VOLUME INTEGRAL'
      DX=1., DY=1., DZ=0.05, POINTS=50, Z_ID='z', ID='qdot' /
&CTRL ID='pct', FUNCTION_TYPE='PERCENTILE', INPUT_ID='qdot', PERCENTILE=0.95 /
&DEVC ID='L_F', XYZ=0,0,0, QUANTITY='CONTROL VALUE', CTRL_ID='pct', UNITS='m' /
```

The first `DEVC` line represents a vertical profile of the heat release rate per unit volume integrated over horizontal slices of dimension $2*DX$, $2*DY$, and $2*DZ$. The `CTRL` function takes these 50 values at 50 uniformly spaced heights between 0.05 m and 4.95 m and calculates the height at which 95 % of the fire's energy has been released. Note that for a 10 cm grid, the vertical array of points are located at cell centers, which is why the discretized integration is done the way it is described above. The second `DEVC` line simply takes the value calculated by the control function and prints it out in the file of time histories, `CHID_devcc.csv`. The 50 integrals of 'HRRPUV' are time-averaged and written to the file, `CHID_line.csv`. Note that the flame heights written to `CHID_devcc.csv` are time-averaged over the time interval between printouts. If you set `DT_DEVC` to a very small value (i.e. less than the time step), you will obtain a time-history of instantaneous flame heights.

18.5.9 Combining Control Functions: A Deluge System

For a deluge sprinkler system, the normally dry sprinkler pipes are flooded when a detection event occurs. For this example, the detection event is when two of four smoke detectors alarm. It takes 30 s to flood the piping network. The nozzle is a `DEVC` named 'NOZZLE 1' controlled by the `CTRL` named 'nozzle trigger'. The nozzle activates when both detection and the time delay have occurred. Note that the `DEVC` is specified with `QUANTITY='CONTROL'`.

```
&DEVC XYZ=1,1,3, PROP_ID='Acme Smoker', ID='SD_1' /
&DEVC XYZ=1,4,3, PROP_ID='Acme Smoker', ID='SD_2' /
&DEVC XYZ=4,1,3, PROP_ID='Acme Smoker', ID='SD_3' /
&DEVC XYZ=4,4,3, PROP_ID='Acme Smoker', ID='SD_4' /
&DEVC XYZ=2,2,3, PROP_ID='Acme Nozzle', QUANTITY='CONTROL',
      ID='NOZZLE 1', CTRL_ID='nozzle trigger' /

&CTRL ID='nozzle trigger', FUNCTION_TYPE='ALL', INPUT_ID='smokey','delay' /
&CTRL ID='delay', FUNCTION_TYPE='TIME_DELAY', INPUT_ID='smokey', DELAY=30. /
&CTRL ID='smokey', FUNCTION_TYPE='AT_LEAST', N=2, INPUT_ID='SD_1','SD_2','SD_3','SD_4' /
```

Example Case: control_test_2

The `control_test_2` example demonstrates the use of the mathematical and PID control functions. Two compartments are defined with the left hand compartment initialized to 20 °C and the right hand compartment to 10 °C. Control functions are defined to:

- Add the temperatures in the two compartments
- Subtract the right hand compartment temperature from the left hand compartment temperature
- Multiply the left hand temperature by 0.5
- Divide the left hand temperature by the right hand temperature
- Take the square root of the right hand temperature
- Use the time as input to a PID function with a target value of 5 and $K_p=-0.5$, $K_i=0.001$, and $K_d=1$

```
&CTRL ID='Add',FUNCTION_TYPE='SUM',INPUT_ID='LHS Temp','RHS Temp'/
&CTRL ID='Subtract',FUNCTION_TYPE='SUBTRACT',INPUT_ID='RHS Temp','LHS Temp'/
&CTRL ID='Multiply',FUNCTION_TYPE='MULTIPLY',INPUT_ID='LHS
Temp','CONSTANT',CONSTANT=0.5/
&CTRL ID='Divide',FUNCTION_TYPE='DIVIDE',INPUT_ID='LHS Temp','RHS Temp'/
&CTRL ID='Power',FUNCTION_TYPE='POWER',INPUT_ID='RHS Temp','CONSTANT',CONSTANT=0.5/
&CTRL ID='PID',FUNCTION_TYPE='PID',INPUT_ID='Time',TARGET_VALUE=5.,
PROPORTIONAL_GAIN=-0.5,INTEGRAL_GAIN=0.001,DIFFERENTIAL_GAIN=1./
```

Results are shown in Fig. 18.7.

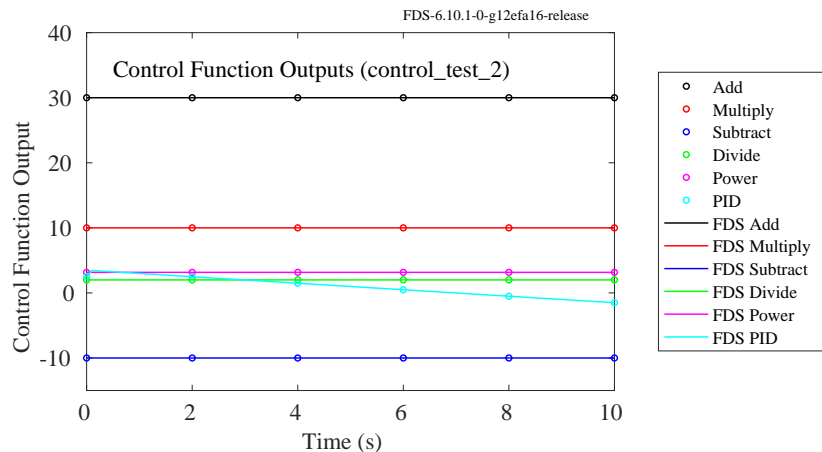


Figure 18.7: Results of the `control_test_2` case.

18.5.10 Combining Control Functions: A Dry Pipe Sprinkler System

For a dry-pipe sprinkler system, the normally dry sprinkler pipes are pressurized with gas. When a link activates in a sprinkler head, the pressure drop allows water to flow into the pipe network. For this example it takes 30 s to flood the piping network once a sprinkler link has activated. The sequence of events required for operation is first ANY of the links must activate which starts the 30 s `TIME_DELAY`. Once the 30 s delay has occurred, each nozzle with an active link, the ALL control functions, will then flow water.

```

&DEVC XYZ=2,2,3, PROP_ID='Acme Sprinkler Link', ID='LINK 1' /
&DEVC XYZ=2,3,3, PROP_ID='Acme Sprinkler Link', ID='LINK 2' /

&PROP ID='Acme Sprinkler Link', QUANTITY='LINK TEMPERATURE',
ACTIVATION_TEMPERATURE=74., RTI=30./

&DEVC XYZ=2,2,3, PROP_ID='Acme Nozzle', QUANTITY='CONTROL',
ID='NOZZLE 1', CTRL_ID='nozzle 1 trigger' /
&DEVC XYZ=2,3,3, PROP_ID='Acme Nozzle', QUANTITY='CONTROL',
ID='NOZZLE 2', CTRL_ID='nozzle 2 trigger' /

&CTRL ID='check links', FUNCTION_TYPE='ANY', INPUT_ID='LINK 1','LINK 2'/
&CTRL ID='delay', FUNCTION_TYPE='TIME_DELAY', INPUT_ID='check links', DELAY=30. /
&CTRL ID='nozzle 1 trigger', FUNCTION_TYPE='ALL', INPUT_ID='delay','LINK 1'/
&CTRL ID='nozzle 2 trigger', FUNCTION_TYPE='ALL', INPUT_ID='delay','LINK 2'/

```

18.5.11 Example Case: activate_vents

The simple test case called `activate_vents` demonstrates several of the control functions. Figure 18.8 shows seven differently colored vents that activate at different times, depending on the particular timing or control function.

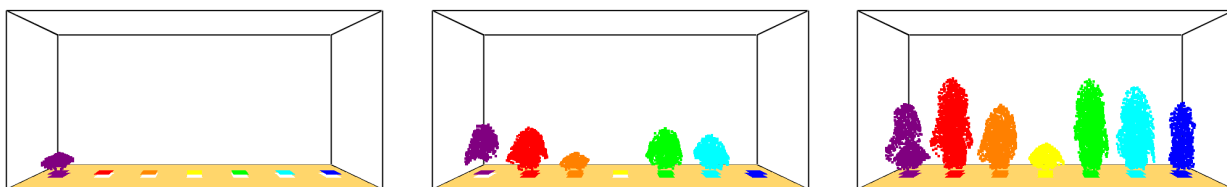


Figure 18.8: Output of the `activate_vents` test case at 5, 10, and 15 s.

18.6 Controlling a RAMP

18.6.1 Changing the Independent Variable

For any user-defined `RAMP`, the normal independent variable, for example time for `RAMP_V`, can be replaced by the output of a `DEVC`. This is done by specifying the input `DEVC_ID` on one of the `RAMP` input lines. When this is done, the current output of the `DEVC` is used as the independent variable for the `RAMP`. A `CTRL_ID` can also be specified as long as the control function outputs a numerical value (i.e., is a mathematical function (Sec. 18.5.6) or a PID function (Sec. 18.5.7)). In the following example a blower is ramped from 0 % flow at 20 °C, to 50 % flow when the temperature exceeds 100 °C, and to 100 % flow when the temperature exceeds 200 °C. This is similar functionality to the `CUSTOM` control function, but it allows for variable response rather than just on or off.

```

&SURF ID='BLOWER', VEL=-2, RAMP_V='BLOWER RAMP' /
&DEVC XYZ=2,3,3, QUANTITY='TEMPERATURE', ID='TEMP DEVC' /
&RAMP ID='BLOWER RAMP', T= 20,F=0.0, DEVC_ID='TEMP DEVC' /
&RAMP ID='BLOWER RAMP', T=100,F=0.5 /

```

```
&RAMP ID='BLOWER RAMP', T=200,F=1.0 /
```

18.6.2 Changing the Dependent Variable

The output of any `DEVC` or `math CTRL` can be used in place of a `RAMP` if either `DEVC_ID_DEP` or `CTRL_ID_DEP` is specified on the `RAMP` input. With either input, the numerical value of the `DEVC` or `CTRL` is used as the `RAMP` output. For example, using the mathematical functions built into the control feature, you can specify the sine function:

$$f(t) = \sin\left(\frac{2\pi(t-5)}{10}\right) \quad (18.15)$$

via the lines

```
&DEVC ID='t', QUANTITY='TIME', XYZ=0,0,0 /
&CTRL ID='t-t0', FUNCTION_TYPE='SUM', INPUT_ID='t','CONSTANT', CONSTANT=-5. /
&CTRL ID='2*pi*(t-t0)/10', FUNCTION_TYPE='MULTIPLY', INPUT_ID='CONSTANT','t-t0',
CONSTANT=0.62831853 /
&CTRL ID='sin(2*pi*(t-t0)/10)', FUNCTION_TYPE='SIN', INPUT_ID='2*pi*(t-t0)/10' /
&RAMP ID='ramp sin', CTRL_DEP_ID='sin(2*pi*(t-t0)/10)' /
```

The function is shown in Fig. 18.9.

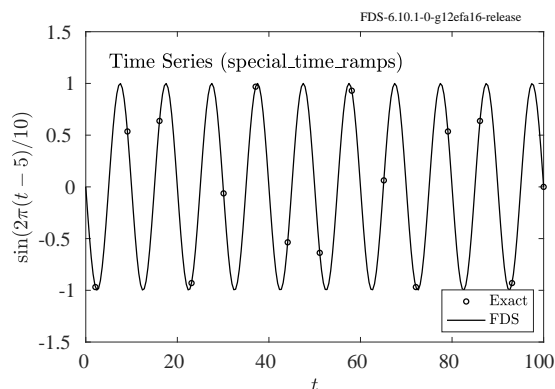


Figure 18.9: Sample sine function time ramp.

18.6.3 Freezing the Output Value, Example Case: `hrr_freeze`

There are occasions where you may want the value of a `RAMP` to stop updating. For example, if you are simulating a growing fire in a room with sprinklers, you may wish to stop the fire from growing when a sprinkler over the fire activates. This type of action can be accomplished by changing the input of the `RAMP` to a `DEVC` (see the previous section) and then giving that `DEVC` either a `NO_UPDATE_DEVC_ID` or a `NO_UPDATE_CTRL_ID`. When the specified controller changes its state to `T` it will cause the `DEVC` to stop updating its value. Since the `DEVC` is being used as the independent variable to a `RAMP`, the `RAMP` will have its output remain the same. This is shown in the example below. A fire is given a linear `RAMP` from 0 to 1000 kW/m² over 50 s. Rather than using the simulation time, the `RAMP` uses a `DEVC` for the time. The timer is set to freeze when another `DEVC` measuring time reaches 200 °C. Figure 18.10 shows the result of these inputs in the test case

`hrr_freeze` where it can be seen that the pyrolysis rate stops increasing once the gas temperature reaches 200 °C.

```
&SURF ID='FIRE', HRRPUA=1000., RAMP_Q='FRAMP', COLOR='ORANGE' /
&RAMP ID='FRAMP', T= 0, F=0, DEVC_ID='FREEZE TIME' /
&RAMP ID='FRAMP', T=50, F=1 /
&DEVC XYZ=..., QUANTITY='TEMPERATURE', SETPOINT=200., INITIAL_STATE=F,
      ID='TEMP' /
&DEVC XYZ=..., QUANTITY='TIME', NO_UPDATE_DEVC_ID='TEMP', ID='FREEZE TIME' /
```

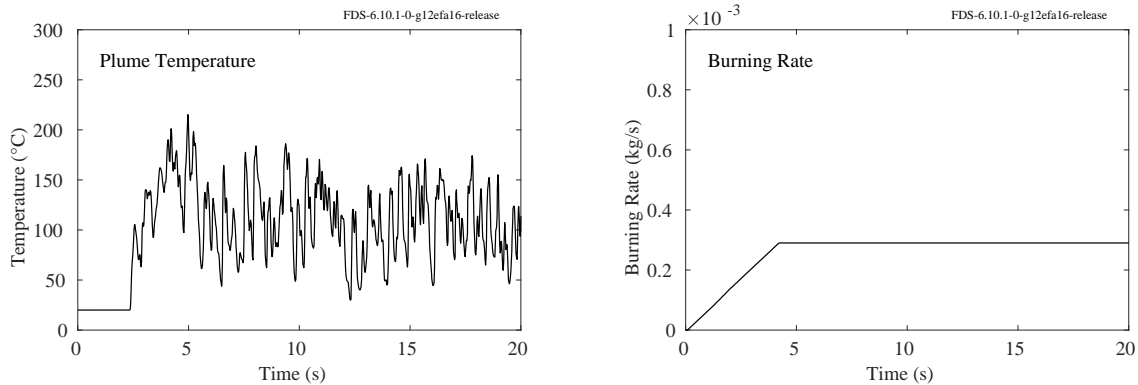


Figure 18.10: Temperature (left) and burning rate (right) outputs of the `hrr_freeze` test case.

It should be noted that devices are updated sequentially in the order that they are listed in the input file and that devices in different meshes do not share values until the end of a time step. This means that if the device being frozen is on a different mesh or is listed before the device that freezes it, it will not be frozen until the next time step.

18.6.4 Example Case: Heat Release Rate of a Spreading Fire

In this example, a fire spreads radially from a single point as directed by the parameters `SPREAD_RATE` and `XYZ` on a `VENT` line. Usually, the user specifies the heat release rate per unit area (`HRRPUA`) for each burning surface cell on the corresponding `SURF` line, but in this case, a specific time `RAMP` for the *total* heat release rate is specified. The following input lines show how the user-specified `RAMP` called 'HRR' controls the total HRR of the growing fire. The key point is that the user-specified *total* HRR is divided by the area of burning surface, and this heat release rate per unit area is imposed on all burning cells. Normally FDS will adjust a mass flux input (`MASS_FLUX`, `HRRPUA`, etc.) input to account for any differences in the area of the `VENT` as specified with `XB` and the area it is actually resolved on the grid. In this case we are using control functions to determine the heat release rate. In this case the control logic is directly computing the required flux based on the area as resolved so no additional correction is needed. When false, the `AREA_ADJUST` flag prevents any additional adjustment. Regardless of the fact that the spreading fire reaches a barrier and is stopped from spreading radially, the user-specified `RAMP` controls the HRR, as shown in Fig. 18.11.

```
&VENT SURF_ID='FIRE', XB=1.0,7.0,1.0,7.0,0.0,0.0, XYZ=3.0,3.0,0.0, SPREAD_RATE=0.1,
      AREA_ADJUST=F /

&SURF ID='FIRE', HRRPUA=1., RAMP_Q='HRRPUA RAMP' /
&RAMP ID='HRRPUA RAMP', T=0, F=0, CTRL_ID_DEP='HRRPUA CTRL' /
&DEVC ID='TIMER', XYZ=1,1,1, QUANTITY='TIME' /
```

```

&CTRL ID='HRR', FUNCTION_TYPE='CUSTOM', INPUT_ID='TIMER', RAMP_ID='HRR' /
&CTRL ID='HRR+EPS', FUNCTION_TYPE='SUM', INPUT_ID='HRR', 'CONSTANT',
    CONSTANT=1E-10 / Prevents divide-by-zero
&DEVC ID='Fire Area', QUANTITY='HRRPUA', QUANTITY_RANGE(1)=1E-10,
    XB=0,8,0,8,0,1, SURF_ID='FIRE', SPATIAL_STATISTIC='SURFACE AREA' /
&CTRL ID='HRRPUA CTRL', FUNCTION_TYPE='DIVIDE', INPUT_ID='HRR+EPS','Fire Area' /

&RAMP ID='HRR', T= 0., F= 0.0 /
&RAMP ID='HRR', T= 10., F= 100.0 /
&RAMP ID='HRR', T= 20., F= 400.0 /
...

```

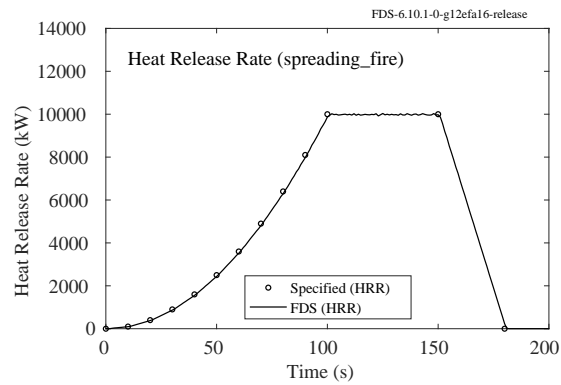


Figure 18.11: Specified (circles) and simulated (line) HRR curve for a spreading fire.

18.7 Visualizing FDS Devices in Smokeview

This section provides an overview of various objects that can be drawn by Smokeview and how to customize their appearance. Further technical details may be found in the Smokeview User's Guide [2].

18.7.1 Devices that Indicate Activation

Devices like sprinklers and smoke detectors can be drawn in one of two ways so as to indicate activation. When FDS determines that a device has activated it places a message in the .smv file indicating the object number, the activation time and the state (0 for inactive or 1 for active). Smokeview then draws the corresponding object. See Tables 18.3 and 18.4 for images.

The character string, SMOKEVIEW_ID, on the PROP line associates an FDS device with a Smokeview object. For example, the following lines instruct Smokeview to draw the device in the shape of a 'target':

```
&PROP ID='my target', SMOKEVIEW_ID='target' /  
&DEVC XYZ=0.5,0.8,0.6, QUANTITY='TEMPERATURE', PROP_ID='my target' /
```

Table 18.3: Single frame static objects



SMOKEVIEW_ID	Image
sensor	
target	

Table 18.4: Dual frame static objects

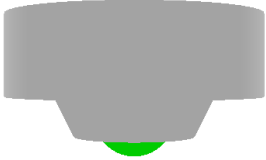



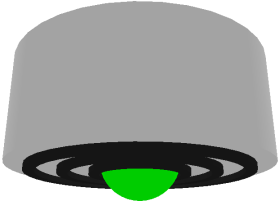
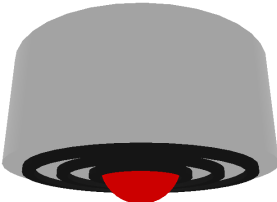

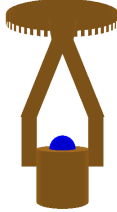

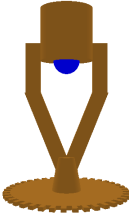
SMOKEVIEW_ID	Image	
	inactive	active
heat_detector		
nozzle		
smoke_detector		
sprinkler_upright		

Table 18.4: Dual frame static objects (continued)

SMOKEVIEW_ID	Image	
	inactive	active
sprinkler_pendent		

18.7.2 Devices with Variable Properties

The appearance of Smokeview objects may be modified using data specified with the array of character strings called `SMOKEVIEW_PARAMETERS` on the `PROP` line. For example, the input lines

```
&PROP ID='ballprops', SMOKEVIEW_ID='ball',
      SMOKEVIEW_PARAMETERS (1:6)='R=255','G=0','B=0','DX=0.5','DY=0.25','DZ=0.1' /
&DEVC XYZ=0.5,0.8,1.5, QUANTITY='TEMPERATURE', PROP_ID='ballprops' /
```

create an ellipsoid colored red with x , y , and z axis diameters of 0.5 m and 0.25 m and 0.1 m, respectively. Note that these parameters are enclosed within single quotes because they are character strings passed to Smokeview.

Table 18.5 lists objects with variable properties. Note that the `tsphere` object uses a texture map or image to alter its appearance. The texture map is specified by placing the characters `t%` before the texture file name, for example, `t%texturefile.jpg`.

Table 18.5: Dynamic Smokeview objects



SMOKEVIEW_ID	SMOKEVIEW_PARAMETERS	Image
ball	<p>SMOKEVIEW_PARAMETERS (1:6) = 'R=128', 'G=192', 'B=255', 'DX=0.5', 'DY=.75', 'DZ=1.0'</p> <p>R, G, B - color components (0 to 255) DX, DY, DZ - amount ball is stretched along x, y, z axis (m)</p>	
cone	<p>SMOKEVIEW_PARAMETERS (1:5) = 'R=128', 'G=255', 'B=192', 'D=0.4', 'H=0.6'</p> <p>R, G, B - color components (0 to 255) D, H - diameter and height (m)</p>	

Table 18.5: Dynamic Smokeview objects (continued)

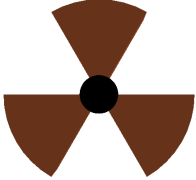



SMOKEVIEW_ID	SMOKEVIEW_PARAMETERS	Image
fan	<pre>SMOKEVIEW_PARAMETERS (1:11) = 'HUB_R=0', 'HUB_G=0', 'HUB_B=0', 'HUB_D=0.1', 'HUB_L=0.12', 'BLADE_R=128', 'BLADE_G=64', 'BLADE_B=32', 'BLADE_ANGLE=60.0', 'BLADE_D=0.5', 'BLADE_H=0.09'</pre> <p>HUB_R, HUB_G, HUB_B - color components of fan hub (0 to 255) HUB_D, HUB_L - diameter and length of fan hub (m) BLADE_R, BLADE_G, BLADE_B - color components of fan blades (0 to 255) BLADE_ANGLE, BLADE_D, BLADE_H - angle, diameter and height of a fan blade</p>	
tsphere	<pre>SMOKEVIEW_PARAMETERS (1:9) = 'R=255', 'G=255', 'B=255', 'AX0=0.0', 'ELEV0=90.0', 'ROT0=0.0', 'ROTATION_RATE=10.0', 'D=1.0', 'tfile="t%sphere_cover_04.png"'</pre> <p>R, G, B - color components (0 to 255) AX0, ELEV0, ROT0 - initial azimuth, elevation and rotation angle (deg) ROTATION_RATE - rotation rate about z axis (deg/s) D - diameter (m) tfile - name of texture map file</p>	

Table 18.5: Dynamic Smokeview objects (continued)

SMOKEVIEW_ID	SMOKEVIEW_PARAMETERS	Image
vent	<pre>SMOKEVIEW_PARAMETERS(1:6) = 'R=192', 'G=192', 'B=128', 'W=0.5', 'H=1.0', 'ROT=90.0'</pre> <p>R, G, B - color components (0 to 255) W, H - width and height (m) ROT - rotation angle (deg)</p>	 <p>inactive vent</p>  <p>active vent</p>

18.7.3 Objects that Represent Lagrangian Particles

Lagrangian particles, like water droplets or small solid particles, are represented in Smokeview as tiny points. However, it is possible to draw Lagrangian particles in other ways, such as those depicted in Table 18.6. For example, the following lines define particles that represent segments of electrical cables that are 10 cm long with a diameter of 1.24 cm:

```
&PART ID='cables', QUANTITIES(1)='PARTICLE TEMPERATURE', ..., PROP_ID='cable image' /
&PROP ID='cable image', SMOKEVIEW_ID='tube', SMOKEVIEW_PARAMETERS='L=0.1', 'D=0.0124' /
```

By default, the cables are colored black, but you can specify your own default color using the parameters R, G, and B. In addition, you can color the particles according to the listed QUANTITIES on the PART line. Menus in Smokeview allow you to toggle between the various color options.

You can control the orientation of the 'tube' objects using a parameter such as 'RANDXY=1' that causes the cylinders to be drawn randomly in the $x-y$ plane. Objects with the parameters U-VEL, V-VEL, and W-VEL stretch according to the respective velocity components associated with the moving particles.

Table 18.6: Dynamic Smokeview objects for Lagrangian particles

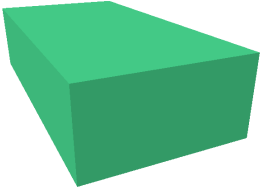


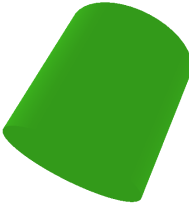
SMOKEVIEW_ID	SMOKEVIEW_PARAMETERS	Image
box	<pre>SMOKEVIEW_PARAMETERS(1:6) = 'R=192', 'G=255', 'B=128', 'DX=0.25', 'DY=.5', 'DZ=0.125'</pre> <p>R, G, B - color components (0 to 255) DX, DY, DZ - amount box is stretched along axes</p>	

Table 18.6: Dynamic Smokeview objects for Lagrangian particles (continued)

SMOKEVIEW_ID	SMOKEVIEW_PARAMETERS	Image
tube	<p>SMOKEVIEW_PARAMETERS (1:6) = 'R=255', 'G=0', 'B=0', 'D=0.2', 'L=0.6', 'RANDXY=1'</p> <p>R, G, B - color components (0 to 255) D, L - diameter and length (m) RANDXY - randomly orient in x-y plane RANDXZ - randomly orient in x-z plane RANDYZ - randomly orient in y-z plane RANDXYZ - random orientation DIRX, DIRY, DIRZ - orient along axis</p>	
velegg	<p>SMOKEVIEW_PARAMETERS (1:9) = 'R=192', 'G=64', 'B=32' 'U-VEL=1.', 'V-VEL=1.', 'W-VEL=1.' 'VELMIN=0.01', 'VELMAX=0.2', 'D=1.0'</p> <p>R, G, B - color components (0 to 255) U-VEL, V-VEL, W-VEL - velocity components (m/s) VELMIN, VELMAX - minimum and maximum velocity D - diameter of egg at maximum velocity (m)</p>	
veltube	<p>SMOKEVIEW_PARAMETERS (1:9) = 'R=0', 'G=0', 'B=0' 'U-VEL=1.', 'V-VEL=1.', 'W-VEL=1.' 'VELMIN=0.01', 'VELMAX=0.2', 'D=0.1'</p> <p>R, G, B - color components (0 to 255) U-VEL, V-VEL, W-VEL - velocity components (m/s) VELMIN, VELMAX - minimum and maximum velocity D - diameter of tube at VELMAX (m)</p>	

18.8 External Control of FDS (Beta)

It is possible to externally control some aspects of an FDS simulation while the simulation is running. Any FDS input that has either a `RAMP_ID` (except for the output time ramps on `DUMP` such as `RAMP_BNDF`) or a `CTRL_ID` as one of its inputs can be externally controlled. A `RAMP` can be externally controlled by setting `EXTERNAL_FILE` on `RAMP`. A `CTRL` can be externally controlled by setting `FUNCTION_TYPE='EXTERNAL'`.

When external control is selected, FDS will set the value of the `RAMP` or the logical state of the `CTRL` based on values contained in a csv file whose name is given by `EXTERNAL_FILENAME` on `MISC`. This file will be checked for new values every `DT_EXTERNAL` on `TIME` seconds. Only those inputs whose values are being changed need to be specified. For inputs not specified or if the file does not exist or cannot be opened (e.g., locked by the operating system during a file write), the current values will be kept. The initial values are defined with either `INITIAL_VALUE` on `RAMP` or `INITIAL_STATE` on `CTRL`. FDS can be forced to wait if no file is present or use an alternate `EXTERNAL_FILENAME` by specifying `DT_EXTERNAL_HEARTBEAT` and `EXTERNAL_HEARTBEAT_FILENAME` on `TIME`. With this approach FDS looks for `EXTERNAL_HEARTBEAT_FILENAME`. If the file is not found FDS will wait up to `DT_EXTERNAL_HEARTBEAT` seconds. If `EXTERNAL_HEARTBEAT_FILENAME` is still not present, FDS will stop trying to update the external controls. If `HEARTBEAT_FAIL` on `TIME` is true, FDS will stop the simulation. If `EXTERNAL_HEARTBEAT_FILENAME` is present, FDS will read `EXTERNAL_FILENAME` from it.

The csv file format is:

```
Type(1), ID(1), Value(1)
Type(2), ID(2), Value(2)
.
.
```

where `Type` is either `RAMP` or `CTRL`, `ID` is the corresponding `ID` in the FDS input files, and `Value` is the new output value for a `RAMP` or the new logical state for a `CTRL` where positive values mean the `CTRL` evaluates as `TRUE` and negative values mean the function evaluates as `FALSE`.

An example set of inputs is below:

```
&MISC EXTERNAL_FILENAME='external_test_input.csv'/
&TIME T_END=10,DT_EXTERNAL=5/

&OBST XB=0.0,0.1,0,1,0,1/
&VENT XB=0.1,0.1,0,1,0,1,SURF_ID='FLOW',CTRL_ID='VENT CTRL'/
&SURF ID='FLOW',VEL=-1,COLOR='BLUE',TAU_V=0/
&CTRL ID='VENT CTRL',FUNCTION_TYPE='EXTERNAL',INITIAL_STATE=T/

&VENT XB=2,2,0,1,0,1,SURF_ID='HOT'/
&SURF ID='HOT',TMP_FRONT=1000,RAMP_T='T RAMP',COLOR='RED'/
&RAMP ID='T RAMP',EXTERNAL_FILE=T,INITIAL_VALUE=0.081633/
```

In these inputs a vent with a fixed velocity is externally controlled by a `CTRL` that is initially `TRUE`, and a vent with a fixed temperature is externally controlled with a `RAMP` with an initial ramp value that results in a temperature of 100 °C. These vents are applied to two faces of a 1 m³ box. At 5 s, the csv file is read with contents of:

```
RAMP,"T RAMP",1
CTRL,"VENT CTRL",-1
```

This turns off the velocity vent (a negative `CTRL` value means set the state to `.FALSE.`) and sets the fixed temperature vent to 1000 °C (a `RAMP` value of 1 means apply the full value of `TMP_FRONT`). Note that the

ID strings are enclosed in quotes. This is only required if the string has a space or comma in it. Results of running the case are shown in Fig. 18.12

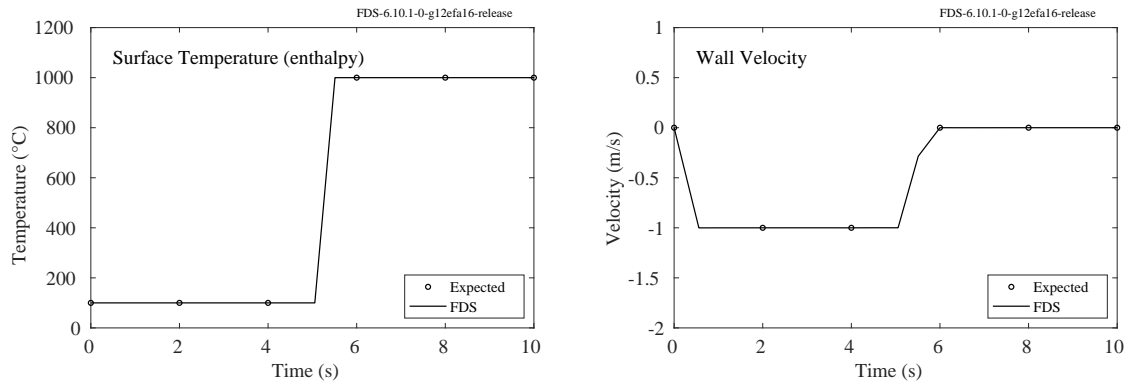


Figure 18.12: (Left) Surface temperature. (Right) Wall velocity.

Chapter 19

Numerical Considerations

This chapter describes various numerical parameters; that is, parameters that control how the governing equations are solved. Most of these parameters are specified on the `MISC` line and for most large-scale fire simulations, their default values are recommended. `MISC` is the namelist group of input parameters that do not fall into any one category (Table 23.18).

It is good practice to use only one `MISC` line in the data file. Using multiple `MISC` lines is possible, but be careful about not over writing a parameter. The last parameter read will take precedence.

19.1 Simulation Mode

There are four basic modes of operation in FDS: `'DNS'` (Direct Numerical Simulation), `'LES'` (Large Eddy Simulation), `'VLES'` (Very Large Eddy Simulation), and `'SVLES'` (Simple Very Large Eddy Simulation—VLES with simplified physics). These modes govern a number of physical and numerical parameters that determine the level of physics and the accuracy of the numerical model. They are specified using `SIMULATION_MODE` on the `MISC` line. For example

```
&MISC ..., SIMULATION_MODE='DNS' /
```

indicates a direct numerical simulation. The default value is `'VLES'`.

Table 19.1 indicates the value of various parameters for each of the four simulation modes and the sections where these parameters are explained in detail.

Table 19.1: Parameters effected by `SIMULATION_MODE`.

Key Parameter	Section	'DNS'	'LES'	'VLES'	'SVLES'
<code>CFL_VELOCITY_NORM</code>	19.3.1	1	1	2	3
<code>CHECK_VN</code>	19.3.2	T	T	T	F
<code>FLUX_LIMITER</code>	19.4	'CHARM'	'CHARM'	'SUPERBEE'	'SUPERBEE'
<code>CONSTANT_SPECIFIC_HEAT_RATIO</code>	12.1.3	F	F	F	T
<code>EXTINCTION_MODEL</code>	13.1.6	2	2	1	1

19.2 Large Eddy Simulation Parameters

By default FDS uses the Deardorff [68, 69] turbulent viscosity,

$$(\mu_{\text{LES}}/\rho) = C_v \Delta \sqrt{k_{\text{sgs}}} \quad (19.1)$$

where $C_v = 0.1$ and the subgrid scale (sgs) kinetic energy is taken from an algebraic relationship based on scale similarity (see the FDS Technical Reference Guide [3]). The LES filter width is taken as the geometric mean¹ of the local mesh spacing in each direction, $\Delta = (\delta x \delta y \delta z)^{1/3}$. You may also choose a constant filter width by setting `FIXED_LES_FILTER_WIDTH` on the `MISC` line, in meters. This is not a recommended thing to do in practice, but may be useful for convergence studies and other diagnostic tests.

Options for the `TURBULENCE_MODEL` on the `MISC` line are listed in Table 19.2. Note that the model used in FDS versions 1-5 is `'CONSTANT SMAGORINSKY'`. The thermal conductivity and material diffusivity are related to the turbulent viscosity by:

$$k_{\text{LES}} = \frac{\mu_{\text{LES}} c_p}{\text{Pr}_t} \quad ; \quad (\rho D)_{\text{LES}} = \frac{\mu_{\text{LES}}}{\text{Sc}_t} \quad (19.2)$$

The turbulent Prandtl number Pr_t and the turbulent Schmidt number Sc_t are assumed to be constant for a given scenario. Although it is not recommended for most calculations, you can modify $\text{Pr}_t = 0.5$, and $\text{Sc}_t = 0.5$ via the parameters `PR`, and `SC` on the `MISC` line. A more detailed discussion of these parameters is given in the FDS Technical Reference Guide [3].

Table 19.2: Turbulence model options.

TURBULENCE_MODEL	Description	Coefficient(s)
'CONSTANT SMAGORINSKY'	Constant-coefficient Smagorinsky [70]	<code>C_SMAGORINSKY</code>
'DYNAMIC SMAGORINSKY'	Dynamic Smagorinsky [71, 72]	None
'DEARDORFF'	Deardorff [68, 69]	<code>C_DEARDORFF</code>
'VREMAN'	Vreman's eddy viscosity [73]	<code>C_VREMAN</code>
'WALE'	Wall-Adapting Local Eddy-viscosity [74]	<code>C_WALE</code>

Near-Wall Turbulence Model

By default, FDS uses the WALE model of Nicoud and Ducros [74] for the eddy viscosity in the first off-wall grid cell because the test filtering operation for the Deardorff model is ill-defined near a wall. The WALE model has the advantage that the eddy viscosity tends to zero at the correct rate as you approach the wall where the no slip condition applies.

As an alternative, you may choose to use the constant coefficient Smagorinsky [70] model with Van Driest damping (see [75]). To invoke this model, set `NEAR_WALL_TURBULENCE_MODEL='VAN DRIEST'` on the `SURF` line.

For diagnostic purposes, or in cases where neither of the other wall models above is appropriate, it is possible to set a constant value of the near wall kinematic eddy viscosity for a given `SURF` using `NEAR_WALL_EDDY_VISCOSITY`.

Note that the near-wall turbulence model sets the eddy viscosity, μ_t , near the wall, *not* the wall stress, τ_w . The wall stress depends on the choice of wall function, as discussed in Sec. 10.1.7.

¹An alternative to the geometric mean filter width type is to use the maximum cell dimension, $\Delta = \max(\delta x, \delta y, \delta z)$. This should be used sparingly but may be necessary when the cell aspect ratios are high. Using `LES_FILTER_TYPE='MAX'` on `MISC` will lead to a larger value of turbulent viscosity and hence a more dissipative numerical solution.

19.3 Numerical Stability Parameters

FDS uses an explicit time advancement scheme; thus, the time step plays an important role in maintaining numerical stability and accuracy. Below we examine the constraints on the time step necessary for stability in the presence of advection, diffusion, and expansion of the velocity and scalar fields. In addition, there are additional constraints that ensure accuracy of various algorithms.

19.3.1 The Courant-Friedrichs-Lewy (CFL) Constraint

The well-known CFL constraint given by

$$\text{CFL} = \delta t \frac{\|\mathbf{u}\|}{\Delta} < 1 \quad (19.3)$$

places a restriction on the time step due to the advection velocity. The limits for the CFL are set by `CFL_MIN` (default 0.8) and `CFL_MAX` (default 1) on `MISC`. Physically, the constraint says that a fluid element should not traverse more than one cell width, Δ , within one time step, δt . For LES, this constraint has the added advantage of keeping the implicit temporal and spatial filters consistent with each other. In other words, in order to resolve an eddy of size Δ , the time step needs to obey the CFL constraint. If one were to employ an implicit scheme for the purpose of taking time steps ten times larger than the CFL limit, the smallest resolvable turbulent motions would then be roughly ten times the grid spacing, which would severely limit the benefit of using LES. In most cases, if you want the simulation to run faster, a better strategy is to coarsen the grid resolution while keeping the CFL close to 1.

The exact CFL needed to maintain stability depends on the order (as well as other properties) of the time integration scheme and the choice of velocity norm. Four choices for velocity norm are available in FDS (set on `MISC`):

`CFL_VELOCITY_NORM=0` (corresponds to L_∞ norm of velocity vector, despite the numerical code this is less restrictive than 1 or 2)

$$\frac{\|\mathbf{u}\|}{\Delta} = \max \left(\frac{|u|}{\delta x}, \frac{|v|}{\delta y}, \frac{|w|}{\delta z} \right) + |\nabla \cdot \mathbf{u}| \quad (19.4)$$

`CFL_VELOCITY_NORM=1` (DNS and LES defaults, most restrictive, corresponds to L_1 norm of velocity vector)

$$\frac{\|\mathbf{u}\|}{\Delta} = \frac{|u|}{\delta x} + \frac{|v|}{\delta y} + \frac{|w|}{\delta z} + |\nabla \cdot \mathbf{u}| \quad (19.5)$$

`CFL_VELOCITY_NORM=2` (VLES default, L_2 norm of velocity vector)

$$\frac{\|\mathbf{u}\|}{\Delta} = \sqrt{\left(\frac{u}{\delta x} \right)^2 + \left(\frac{v}{\delta y} \right)^2 + \left(\frac{w}{\delta z} \right)^2} + |\nabla \cdot \mathbf{u}| \quad (19.6)$$

`CFL_VELOCITY_NORM=3` (SVLES default, least restrictive, corresponds to L_∞ norm of velocity vector without the velocity divergence)

$$\frac{\|\mathbf{u}\|}{\Delta} = \max \left(\frac{|u|}{\delta x}, \frac{|v|}{\delta y}, \frac{|w|}{\delta z} \right) \quad (19.7)$$

The last listed form of the constraint is the least restrictive, but also the most dangerous in the sense that a numerical instability is more likely to occur when the CFL constraint is least restrictive. This option is akin to a high optimization level of a computer program compiler—there is a trade-off between added speed and added risk of failure.

Notice that the CFL norms 0-2 include the divergence of the velocity field. This is an added safeguard because often numerical instabilities arise when there is a sudden release of energy and a corresponding increase in the divergence within a single grid cell. In an explicit Euler update of the continuity equation, if the time increment is too large the grid cell may be totally drained of mass, which, of course, is not physical. The constraint $\rho^{n+1} > 0$ therefore leads to the following restriction on the time step:

$$\delta t < \frac{\rho^n}{\bar{\mathbf{u}}^n \cdot \nabla \rho^n + \rho^n \nabla \cdot \mathbf{u}^n} \quad (19.8)$$

We can argue that the case we are most concerned with is when ρ^n is near zero. A reasonable approximation to (19.8) then becomes

$$\delta t < \frac{\rho}{\bar{u}_i \left(\frac{\rho-0}{\delta x_i} \right) + \rho \nabla \cdot \mathbf{u}} = \left[\frac{\bar{u}_i}{\delta x_i} + \nabla \cdot \mathbf{u} \right]^{-1} \quad (19.9)$$

Equation (19.9) adds the effect of thermal expansion to the CFL constraint. Further, note that the dot product implies summation over the subscripts i , which provides incentive for using the L_1 norm of the velocity vector as in `CFL_VELOCITY_NORM=1`.

Handling cells with large aspect ratios In LES, it is usually better to use cubic cells to accurately capture turbulence. However, there are situations where elongated or pancake shaped cells become necessary for computational efficiency, usually in either tunnel or atmospheric flow applications. As the cell aspect ratio increases, the choice of velocity norm for the time step restriction becomes more important. Numerical experiments have shown that beyond an aspect ratio of about 4:1, using `CFL_VELOCITY_NORM=1` is required to avoid unphysical oscillations in the ambient temperature. Thus, if not explicitly stated in the FDS input file, beyond a maximum cell aspect ratio of 4:1 FDS will automatically switch the stability check to use `CFL_VELOCITY_NORM=1`. The velocity norm used by FDS is reported in the output file (`CHID.out`).

Time step restrictions to avoid clipping If the CFL constraint is not sufficient to maintain realizable mass fractions ($0 \leq Z_\alpha \leq 1$) then FDS will attempt to redistribute mass to the neighboring cells, as discussed in the FDS Tech Guide [3]. If this fails, then a 10% time step restriction will be applied and the mass transport equations will be reiterated. Controlling the number of time step restrictions is discussed in Sec. 19.5.

19.3.2 The Von Neumann Constraint

The Von Neumann constraint is given by

$$\text{VN} \equiv 2 \delta t \max \left[\frac{\mu}{\rho}, D_\alpha \right] \left(\frac{1}{\delta x^2} + \frac{1}{\delta y^2} + \frac{1}{\delta z^2} \right) < 1 \quad (19.10)$$

The limits for VN may be adjusted using `VN_MIN` (default 0.8 for all forms of LES, 0.4 for DNS) and `VN_MAX` (default 1.0 for all forms of LES, 0.5 for DNS) on `MISC`. We can understand this constraint in a couple of different ways. First, we could consider the model for the diffusion velocity of species α in direction i , $V_{\alpha,i} Y_\alpha = -D_\alpha \partial Y_\alpha / \partial x_i$, and we would then see that VN is simply a CFL constraint due to diffusive transport.

We can also think of VN in terms of a total variation diminishing (TVD) constraint. That is, if we have variation (curvature) in the scalar field, we do not want to create spurious oscillations that can lead to an instability by overshooting the smoothing step. Consider the following explicit update of the heat equation for u in 1-D. Here subscripts indicate grid indices and v is the diffusivity.

$$u_i^{n+1} = u_i^n + \frac{\delta t v}{\delta x^2} (u_{i-1}^n - 2u_i^n + u_{i+1}^n) \quad (19.11)$$

Very simply, notice that if $\delta t v / \delta x^2 = 1/2$ then $u_i^{n+1} = (u_{i-1}^n + u_{i+1}^n)/2$. If the time step is any larger we overshoot the straight line connecting neighboring cell values. Of course, this restriction is only guaranteed to be TVD if the u field is “smooth”; otherwise, the neighboring cell values may be shifted in the opposite direction. Unfortunately, in LES there is no such guarantee and so the VN constraint can be particularly devilish in generating instabilities. For this reason, some practitioners like to employ implicit methods for the diffusive terms. The VN constraint is checked by default in all simulation modes. It may be turned off by setting `CHECK_VN=F` on `MISC`.

19.3.3 Stability of particle transport

The movement of Lagrangian particles over the course of a time step is calculated using an analytical solution and remains stable regardless of the time step used by the flow solver. However, if the particle moves over the width of several grid cells in a single time step, the momentum transferred between the particle and the gas cannot be allocated properly to all of the affected cells. To overcome this problem, FDS subdivides the gas phase time step based on each particle’s velocity. For example, if the particle travels across two cells in a single gas phase time step, then its trajectory is calculated by subdividing the time step into two.

In some cases with extremely fast particles, however, the stability of the overall flow behavior may require setting an additional parameter that limits the time step of the flow solver according to the speed of the fastest particle in the simulation. The actual value of the constraint is set using `PARTICLE_CFL_MAX` on the `MISC` line. A value of 1 (default) means that the fastest moving particle can move a distance of one grid cell during the time step. Because very fast nozzle velocities can cause extremely small time steps and hence very long run times, the `PARTICLE_CFL` constraint is set to `F` by default. Setting `PARTICLE_CFL` to `T` on the `MISC` line activates this constraint.

19.3.4 Heat Transfer Constraint

Note that the heat flux has units of W/m^2 . Thus, a velocity scale may be formed from $(\dot{q}_w''/\rho_w)^{1/3}$, where ρ_w is the gas phase density at the wall and \dot{q}_w'' is the total heat flux (convective plus net radiative) at the wall. Anytime we have a velocity scale to resolve, we have a CFL-type stability restriction. Therefore, the heat transfer stability check loops over all wall cells to ensure $\delta t < (\delta x/2)/(\dot{q}_w''/\rho_w)^{1/3}$. This check is invoked by setting `CHECK_HT=T` on the `MISC` line. It is `F` by default.

19.4 Flux Limiters

FDS employs *total variation diminishing* (TVD) schemes for scalar transport. The default for VLES (FDS default `SIMULATION_MODE`) is Superbee [76], so chosen because this scheme does the best job preserving the scalar variance in highly turbulent flows with coarse grid resolution. The default scheme for DNS and LES is CHARM [77] because the gradient steepening used in Superbee forces a stair step pattern at high resolution, while CHARM is convergent. A few other schemes (including Godunov and central differencing) are included for completeness; more details can be found in the Tech Guide [1]. Table 19.3 below shows the character strings which may be used to invoke the various limiter schemes.

```
&MISC FLUX_LIMITER='GODUNOV' / ! invoke Godunov (first-order upwind scheme)
```

Table 19.3: Flux limiter options.

Scheme	FLUX_LIMITER
Central differencing	'CENTRAL '
Godunov	'GODUNOV '
Superbee (VLES, SVLES default)	'SUPERBEE '
MINMOD	'MINMOD '
CHARM (DNS, LES default)	'CHARM '
MP5	'MP5 '

19.5 Limiting the Bounds of Key Variables

The algorithms in FDS are designed to work within a certain range of values for density, temperature and mass fraction. To prevent unphysical results, there are bounds placed on these variables to prevent a single spurious value from causing a numerical instability. It also prevents out of range errors from calls to temperature-dependent look-up tables. By default, FDS determines the lowest and highest values of the variables based on your input, but it is not possible in all cases to anticipate just how low or high a given value might be. Thus, on rare occasions you might need to set upper or lower bounds on the density or temperature. Temperature and density bounds are input under the namelist group called `CLIP`. The parameters are listed in Table 23.4. You only need to set these values if you notice that one of them appears to be “cut off” when examining the results in Smokeview. For typical fire scenarios, you need not set these values, but if you anticipate relatively low or high values in an unusual case, take a look at the calculation results to determine if a change in the bounds is needed.

19.5.1 Temperature

The gas temperature is not solved for directly. Rather, the gas density and species mass fractions determine the gas temperature via the equation of state. Nevertheless, the calculation of the temperature of liquid droplets and solid obstructions do make use of temperature bounds. The default bounds are

$$\min(T_{\infty}, T_m) - 10 < T < 2727 \text{ }^{\circ}\text{C} \quad (19.12)$$

where T_{∞} is the user-specified ambient temperature and T_m is the melting temperature of water. These bounds are widened by user-specified temperatures, like an initial solid or droplet temperature. To set your own bounds, use `MINIMUM_TEMPERATURE` and/or `MAXIMUM_TEMPERATURE` on the `CLIP` line.

One other consideration related to high temperatures is that FDS uses tabulated gas and solid property data up to a temperature of 5000 K. If for some reason you expect higher temperatures, set the integer parameter `I_MAX_TEMP` on the `MISC` line. This sets the upper dimension of many property arrays. Even though this parameter is an integer, it can be thought of as a maximum temperature in units of K. Note that temperature-dependent properties like specific heat at temperatures above 5000 K remain fixed, but integrated quantities like enthalpy take into account the higher temperature.

19.5.2 Density

The density of the gas has a natural lower bound of zero, but if the density in a cell decreases to nearly zero, the temperature would then increase to an extremely high value due to the equation of state. Thus, by default, the density is kept within the following range:

$$\min\left(0.1\rho_{\infty}, \frac{2p_{\infty}W_{\min}}{RT_{\max}}\right) < \rho < \frac{2p_{\infty}W_{\max}}{RT_{\min}} \quad (19.13)$$

where W_{\min} and W_{\max} are the minimum and maximum values of the molecular weight of the tracked gas species in units of g/mol, and R is the universal gas constant, 8314.5 J/(kmol·K). T_{\min} and T_{\max} are described above. To override the limits of density, specify `MINIMUM_DENSITY` and/or `MAXIMUM_DENSITY` on the `CLIP` line in units of kg/m³.

Clipping of density and mass fractions violates mass conservation, so it is preferable to avoid clipping if possible. As discussed in Sec. 19.3, the time step is set to adhere to the CFL constraints of the flow field. The proper `DT` combined with flux limiters generally avoids the need for clipping. Beyond this, FDS then employs a mass redistribution scheme, as discussed in FDS Technical Guide [3]. If this fails, there is yet one more attempt to avoid clipping—the time step is decreased by 10 % ($\delta t_{\text{new}} = 0.9 \delta t$) and the scalar transport equations are reiterated. This process is carried out a maximum of `CLIP_DT_RESTRICTIONS_MAX` times; the default is 5. In some very extreme circumstances, this loop can drive the time step into numerical instability range ($\delta t / \delta t_{\text{init}} < \text{LIMITING_DT_RAT}$). You can control the max number of time step restrictions by setting the parameter `CLIP_DT_RESTRICTIONS_MAX` on the `CLIP` line (set to 0 to bypass the algorithm altogether). The number of restrictions (if any) is noted in the `CHID.out` file for a given time step.

Chapter 20

Changing the Initial Conditions

Typically, an FDS simulation begins at time $t = 0$ with ambient conditions. The air temperature is assumed constant with height, and the density and pressure decrease with height (the z direction). This decrease is not noticed in most building scale calculations, but it is important in large outdoor simulations.

There are some scenarios for which it is convenient to change the ambient conditions within rectangular regions of the domain using the namelist keyword `INIT` (Table 23.14). There can be multiple `INIT` lines. If two rectangular regions defined by `INIT` overlap, it is the second of the overlapping regions that takes precedence.

20.1 Gas Species

Species concentrations can be initialized as follows:

```
&INIT XB=0.0,0.1,0.0,0.025,0.0,0.1,  
      SPEC_ID(1)='OXYGEN', MASS_FRACTION(1)=0.23,  
      SPEC_ID(2)='PROPANE', MASS_FRACTION(2)=0.06 /
```

where `XB` indicates a rectangular volume within the domain where the initial mass fractions of oxygen and propane will be initialized to 0.23 and 0.06, respectively. Note the following rules:

- The indices of `SPEC_ID` and `MASS_FRACTION` are not necessarily indicative of the order in which the species are listed in the input file, but rather should come in consecutive order starting from 1.
- `VOLUME_FRACTION(N)` can be used instead of `MASS_FRACTION(N)` as long as they are not used together on the same input line.
- Instead of specifying a `MASS_FRACTION(N)` or `VOLUME_FRACTION(N)`, you can specify a `RAMP_MF_Z(N)` or `RAMP_VF_Z(N)` to indicate a vertical profile. The corresponding `RAMP` lines should use the parameter `Z` for the height z (m) and `F` for the value of the mass or volume fraction. A single `INIT` line can have both fixed and ramped values, but they must refer to either mass or volume fraction.
- Specify all species components on the same `INIT` line.
- The initial concentration of the background gas species cannot be specified this way. The mass or volume fraction of the background species will be set to account for the unspecified fraction.
- All gas species must be specified using the `SPEC` or `REAC` namelist groups. In other words, any species listed must be individually tracked and not just a component of a lumped species. See Chapter 12 for details.

- You may use the shortcut `DB='WHOLE DOMAIN'` in place of `XB=...`, which is equivalent to specifying the entire domain as the region of initialization.

20.2 Temperature and Pressure

You can set the ambient temperature and pressure using the following parameters on the `MISC` line:

- `P_INF` Background pressure (at the ground) in Pa. The default is 101325 Pa.
- `TMPA` Ambient temperature, the temperature of everything at the start of the simulation. The default is 20 °C.

To modify the local initial temperature in certain regions of the domain, add lines of the form,

```
&INIT XB=0.0,0.1,0.0,0.025,0.0,0.1, TEMPERATURE=60. /
```

This indicates that the temperature shall be 60 °C instead of the ambient within the bounds given by `XB`. The `INIT` construct may be useful in examining the influence of stack effect in a building, where the temperature is different inside and outside. To initialize both temperature and species concentration in the same volume, use the same `INIT` line,

```
&INIT XB=0.0,0.1,0.0,0.025,0.0,0.1,
      MASS_FRACTION(1)=0.23, SPEC_ID(1)='OXYGEN',
      MASS_FRACTION(2)=0.06, SPEC_ID(2)='PROPANE',
      TEMPERATURE=60. /
```

You can also specify a `RAMP_TMP_Z` vertical profile. The corresponding `RAMP` lines should use the parameter `Z` for the height z (m) and `F` for the value of the temperature (°C).

20.3 Heat Release Rate Per Unit Volume

The `INIT` line may also be used to specify a volumetric heat source term. For example,

```
&INIT XB=0.0,0.1,0.0,0.025,0.0,0.1, HRRPUV=1000., RADIATIVE_FRACTION=0.25 /
```

indicates that the region bounded by `XB` shall generate 1000 kW/m³, 25 % of which is radiative. The default value of `RADIATIVE_FRACTION` in this context is 0. This feature is mainly useful for diagnostics, or to model a fire in a very simple way. You may specify a time ramp for volumetric heat source via `RAMP_Q`. For example,

```
&RAMP ID='q1', T=0, F=0/
&RAMP ID='q1', T=60, F=1/
&INIT XB=0.0,0.1,0.0,0.025,0.0,0.1, HRRPUV=1000., RAMP_Q='q1' /
```

ramps the heat source up from 0 to 1000 kW/m³ linearly over 60 seconds.

20.4 Velocity Field

20.4.1 Default Initial Velocity Field

FDS initializes the flow field with a very small amount of “noise” to prevent the development of a perfectly symmetric flow when the boundary and initial conditions are perfectly symmetric. This is done by adding a random vortex to each cell. To turn this off, set `NOISE=F`. To control the amount of noise (i.e., the maximum strength of the vortex), set `NOISE_VELOCITY`. Its default value is 0.005 m/s.

FDS uses a predefined random seed for the random number generator which ensures that each mesh is seeded with a different number. A predefined seed means that all simulations will use the same sequence of random numbers for a given mesh. You can change the random seed by specifying `RND_SEED` on the `MISC` line. This defines a constant value that is added to the predefined random seed on each mesh.

20.4.2 Turning off the Flow Field

For certain types of diagnostic tests, it is useful to turn off the velocity field and exercise some other aspect of the model, like radiation or particle transport. To do this, set `FREEZE_VELOCITY=T` on the `MISC` line.

20.4.3 One-Dimensional Velocity Component Fields

For diagnostic or verification purposes it may be useful to initialize velocity components with 1D fields. This can be used in combination with `FREEZE_VELOCITY=T`, for example, to control precisely the amount of drag a particle sees, and so on. The linear fields are specified with the `RAMP` functionality and the ramp is identified using `RAMP_UX`, for example, on `MISC` to specify variation of the u component of velocity in the x direction. Other velocity components and directions are specified similarly (see Table 23.18).

```
&MISC FREEZE_VELOCITY=T, RAMP_UX='u1' /
&RAMP ID='u1', X= 0, F=0 /
&RAMP ID='u1', X= 2, F=4 /
&RAMP ID='u1', X= 4, F=16/
```

20.5 Nonuniform Initial Fields: Velocity, Temperature, Species

It may be useful to start a calculation from an established flow field. Usually this can be accomplished with the normal restart functionality, but you may want to specify your own flow field throughout the domain. This section discusses how this may be done for velocity, temperature, and tracked species.

The fields are stored in a comma-separated value (.csv) files. You have the option of creating these files using FDS or on your own. To generate the file(s) with FDS, specify the time interval between outputs using `DT_UVW`, `DT_TMP`, and `DT_SPEC`, for velocity, temperature, and species, respectively, on the `DUMP` line; you may also use `RAMP_***` (see Sec. 22.1). For example, if you want to write the velocity, temperature, species fields every 10 minutes, add the following:

```
&DUMP DT_UVW=600, DT_TMP=600, DT_SPEC=600 /
```

FDS will then write `CHID_uvw_tN_mM.csv`, etc., for each time, N , and mesh, M .

Velocity format

The format of the files is very simple. They are ASCII formatted files, comma-separated, with integer header and double-precision real data. The top row just indicates the structured mesh extents (relative to a specific mesh). For velocity, note that the raw data values are face centered with index 1 denoting the + side of the face. Hence, `IMIN` will usually be 0, representing the left mesh boundary. (It is perhaps easiest to understand the format by just opening a file that FDS has written for you.)

```
WRITE (LU_UVW) IMIN, IMAX, JMIN, JMAX, KMIN, KMAX
DO K=KMIN, KMAX
  DO J=JMIN, JMAX
    DO I=IMIN, IMAX
      WRITE (LU_UVW, *) U(I, J, K), ', ', V(I, J, K), ', ', W(I, J, K) ! units are m/s
    ENDDO
  ENDDO
ENDDO
```

Temperature format

Similarly, for temperature, except temperature is a scalar stored at the cell center. Hence `IMIN` will usually be 1. Note that units are in K.

```
WRITE (LU_UVW) IMIN, IMAX, JMIN, JMAX, KMIN, KMAX
DO K=KMIN, KMAX
  DO J=JMIN, JMAX
    DO I=IMIN, IMAX
      WRITE (LU_TMP, *) TMP(I, J, K) ! units are K
    ENDDO
  ENDDO
ENDDO
```

Species format

Tracked species are written out as a comma-separated list with each cell as a row in the file. Similar to temperature, species mass fractions are scalars that live at the cell centers.

```
WRITE (LU_UVW) IMIN, IMAX, JMIN, JMAX, KMIN, KMAX
DO K=KMIN, KMAX
  DO J=JMIN, JMAX
    DO I=IMIN, IMAX
      ! FMT is comma-separated
      ! units are mass fraction
      WRITE (LU_SPEC, FMT) ( ZZ(I, J, K, N), N=1, N_TRACKED_SPECIES )
    ENDDO
  ENDDO
ENDDO
```

Reading the Fields

You may read in the 3-D fields using a `CSVF` line. Only one `CSVF` line is allowed. For example:

```
&CSVF UVWFILE ='my_velocity_field.csv',
```

```
TMPFILE='my_temperature_field.csv'
SPECFILE='my_species_field.csv' /
```

If multiple meshes are involved, enter the name of the first .csv file only. The file must end in “1.csv”. FDS will increment the mesh number to the total number of meshes. For example, if you had 16 meshes you can use

```
&CSVF UVWFILE='CHID_uvw_t1_m1.csv',
      TMPFILE='CHID_tmp_t1_m1.csv',
      SPECFILE='CHID_spec_t1_m1.csv' /
```

and FDS will read, for example, CHID_uvw_t1_m1.csv through CHID_uvw_t1_m16.csv in the order of the MESH lines (note that a MULT line for the meshes may also be used).

20.6 Unfreezing the Initial Flow Field

It may be useful to allow the solid phase to respond to an established flow field at the start of a simulation. This could be accomplished with the normal restart functionality; however, this limits the number of changes that can be made to the model before a restart is made. The UNFREEZE_TIME parameter on the MISC line can be used for this purpose. When set, the initial flow field is frozen with the options FREEZE_VELOCITY and SOLID_PHASE_ONLY set until the simulation time exceeds UNFREEZE_TIME.

20.7 Gravity

By default, gravity points in the negative z direction, or more simply, downward. However, to change the direction of gravity to model a sloping roof or tunnel, for example, specify the gravity vector on the MISC line with a triplet of numbers of the form $GVEC=0., 0., -9.81$, with units of m/s^2 . This is the default, but it can be changed to be any direction.

There are a few special applications where you might want to vary the gravity vector as a function of time or as a function of the first spatial coordinate, x . For example, on board space craft, small motions can cause temporal changes in the normally zero level of gravity, an effect known as “g-jitter.” More commonly, in tunnel fire simulations, it is sometimes convenient to change the direction of gravity to mimic the change in slope. The slope of the tunnel might change as you travel through it; thus, you can tell FDS where to redirect gravity. For either a spatially or temporally varying direction and/or magnitude of gravity, do the following. First, on the MISC line, set the three components of gravity, GVEC, to some “base” state like $GVEC=1., 1., 1.$, which gives you the flexibility to vary all three components. Next, designate “ramps” for the individual components, RAMP_GX, RAMP_GY, and RAMP_GZ, all of which are specified on the MISC line. There is more discussion of RAMPS in Sec. 11, but for now you can use the following as a simple template to follow:

```
&MISC GVEC=1., 0., 1., RAMP_GX='x-ramp', RAMP_GZ='z-ramp' /

&RAMP ID='x-ramp', X= 0., F=0.0 /
&RAMP ID='x-ramp', X= 50., F=0.0 /
&RAMP ID='x-ramp', X= 51., F=-0.49 /
&RAMP ID='x-ramp', X=100., F=-0.49 /

&RAMP ID='z-ramp', X= 0., F=-9.81 /
&RAMP ID='z-ramp', X= 50., F=-9.81 /
&RAMP ID='z-ramp', X= 51., F=-9.80 /
```

```
&RAMP ID='z-ramp', X=100., F=-9.80 /
```

Note that both the x and z components of gravity are functions of x . FDS has been programmed to only allow variation in the x coordinate. Note also that F is just a multiplier of the “base” gravity vector components, given by $GVEC$. This is why using the number 1 is convenient – it allows you to specify the gravity components on the `RAMP` lines directly. The effect of these lines is to model the first 50 m of a tunnel without a slope, but the second 50 m with a 5 % slope upwards. Note that the angle from vertical of the gravity vector due to a 5 % slope is $\tan^{-1} 0.05 = 2.86^\circ$ and that 0.49 and 9.80 are equal to the magnitude of the gravity vector, 9.81 m/s^2 , multiplied by the sine and cosine of 2.86° , respectively. To check your math, the square root of the sum of the squares of the gravity components ought to equal 9.81. Notice in this case that the y direction has been left out because there is no y variation in the gravity vector. To vary the direction and/or magnitude of gravity in time, follow the same procedure but replace the x in the `RAMP` lines with a T .

Note that in a case with sprinklers, changing $GVEC$ will change how droplets move in the gas but not how droplets move on solid surfaces. On solid surfaces droplet movement will always consider down to be the negative- z direction.

Chapter 21

Pressure

Normally, you need not set any parameters related to the solution of the Poisson equation for pressure. However, there are circumstances when you might need to change default numerical values. This is done through the `PRES` namelist group (Table 23.23).

A unique feature of FDS, which distinguishes it from other CFD models, is that it employs a low Mach number approximation of the Navier-Stokes equations. For low speed flow simulations, it can be assumed that sound waves and pressure disturbances travel infinitely fast, rather than at the speed of sound, which is approximately 340 m/s at ambient temperature and pressure. Typical compartment fires induce flows of a few tens of meters per second, much less than the sound speed. However, for a simulation of a fire within a 1000 m tunnel, say, the propagation time of disturbances is a few seconds. If the tunnel has forced flow at one end and an opening at the other, it would take a few seconds for the pressure pulse from the fan to reach the opposite end. However, in the low Mach number formulation, the pulse reaches the other end instantaneously.

In FDS, the pressure, $p(\mathbf{x}, t)$, a function of space and time, is decomposed into a “background” component, $\bar{p}(z, t)$, and a perturbation, $\tilde{p}(\mathbf{x}, t)$. The background pressure is a function only of time and the vertical spatial coordinate, z , which accounts for the ambient stratification of the atmosphere. For a typical compartment fire simulation that is open to the atmosphere, \bar{p} is essentially constant. The perturbation pressure drives the fire-induced flow field.

The equation for \tilde{p} is an elliptic partial differential equation (PDE) that is formed by taking the divergence of the momentum equation, which contains the forcing term $\nabla \tilde{p} / \rho$. Ideally, this PDE would be of the form:

$$\nabla \cdot \left(\frac{1}{\rho} \right) \nabla \tilde{p} = \dots \quad (21.1)$$

The discretized form of this PDE, that is, its approximation on a numerical grid, is a linear system of equations, $\mathbf{A} \tilde{\mathbf{p}} = \mathbf{b}$, where \mathbf{A} is a sparse (i.e. mostly zeros) matrix, $\tilde{\mathbf{p}}$ is a vector whose number of elements equals the number of grid cells in the computational domain, and \mathbf{b} is a vector representing the various discretized terms of the momentum equation. The CPU and memory requirements necessary to solve this huge system of equations could easily dwarf all other parts of the simulation. There are two remedies to this problem. First, the pressure field is solved within each individual mesh of the domain, and the individual pressure fields are “stitched” together using multiple iterations of the solver. Second, the matrix \mathbf{A} , representing the operator $\nabla \cdot (1/\rho) \nabla$ is simplified by replacing the variable density $\rho(\mathbf{x}, t)$ by its ambient value so that the matrix \mathbf{A} need not be recomputed at each time step, and a very fast and efficient solver using fast Fourier transforms (FFT) can be used.

In the next two sections, these simplifications of the Poisson equation will be discussed, along with parameters that you might need to set to increase the accuracy of the solution for certain applications.

21.1 Accuracy of the Pressure Solver

For a single mesh, the solution of the simplified (i.e. separable) form of the Poisson equation for the pressure can be solved relatively quickly and accurately (i.e. to machine precision) using an FFT-based solver. However, for multiple meshes of varying grid resolution and configuration, it is not possible to use the fast solver for the global solution of the Poisson equation. Instead, the fast solver is employed mesh by mesh in parallel, and the pressure field on each mesh is forced to match at the mesh boundaries through repeated solves on the individual meshes. It is prohibitively time-consuming to iterate the global pressure solution to the same level of accuracy as the local pressure solutions. The inaccuracy of the global pressure solution is manifested in a slight mismatch in the normal component of velocity at the mesh boundaries. An error tolerance is specified for this mismatch in velocity. These small errors in the normal component of velocity also appear within an individual mesh at solid, internal boundaries. The fast, FFT-based solver can only enforce exact no-flux conditions at exterior boundaries of the mesh.

If either the error in the normal component of the velocity at a mesh interface or at a solid boundary is large, you can reduce it by making more than the default number of calls to the pressure solver at each time step. To do so, specify `VELOCITY_TOLERANCE` on the `PRES` line to be the maximum allowable normal velocity component on the solid boundary or the largest error at a mesh interface. It is in units of m/s. If you set this, experiment with different values, and monitor the number of pressure iterations required at each time step to achieve your desired tolerance. The default value is $\delta x/2$, where δx is the characteristic grid cell size. The number of iterations are written out to the file `CHID.out`. If you use a value that is too small, the CPU time required might be prohibitive. The maximum number of iterations for each half of the time step is given by `MAX_PRESSURE_ITERATIONS`, also on the `PRES` line. Its default value is 10.

To verify the accuracy of the pressure solution on each mesh, set `CHECK_POISSON` to `T` on the `PRES` line, instructing FDS to check that the left-hand and right-hand sides of the Poisson equation are equivalent (see the FDS Tech Guide [3]). The error is printed to the `CHID.out` file. Note that this verifies the accuracy of the *local*, not the global, pressure solution. To check the accuracy of the global pressure solution, see Sec. 21.3.

It is possible for the pressure iteration scheme to seem “stuck”—the solution convergence rate may slow dramatically and further iterations are only consuming cpu resources without improving the solution. You can tell FDS to break out of these situations prior to hitting the maximum number of iterations by setting `SUSPEND_PRESSURE_ITERATIONS=T` on the `PRES` line. FDS will then suspend the pressure iterations if the error does not drop below `ITERATION_SUSPEND_FACTOR` (0.95, by default) of its previous value.

21.1.1 Optional Pressure Solvers

The default Poisson solver in FDS (`SOLVER='FFT'` on the `PRES` line) is based on the package of linear algebra routines called CrayFishpak. However, in certain circumstances you may need to use one of several alternatives that is based on the Intel® MKL Sparse Cluster Solver or the HYPRE solver library from Lawrence Livermore National Laboratories (LLNL) [78].

- `SOLVER='ULMAT'` This solver is useful when the computational domain has relatively small, sealed cavities enclosed with thin (i.e. zero cell thick) walls. The small error in the normal component of velocity that you incur with the FFT solver can lead to unphysical changes in temperature, density and pressure that may eventually cause the calculation to become numerically unstable. For example, suppose you have hollow steel columns or hollow aluminum ducts with thin walls and you want to simulate the heat penetration within these spaces.

With the `ULMAT` solver, the unknown values of the pressure live only in gas-phase cells, allowing for the normal components of velocity at a solid surface to be computed exactly with no penetration error. Strictly speaking, in this mode, FDS is no longer using an “immersed boundary method” for flow

obstructions—the pressure solution is now “unstructured,” hence the `u` in its name. This solver does not guarantee that the normal component of velocity matches perfectly at mesh boundaries, and the iterative procedure used by the default `'FFT'` solver is still used to drive the mesh interface velocity normals closer together. This solver option does allow for refinement at mesh boundaries and for internal sealed regions within the domain for which additional matrices must be computed.

The `ULMAT` solver uses a substantial amount of memory per mesh. Its use should be limited to meshes of at most 150,000 cells. This number depends on the amount of random access memory (RAM) that you can devote to each MPI process, and you might want to experiment with larger meshes if you have, say, tens of gigabytes of RAM per MPI process.

- `SOLVER='ULMAT HYPRE'` This solver uses the same strategy as `ULMAT` but the solver is implemented via LLNL’s `HYPRE` library [78]. `ULMAT HYPRE` uses the algebraic multi-grid (AMG) preconditioned conjugate gradient (PCG) solver. This solver does not require LU decomposition and storage of a dense matrix, so the memory requirements are much less than the `MKL` solver. Hence, `ULMAT HYPRE` may be considered as an option for large mesh blocks (larger than 150,000 cells). Note, however, that for small mesh blocks `HYPRE` is somewhat slower than `MKL`.
- `SOLVER='GLMAT'` This solver is for non-overlapping, non-stretched meshes at the same refinement level covering a single connected domain, i.e., a large rectangular box. With this solver, the pressure in both solid and gas cells is computed using an immersed boundary method for flow obstructions, the same as the default `'FFT'` solver. This mode produces the exact same pressure solution as the `'FFT'` solver would if the domain were one single mesh. That is, velocity errors at any mesh boundaries are eliminated. Note that currently a single discretization matrix is built for all gas cells defined on the model, therefore the solver is not meant to be used in cases where there are internal sealed regions within the domain.

The `GLMAT` solver is used mainly for testing and diagnostics. Because it solves the pressure over the entire computational domain, it is limited to about 1 million cells total unless your computer has an extraordinary amount of RAM.

- `SOLVER='UGLMAT'` This solver is for non-overlapping, non-stretched meshes at the same refinement level. Like the solver `'ULMAT'`, the unknown values of the pressure live only in gas-phase cells, allowing for the normal components of velocity at a solid surface to be computed exactly (no velocity penetration error). In addition, this solver ensures that the normal component of velocity matches perfectly at mesh boundaries. It is limited to about 1 million grid cells, and (3) it is the most CPU-intensive of the alternative solvers.
- `SOLVER='UGLMAT HYPRE'` Performs the same global solution as `'UGLMAT'` calling the `HYPRE` algebraic multi-grid (AMG) preconditioned conjugate gradient (PCG) solver. It is also CPU-intensive, but does not have the global mesh size limitation of the previous method.

Because the `'ULMAT'`, `'GLMAT'`, `'UGLMAT'` alternative methods are based on the Intel® `MKL` parallel LU decomposition solver, they require computation and storage of global factorization matrices. Depending on the size of the problem, these operations can be expensive in terms of both CPU time and memory. It has been found suitable for problems with up to a million cells on a multi-core workstation with 64 GB of RAM.

Table 21.1: Summary of available pressure solvers.

Solver	Allows				Handling Errors		
	Refinement ¹	Stretching	Add/Remove ²	Zones	Velocity @ Mesh ³	Velocity @ Solid ⁴	Inseparability ⁵
FFT (default)	✓	2 directions	✓	✓	iterative	iterative	iterative
ULMAT	✓	✓	✓	✓	iterative	exact	iterative
ULMAT HYPRE ⁶	✓	✓	✓	✓	iterative	exact	iterative
GLMAT	×	×	✓	✓	exact	iterative	iterative
UGLMAT	×	×	✓	✓	exact	exact	iterative
UGLMAT HYPRE ⁷	×	×	✓	✓	exact	exact	iterative

Notes:

¹ “Refinement” refers to allowing a change in mesh resolution for neighboring meshes.

² “Add/Remove” refers to the ability of the solver to handle changes in geometry during the course of the simulation, either by adding or removing OBST or GEOM via CTRL functions.

³ “Velocity @ Mesh” refers to errors in the normal component of velocity at mesh boundaries, so-called “interpolated” boundaries in FDS. For inexact solvers, this error may be controlled using the parameter `VELOCITY_TOLERANCE`. Mass conservation errors are proportional to the velocity tolerance. Solvers listed as “exact” are *global* solvers. “Global” refers to a global matrix solution, one matrix for the whole domain, as opposed to one matrix per mesh. Thus, there are no mesh boundary errors for this type of solver.

⁴ “Velocity @ Solid” refers to errors in the normal component of velocity at solid boundaries, so-called “penetration” errors that are present in *immersed boundary methods* and lead to mass conservation errors. These errors are particularly important to control for tightly sealed pressure zones. For inexact solvers, this error may be controlled using the parameter `VELOCITY_TOLERANCE`. Mass conservation errors are proportional to the velocity tolerance. Solvers listed as “exact” are *unstructured* solvers. “Unstructured” means the solid boundaries (e.g., OBST boundaries) are treated as domain boundaries for the pressure Poisson equation. Unstructured solvers are the opposite of *immersed boundary methods*.

⁵ “Inseparability” refers to handling the error compared to the solution of the *inseparable* Poisson equation incurred by lagging the pressure in the baroclinic term. This error may be controlled using the parameter `PRESSURE_TOLERANCE`.

⁶ ULMAT HYPRE uses the HYPRE library developed at Lawrence Livermore National Laboratories (LLNL). Use this variant of ULMAT if the Intel MKL library is not available *or* if MKL runs out of memory or is too slow due to memory issues with large mesh blocks. Our implementation of HYPRE uses an algebraic multi-grid (AMG) preconditioned conjugate gradient (PCG) method, which does not require excessive amounts of memory.

⁷ Use this variant of UGLMAT for cases where the total number of cells is greater than 1 million.

21.1.2 Example Case: Pressure_Solver/duct_flow

To demonstrate how to improve the accuracy of the pressure solver, consider the flow of air through a square duct that crosses several meshes. In the sample input file, `duct_flow.fds`, air is pushed through a 1 m² duct at 1 m/s. With no thermal expansion, the volume flow into the duct ought to equal the volume flow out of the duct. Figure 21.1 displays the computed inflow and outflow as a function of time and the number of pressure iterations required. The default pressure solution strategy uses block-wise direct FFT solves combined with a direct forcing immersed boundary method to model flow obstructions. Hence, there are two sources of pressure error: one at a mesh interface and one at a solid boundary. The default pressure iteration scheme tries to drive both sources of error to the specified (or default) velocity tolerance. In the `duct_flow` case, the `VELOCITY_TOLERANCE` has been set to 0.001 m/s with `MAX_PRESSURE_ITERATIONS` set to 1000 and the grid cell size is 0.2 m. For the default scheme (FFT + IBM), the outflow does not match the inflow exactly because of inaccuracies at the solid and mesh boundaries. We have also included the results using the 'UGLMAT', 'UGLMAT HYPRE' solvers with an unstructured pressure matrix, which allows the flow solver to enforce the impermeability condition (zero normal component of velocity) at a solid boundary.

```
&PRES SOLVER='UGLMAT' /
```

As seen in Fig. 21.1, the volume flow matches perfectly and the number of required pressure iterations is one. Whether this pressure solver strategy is most efficient depends on the problem and the required error tolerance. In this particular case, the 'UGLMAT' case runs about 25 % faster than the default FFT case, and it is more accurate. The reason is that a tight tolerance has been specified and the FFT solver is forced to iterate tens or hundreds of times per time step. When run with default error tolerances, the 'FFT' solver is faster, but less accurate, than the alternative solvers.

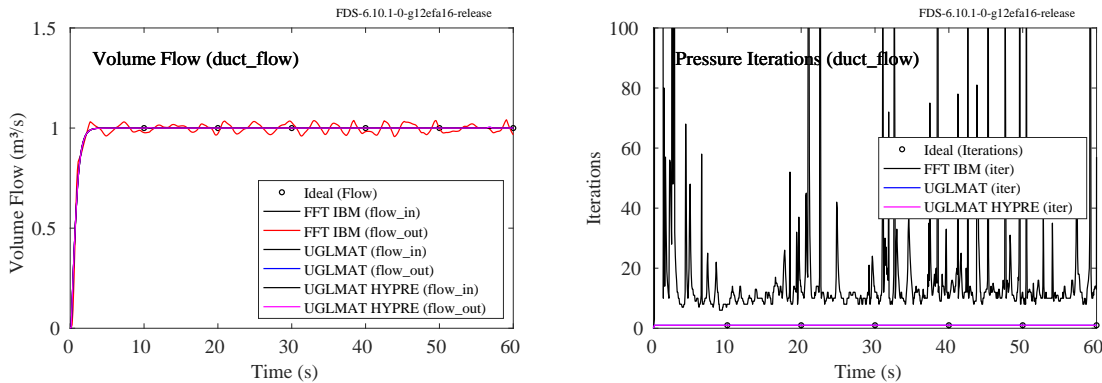


Figure 21.1: (Left) Volume flow into and out of a square duct. (Right) The number of pressure iterations as a function of time.

21.1.3 Example Case: Pressure_Solver/dancing_eddies

In this example, air is pushed through a 30 cm long, two-dimensional channel at 0.5 m/s. A plate obstruction normal to the flow creates a Karman vortex street. The computational domain is divided into 4 meshes. Six simulations are performed: one in which the `VELOCITY_TOLERANCE` is set to the default value, one in which it is set to a relatively small value, two using 'ULMAT' and 'ULMAT HYPRE', and two where 'UGLMAT' and 'UGLMAT HYPRE' are used. Figure 21.2 shows the downstream pressure histories for these cases compared to a simulation that uses only one mesh. The case with the tighter tolerance produces a better match to the

single mesh solution, but at a higher computational cost—about a factor of 2.5 . The `SOLVER='UGLMAT'` case takes just a little longer than the default—by a factor of 1.25—but the error is machine precision. These cases used MPI for domain decomposition, but only a single OpenMP thread. Note, however, that the `'UGLMAT'` solver is threaded, whereas currently the `'FFT'` solver (FDS default) is not.

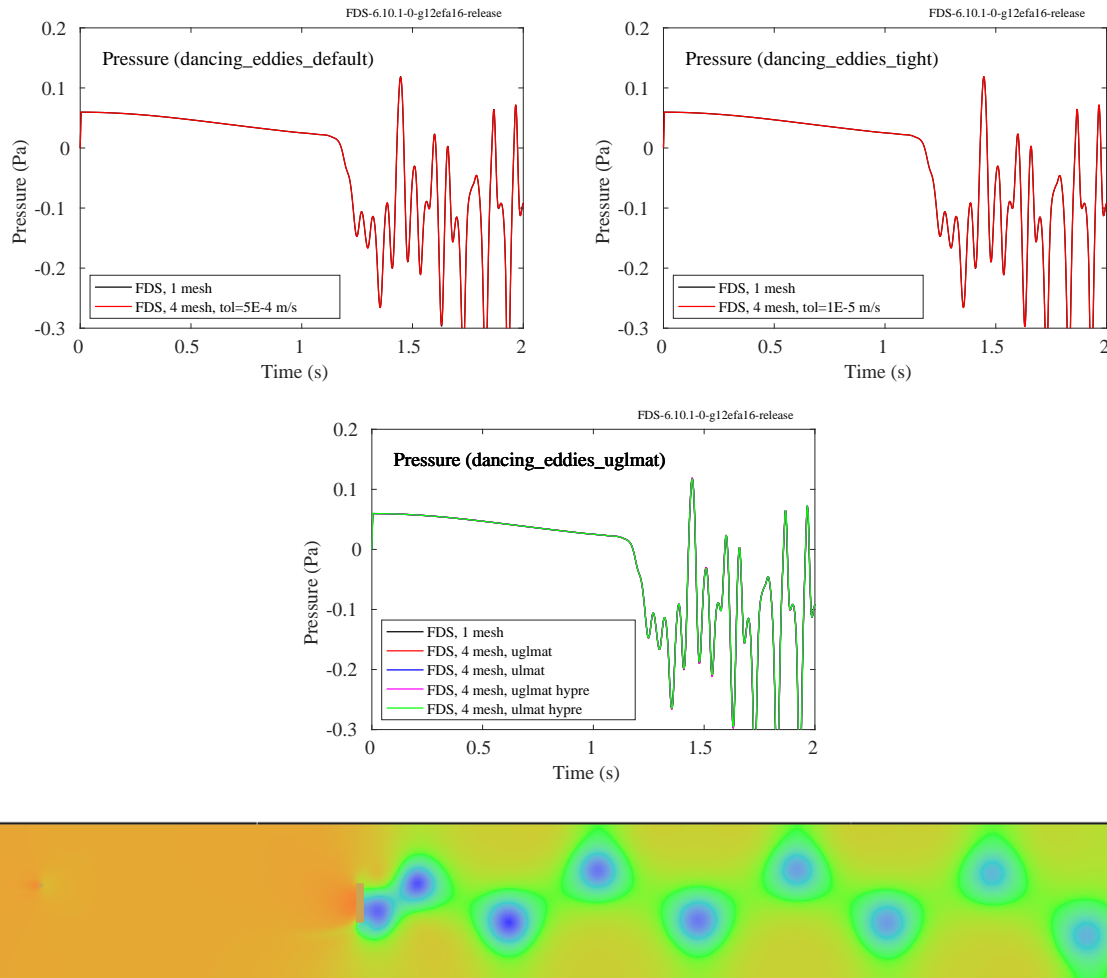


Figure 21.2: (Top) Comparison of pressure traces in the channel for three different settings of `VELOCITY_TOLERANCE`, the default value (upper-left), a tighter tolerance (upper-right), machine precision from the `'UGLMAT'` solver with a single iteration (middle). (Bottom) A contour plot of the pressure after 2 s with the default tolerance.

The case called `dancing_eddies_tight` is rerun without the special pressure pre-conditioner. Figure 21.3 shows the reduction in the number of pressure iterations (left) and the CPU time (right).

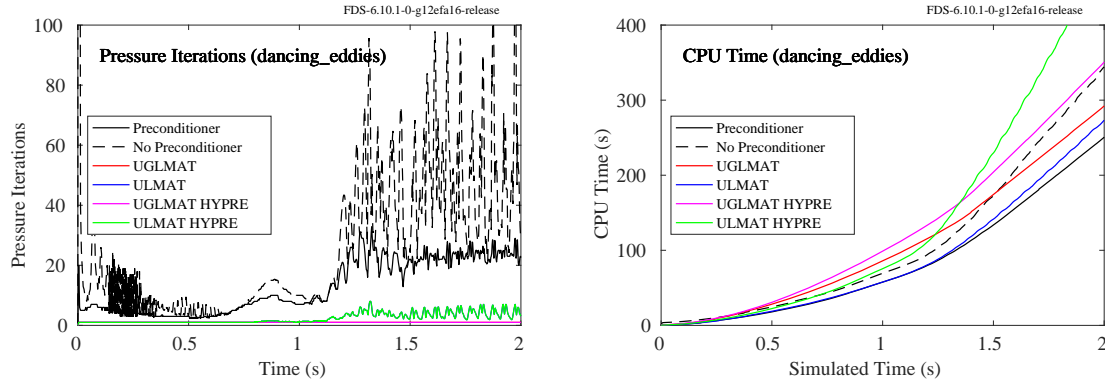


Figure 21.3: The number of pressure iterations (left) and the total CPU time (right) for preconditioned and non-preconditioned pressure solvers.

21.1.4 Example Case: Random Obstructions

In this example, many different types of obstructions are randomly positioned within a domain split into 16 meshes. Some of the obstructions are created or removed at certain times. Sources of heat are added to move the gases about. The grid cell size is 0.025 m and the default normal velocity error tolerance is half that, or 0.0125 m. Figure 21.4 displays the maximum error and the number of iterations of the pressure solver. The spikes in the number of iterations correspond to the creation or removal of obstructions. More iterations are necessary to drive down the error when a sudden change occurs in the flow field.

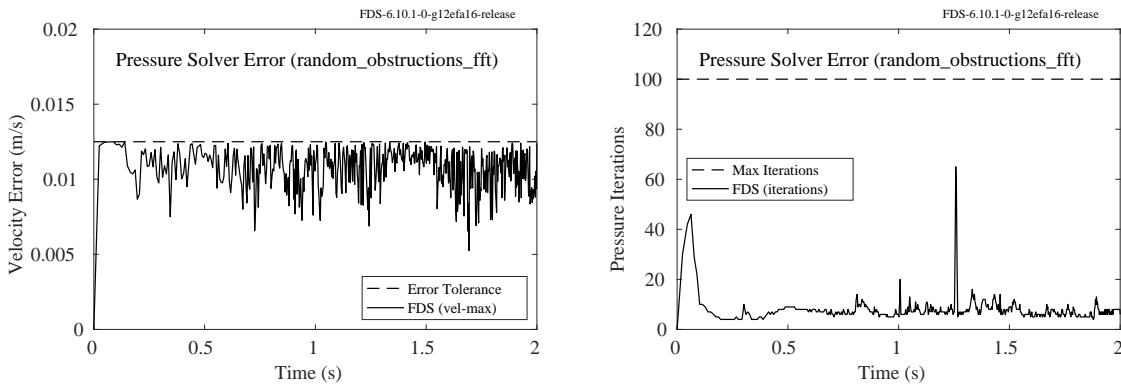


Figure 21.4: The number of pressure iterations (left) and maximum velocity error (right).

21.2 Baroclinic Vorticity

The second challenge in solving the Poisson equation for pressure is that the matrix of the linear system of equations that is formed when discretizing it does not have constant coefficients because the density, $\rho(\mathbf{x}, t)$, changes with each new time step. To get around this problem, the pressure term in the momentum equation is decomposed as follows:

$$\frac{1}{\rho} \nabla \tilde{p} = \nabla \left(\frac{\tilde{p}}{\rho} \right) - \tilde{p} \nabla \left(\frac{1}{\rho} \right) \quad (21.2)$$

which leads to a Poisson equation that can be solved efficiently:

$$\nabla^2 \left(\frac{\tilde{p}^{n,k}}{\rho^n} \right) = \nabla \cdot \tilde{p}^{n,k-1} \nabla \left(\frac{1}{\rho^n} \right) + \dots \quad (21.3)$$

The superscript n represents the time step. The superscript k indicates that this equation is solved multiple times per time step, and k indicates the iteration number. With each iteration, the old and new values of \tilde{p} are driven closer together. The iterations continue¹ until the maximum value of the error term

$$\varepsilon^k = \max_{ijk} \left| \nabla \cdot \left(\frac{1}{\rho^n} \right) \nabla \tilde{p}^{n,k} - \nabla^2 \left(\frac{\tilde{p}^{n,k}}{\rho^n} \right) + \nabla \cdot \tilde{p}^{n,k-1} \nabla \left(\frac{1}{\rho^n} \right) \right| \quad (21.4)$$

drops below the value of `PRESSURE_TOLERANCE` which is specified on the `PRES` line. Its default value is $20/\delta x^2 \text{ s}^{-2}$, where δx is the characteristic grid cell size.

The lagged pressure term on the right hand side of Eq. (21.3) is sometimes referred to as the baroclinic torque, and it is responsible for generating vorticity due to the non-alignment of pressure and density gradients. In versions of FDS prior to 6, the inclusion of the baroclinic torque term was found to sometimes cause numerical instabilities. If it is suspected that the term is responsible for numerical problems, it can be removed by setting `BAROCLINIC=F` on the `MISC` line. For example, in the simple helium plume test case below, neglecting the baroclinic torque changes the puffing behavior noticeably. In other applications, however, its effect is less significant. For further discussion of its effect, see Ref. [79].

Example Case: `Flowfields/helium_2d_isothermal`

This case demonstrates the use of baroclinic correction for an axially-symmetric helium plume. Note that the governing equations solved in FDS are written in terms of a three dimensional Cartesian coordinate system. However, a two dimensional Cartesian or two dimensional cylindrical (axially-symmetric) calculation can be performed by setting the number of cells in the y direction to 1. An example of an axially-symmetric helium plume is shown in Fig. 21.5.

¹Keep in mind that the pressure equation iterations continue until three criteria are satisfied. The first deals with the decomposition of the pressure term, the second deals with the normal component of velocity at internal solid surfaces, and the third deals with the mismatch of normal velocity components at mesh interfaces.

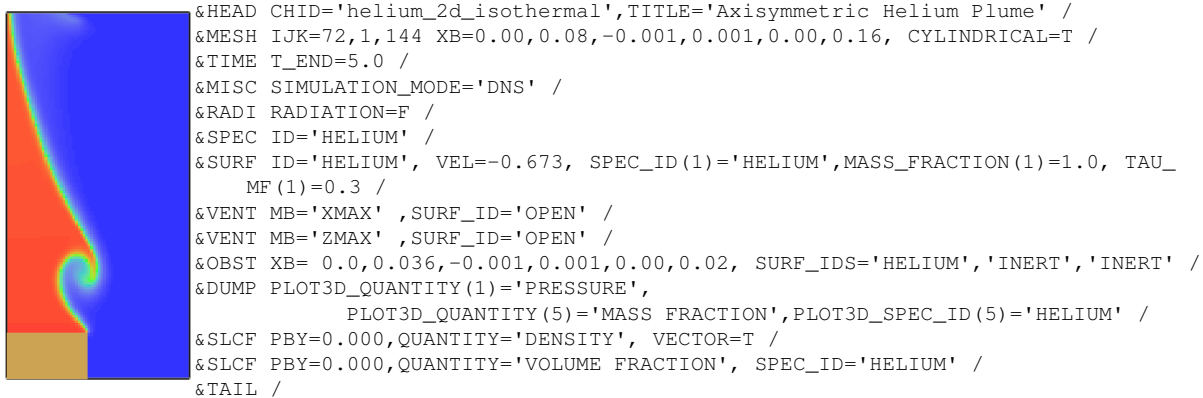


Figure 21.5: Simulation of a helium plume.

21.3 Pressure Considerations in Long Tunnels

A common application of FDS is tunnel fires, but simulations of fires in long, relatively tight tunnels that are made up of multiple meshes can exhibit spurious fluctuations in the pressure field that can lead to numerical instabilities. If a numerical instability occurs in a simulation involving a tunnel, you might try the following remedies, which are listed here in order of effectiveness:

1. If (1) the tunnel is made up of multiple meshes that abut end to end along the x-direction, (2) there are no other meshes outside of the tunnel, (3) there are no OPEN boundaries on the top, bottom or sides of the tunnel, and (4) the number of MPI processes is equal to the number of meshes; set `TUNNEL_PRECONDITIONER=T` on the `PRES` line. This option instructs FDS to solve a 1-D pressure equation that spans the entire tunnel. This 1-D solution can be thought of as an average pressure field which is added to the 3-D pressure fields that are solved for using the default FFT-based solver on each mesh. The 1-D global solution accelerates the convergence of the overall 3-D pressure field on the entire domain.
2. You can increase the value of `MAX_PRESSURE_ITERATIONS` which has a default value of 10 in most cases; 20 when the `TUNNEL_PRECONDITIONER` is invoked. If you experience a numerical instability in a tunnel scenario, try setting the `MAX_PRESSURE_ITERATIONS` to 50 as a starting point, and monitor the number of pressure iterations per time step in the `CHID.out` file.
3. If you are using multiple meshes, reduce the value of `VELOCITY_TOLERANCE` on the `PRES` line to force a tighter match of velocities at the mesh boundaries. The default value of `VELOCITY_TOLERANCE` (with units of m/s) is $\delta x/2$. If you set `TUNNEL_PRECONDITIONER=T`, there is less of a need to decrease the `VELOCITY_TOLERANCE` from its default value, but it still may be necessary to increase the `MAX_PRESSURE_ITERATIONS` to account for occasional excursions of pressure.
4. Create OPEN vents at various points along the length of the tunnel, near or at the floor. This models natural leakage in the tunnel, and alleviates, to some extent, wild oscillations in pressure. Note that even if the pressure equation is solved perfectly over the entire domain, the low Mach number assumption will still lead to fictitiously large fluctuations in pressure. The OPEN boundaries can help relieve these pressure oscillations. If you do choose to create OPEN boundaries along the length of the tunnel, you should *not* set `TUNNEL_PRECONDITIONER` to T

5. Reduce the value of `PRESSURE_TOLERANCE` on the `PRES` line to alleviate the mismatch between old and new pressure fields. The default value is $20/\delta x^2 \text{ s}^{-2}$. A good value to try would be 1/5 to 1/10 of the default value.
6. Set `SOLVER='GLMAT'` on the `PRES` line to strictly enforce the matching of normal velocity and pressure at mesh boundaries or `'UGLMAT'` to do what `'GLMAT'` does plus enforce zero normal velocities at solid obstructions. If you use these solvers, you need not set the various tolerances described above or use the `TUNNEL_PRECONDITIONER`. However, these solvers are very expensive, so it is best to try a simple case first to determine how much CPU time the pressure solver consumes.

Two simple test cases, `Pressure_Solver/tunnel_demo.fds` and `tunnel_demo_glmato.fds`, demonstrate some of the issues discussed in this section. The first case uses the default FFT-based pressure solver with the `TUNNEL_PRECONDITIONER` set to `T`; the second uses the `'GLMAT'` solver. An 8 MW fire is situated in a 4 m by 4 m by 128 m long tunnel that is sloped 10° , closed at the lower end and open at the upper end. The tunnel is divided into 8 meshes, all the same size, with a uniform grid of 20 cm. The cases run for a relatively short amount of time, and diagnostic files² containing information about the velocity and pressure errors are printed out. These files are generated by setting `VELOCITY_ERROR_FILE=T` on the `DUMP` line. The names of the files are `tunnel_demo_pressit.csv` and `tunnel_demo_glmato_pressit.csv`. A description of each column is as follows:

- `Time`: Simulation time (s)
- `Time Step`: The index of the time step
- `Iteration`: The index of the pressure iteration within the time step. Note that for a single time step, there are two series of pressure iterations, one for the predictor phase, and one for the corrector.
- `Total`: The cumulative index of pressure iterations for the entire simulation
- `Mesh`: The mesh where the maximum velocity error occurs
- `I,J,K`: Cell indices where the maximum velocity error occurs
- `velocity Error`: The maximum absolute value of the difference between normal velocity components at mesh interfaces or at solid internal boundaries
- `Mesh,I,J,K`: The mesh and cell indices where the maximum pressure error occurs
- `Pressure Error`: The quantity shown in Eq. (21.4) in units of s^{-2}

Figure 21.6 shows the velocity and pressure errors for the two cases over a short span of the simulation. For each quantity, there is a default error tolerance which may or may not be reached depending on whether or not the `MAX_PRESSURE_ITERATIONS` (default 10) has been reached, or whether or not the error is reduced by at least 25 % over the previous iteration. In some situations, the error decreases slowly, and these criteria have been added to avoid excessive iterations that lead to little improvement in accuracy. The errors shown in Fig. 21.6 are relatively large because fires in long, closed enclosures like tunnels can generate large fluctuations in pressure that challenge both the multiple mesh capability and the pressure solving algorithm discussed above. Note that the simulation using the `'GLMAT'` solver has no velocity error in this case because it solves the Poisson equation directly over the entire domain, not mesh by mesh. However, this simulation still requires iterations of the pressure solver to drive the solution closer to the true solution of the inseparable Poisson equation.

²Caution: the velocity error file can be quite large. Use it only for relatively short simulations only.

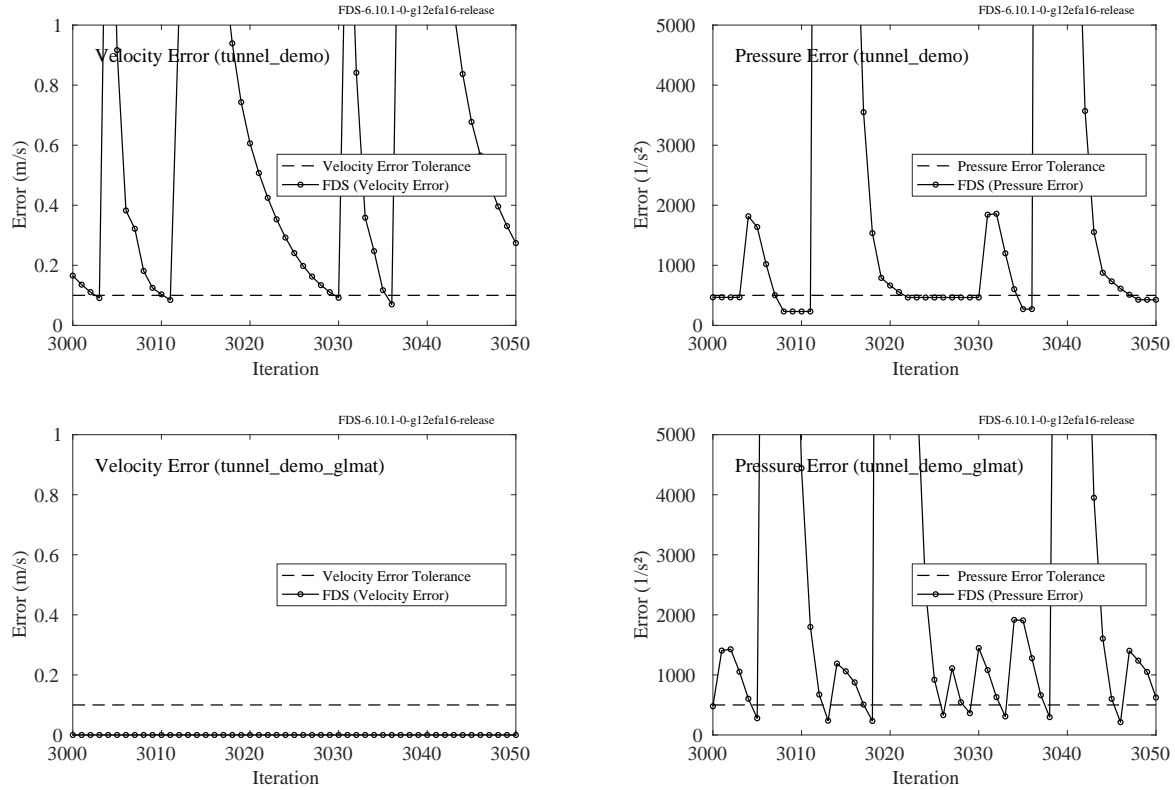


Figure 21.6: Reduction in velocity and pressure error due to iteration of the pressure solver. The top two plots are the result of the FFT-based solver; the lower two 'GLMAT'.

Special Case: Pressure Drop in Long Tunnels

This section describes a set of cases to extract the implied friction factor from an FDS simulation of flow down a long tunnel (Validation/Moody_Chart/FDS_Input_Files/tunnel_pressure_drop series). Note that a more complete mapping of the Moody Chart is provided in the FDS Verification Guide [4], but that series uses a mean pressure gradient to force the flow, which is not a typical user case. Here the flow is forced from a VENT and the axial pressure profile is formed from planar averages DEVCS. The tunnel is 1.6 km (approximately 1 mi) long with a square cross-section 10 m by 10 m for Cases A, B, C, and D. The cross section for Case E is 12.8 m in width by 5 m in height, giving a hydraulic diameter of 7.2 m. Air is forced from the entrance at 2, 4, or 10 m/s for sand-grain roughness heights of 0.0001, 0.01, or 0.1 m. Two grid resolutions are used, with 10 or 20 uniform cells spanning the tunnel height. Stream-wise grid resolution is twice the wall-normal resolution. The case matrix and friction factor results are given in Table 21.2. The target friction factor is taken from the Colebrook equation [80]. FDS results for the 10 and 20 vertical cell cases are also shown along with the maximum relative error as compared to the Colebrook value. The pressure profiles are shown in Fig. 21.7.

While reasonable grid resolution is important, the wall models in FDS are capable of giving the correct mean wall stress even at rather coarse resolution. For example, the y^+ for Case C with 10 cells is, in fact, 5000 (this is the location of the middle of the first grid cell divided by the roughness height). Also, note that a roughness of 0.1 mm is used here only for completeness in testing the code. While this is indeed the value one finds in the literature for the roughness of concrete, a real-world tunnel may have geometric features along the walls that are unresolved by the grid and may act as roughness elements. These elements should

Table 21.2: Friction factors for `tunnel_pressure_drop` cases.

Case	Velocity (m/s)	Roughness (m)	Hydraulic Dia. (m)	f Colebrook	f FDS 10	f FDS 20	Max Rel. Error (%)
A	2	0.0001	10	0.0114	0.0125	0.0118	10
B	2	0.1	10	0.0379	0.0358	0.0348	8.3
C	10	0.0001	10	0.00932	0.00914	0.00908	2.6
D	10	0.1	10	0.0379	0.0354	0.035	7.7
E	4	0.01	7.2	0.0214	0.0196	0.0204	8.8

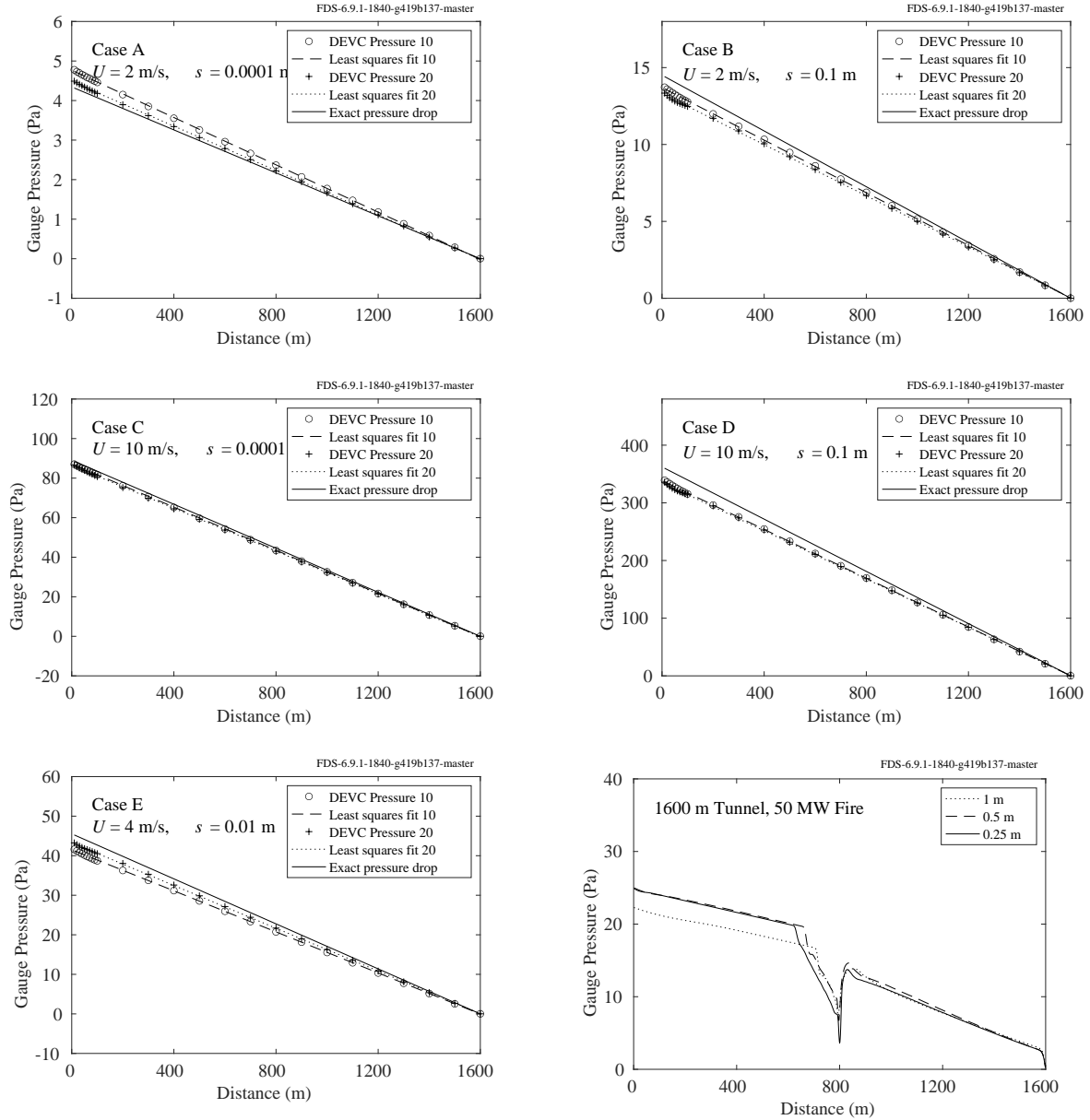


Figure 21.7: Tunnel pressure drop for cases listed in Table 21.2.

be considered when specifying the roughness in FDS.

The lower right plot in Fig. 21.7 displays the pressure drop in Tunnel B (2 m/s flow speed, 0.1 m wall roughness) which now contains a hypothetical 50 MW fire at its midpoint. The grid resolution in the vertical direction is listed in the legend of the plots. For this case, the pressure is evaluated at mid-height along the

tunnel centerline. The relatively rapid drop in pressure at the open end of the tunnel ($x = 1600$ m) is due to the fact that the pressure is assumed to be ambient at an open boundary.

21.4 Pressure Considerations in Stairwells

Another common fire protection application is smoke movement and pressurization within stairwells. A stairwell can be thought of as a vertical tunnel, with similar issues related to the pressure solver. Unlike tunnels, there is no special algorithm to increase the accuracy and stability of simulations within a stairwell, but there are some steps you can take to improve the simulation:

1. As with any tall building, the pressure stratification of the atmosphere can have an impact on the flow field and ventilation. A useful output quantity to include in any building simulation is 'ABSOLUTE PRESSURE'. This displays the background pressure field along with the relatively small perturbation induced by changes in ventilation or a fire. You typically do not see this background pressure field when you invoke the output quantity `PRESSURE`, which displays only the perturbation to the background.
2. It may be necessary to tighten the tolerance of the pressure solver when simulating a stairwell scenario. If you are using multiple meshes, reduce the value of `VELOCITY_TOLERANCE` on the `PRES` line to force a tighter match of velocities at the mesh boundaries. The default value of `VELOCITY_TOLERANCE` (with units of m/s) is $\delta x/2$. Increase the `MAX_PRESSURE_ITERATIONS` beyond the default value of 10. Experiment with the optional `SOLVER='GLMAT'` on the `PRES` line to strictly enforce the matching of normal velocity and pressure at mesh boundaries or 'UGLMAT' to do what 'GLMAT' does plus enforce zero normal velocities at solid obstructions. If you use these solvers, you need not set the various tolerances described above, but keep in mind that these solvers can be expensive. Try a simple case first to determine how much CPU time the pressure solver consumes.

A sample input file called `Pressure_Solver/stairwell.fds` demonstrates some of the important issues discussed in this chapter. The case consists of an eighteen story stairwell that opens to a lobby area. Doors are situated at various locations and are programmed to open when the pressure differential exceeds 20 Pa. A 1 m by 1 m hatch at the top of the stairwell draws air out at a rate of $0.5 \text{ m}^3/\text{s}$, triggering doors to open one after the other until a path is opened up between an open door in the lobby and the extract vent. The plot in Fig. 21.8 demonstrates that once the path is opened, the flow equilibrates to $0.5 \text{ m}^3/\text{s}$.

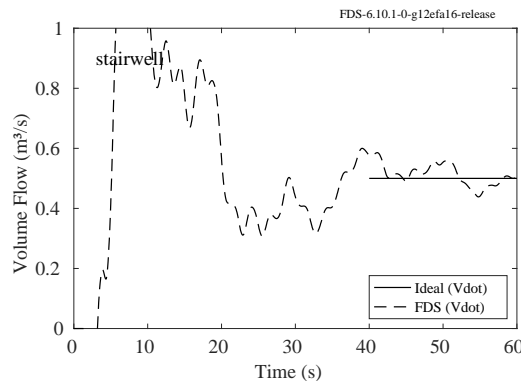


Figure 21.8: Results of the `stairwell` test case showing the volume flow of air through a tall stairwell.

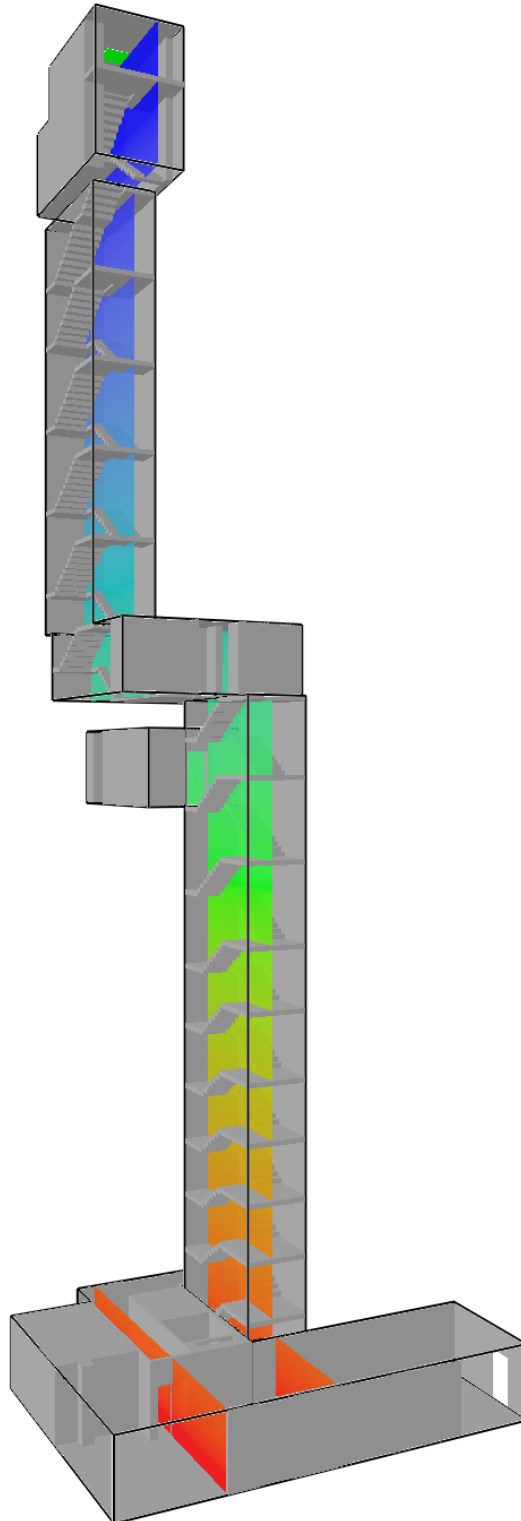


Figure 21.9: Smokeview rendering of a stairwell. The contour plot indicates the `ABSOLUTE PRESSURE`, which in this case ranges from approximately 101 600 Pa at the bottom to 101 000 Pa at the top. Ground level is approximately halfway up.

Chapter 22

Output

FDS has various types of output files that store computed data. Some of the files are in binary format and intended to be read and rendered by Smokeview. Some of the files are just comma-delimited text files. In most cases, you must explicitly declare the data to output. A considerable amount of the input file is usually devoted to output declarations.

- To control the frequency of data outputs, see Sec. [22.1](#).
- To record data at a given point or within a given volume, see Sec. [22.2](#).
- To record data inside solid boundaries, see Sec. [22.3](#).
- To visualize planar contours of scalar or vector quantities, see Sec. [22.4](#).
- To visualize solid surface quantities, see Sec. [22.5](#).
- To visualize 3-D contours of gas phase quantities, see Sec. [22.6](#).
- To visualize various scalar quantities over the entire domain, see Sec. [22.7](#).
- To visualize realistic smoke and fire, see Sec. [22.8](#).
- To visualize particles and droplets, see Sec. [22.9](#).
- For more information about particular output features, see Sec. [22.10](#).
- A tool for extracting data from contour and boundary output files is described in Sec. [22.11](#).
- A list of gas phase output quantities is given in Sec. [22.12](#).
- A list of solid phase output quantities is given in Sec. [22.13](#).
- A list of device-related output quantities is given in Sec. [22.14](#).
- A list of particle and droplet output quantities is given in Sec. [22.15](#).
- A list of HVAC output quantities is given in Sec. [22.16](#).

The namelist group `DUMP` contains parameters (Table [23.9](#)) that control the rate at which output files are written, and various other global parameters associated with output files. There can only be one `DUMP` line in the input file. By default, FDS outputs all result files to the current working directory. A separate output directory for binary files can be specified by the user by setting `RESULTS_DIR` on the `DUMP` line. A single text

file `CHID.smv` is always written with high level information about the case [2]. By default, the information is gathered in MPI rank 0 which writes the file. For computations involving parallel file systems and many MPI processes, the file can also be written in parallel using MPI I/O routines setting `SMV_PARALLEL_WRITE=T` in the `DUMP` line.

22.1 Controlling the Frequency of Output

Point device data, slice (contour) data, particle data, isosurface data, 3D smoke data, boundary data, solid phase profile data, and control function data are written to file every $(T_END - T_BEGIN) / NFRAMES$ seconds unless otherwise specified using the parameters listed in Table 22.1. These parameters are written on the `DUMP` line. You can also set `NFRAMES` to a value other than 1000, its default. The parameters named `DT_XXXX` specify the uniform time interval between data dumps. The parameters `RAMP_XXXX` allow you to specify exactly which times to write out. For example, if you want to write boundary files at 10, 20, and 60 s, add the following:

```
&DUMP ..., RAMP_BNDF='b-ramp' /
&RAMP ID='b-ramp', T=10 /
&RAMP ID='b-ramp', T=20 /
&RAMP ID='b-ramp', T=60 /
```

A few things to note:

- **SMOKE3D** If the simulation involves combustion, FDS automatically writes out Smoke3D files of soot density, heat release rate per unit volume, and temperature. These quantities are rendered in Smokeview as realistic looking smoke and fire. To turn off this feature, set `SMOKE3D=F` on the `DUMP` line.
- **DT_PL3D** The time between Plot3D file output. Note that versions of FDS before 6 output Plot3D files by default. Now, you must specify the interval of output using this parameter. Its default value is 1000000 s, meaning that there is no Plot3D output unless specified.
- **FLUSH_FILE_BUFFERS** FDS purges the output file buffers periodically and forces the data to be written out into the respective output files. It does this to make it easier to view the case in Smokeview while it is running. It has been noticed on Windows machines that occasionally a runtime error occurs because of file access problems related to the buffer flushing. If this happens, set this parameter to `F`, but be aware that it may not be possible to look at output in Smokeview until after the calculation is finished. You may also set `DT_FLUSH` to control the frequency of the file flushing. Its default value is the duration of the simulation divided by `NFRAMES`.
- **STATUS_FILES** If `T`, produces an output file `CHID.notready` which is deleted, if the simulation is completed successfully. This file can be used as an error indicator. It is `F` by default.

Table 22.1: Parameters that control the frequency of output.

Uniform Output	Non-Uniform Output	Purpose
DT_BNDF	RAMP_BNDF	Boundary files
DT_CPU	RAMP_CPU	CPU timings
DT_CTRL	RAMP_CTRL	Control output
DT_DEVC	RAMP_DEVC	Device output
DT_FLUSH	RAMP_FLUSH	File flushing times
DT_HRR	RAMP_HRR	HRR output
DT_HVAC	RAMP_HVAC	HVAC output
DT_ISO	RAMP_ISO	Isosurface output
DT_MASS	RAMP_MASS	Species mass output
DT_PART	RAMP_PART	Particle output
DT_PL3D	RAMP_PL3D	Plot3D output
DT_PROF	RAMP_PROF	In-depth profile output
DT_RADF	RAMP_RADF	Radiation output
DT_RESTART	RAMP_RESTART	Restart times
DT_SLCF	RAMP_SLCF	Slice files
DT_SL3D	RAMP_SL3D	Slice 3D files
DT_SMOKE3D	RAMP_SMOKE3D	Smoke 3D files
DT_SPEC	RAMP_SPEC	Species field output
DT_TMP	RAMP_TMP	Temperature field output
DT_UVW	RAMP_UVW	Velocity field output

22.2 Device Output: The DEVC Namelist Group

Every device (DEVC) records a particular `QUANTITY`. Usually this `QUANTITY` is written out to a comma-delimited spreadsheet file with the suffix `_devc.csv`. The quantities are listed in Tables 22.4 through 22.8.

There are two types of `DEVC` output. The first is a time history of the given `QUANTITY` over the course of the simulation. The second is a time-averaged profile consisting of a linear array of point devices. Each is explained below.

22.2.1 Gas Phase Quantity at a Single Point

If you just want to record the time history of, say, the temperature at a particular point in the gas, add a line like:

```
&DEVC XYZ=6.7,2.9,2.1, QUANTITY='TEMPERATURE', ID='T-1' /
```

and a column will be added to the output file `CHID_devc.csv` under the label `'T-1'`. FDS reports the value of the `QUANTITY` in the cell containing the point `XYZ`. Most scalar quantities, like `TEMPERATURE`, are defined at cell centers and represent the average of that value over the entire cell. If you specify coordinates `XYZ` that place the device on a cell boundary, halfway between two cell centers, FDS chooses the one that has the greater coordinate value. FDS does not take a weighted average among the nearest 8 cell centers. The reason is that some of these cell centers might be on the other side of a thin obstruction.

In cases where the `QUANTITY` is defined on a cell face, like `'U-VELOCITY'`, FDS chooses the nearest cell

face and reports the corresponding value.

22.2.2 Solid Phase Quantity at a Single Point

When prescribing a solid phase quantity, be sure to position the device at a solid surface. It is not always obvious where the solid surface is since the mesh does not always align with the input obstruction locations. To help locate the appropriate surface, the parameter `IOR` *must* be included when designating a solid phase quantity, except when using one of the spatial statistics options that are described in Sec. 22.2.3 in which case the output quantity is not associated with just a single point on the surface. If the orientation of the solid surface is in the positive x direction, set `IOR=1`. If it is in the negative x direction, set `IOR=-1`, and so for the y and z directions. For example, the line

```
&DEVC XYZ=0.7,0.9,2.1, QUANTITY='WALL TEMPERATURE', IOR=-2, ID='...' /
```

designates the surface temperature of a wall facing the negative y direction. There are still instances where FDS cannot determine which solid surface is being designated, in which case an error message appears in the diagnostic output file. Re-position the device and try again. It is best to position the device, via the real triplet `XYZ`, such that the device location is either at or within a cell width *outside* of the solid surface. The search algorithm in FDS will look for the nearest solid surface in the direction opposite to that indicated by `IOR`.

Solid Phase Quantity In-Depth

To record the temperature inside the surface, you can use a device as follows:

```
&DEVC XYZ=..., QUANTITY='INSIDE WALL TEMPERATURE', DEPTH=0.005, ID='Temp_1', IOR=3 /
```

The parameter `DEPTH` (m) indicates a location inside the solid. A positive value indicates the distance from the front surface; a negative value indicates the distance (in absolute value) from the back surface. Note that if the wall thickness is decreasing over time due to the solid phase reactions, and the distance is measured from the current front surface, the measurement point moves toward the back side of the solid. Eventually, the point may emerge from the solid, in which case it returns the ambient temperature. Measuring the distance from the back surface may be more appropriate in this case.

In general, the discrete points within a solid are stretched starting at the surface. If the solid cell center position is needed, it may be output using

```
&DEVC XYZ=..., QUANTITY='INSIDE WALL DEPTH', DEPTH=0.005, ID='XC_1', IOR=3 /
```

The output value should lie within half a solid cell distance from the specified `DEPTH`.

To record the material component's density with time, use the output quantity `'SOLID DENSITY'` in the following way:

```
&DEVC ID='...', XYZ=..., IOR=3, QUANTITY='SOLID DENSITY', MATL_ID='wood', DEPTH=0.001 /
```

This produces a time history of the density of the material referred to as `'wood'`. The density is recorded 1 mm beneath the surface which is oriented in the positive z direction. Note that if `'wood'` is part of a mixture, the density represents the mass of `'wood'` per unit volume of the mixture. Without a `MATL_ID`, the total density will be given.

To record the mass fraction of a material in a solid use `QUANTITY='SOLID MASS FRACTION'`. This quantity requires a `MATL_ID`.

To record the solid conductivity ($\text{W}/(\text{mK})$), use `QUANTITY='SOLID CONDUCTIVITY'`. To record the solid specific heat ($\text{kJ}/(\text{kgK})$), use `QUANTITY='SOLID SPECIFIC HEAT'`. To record the solid enthalpy (kJ/m^3), use `QUANTITY='SOLID ENTHALPY'`. These quantities do not take the optional `MATL_ID` keyword.

Note that these inner solid quantities are allowed only for devices (`DEVC`) or profiles (`PROF`), not contoured boundaries (`BNDF`).

Back Surface Temperature

If you just want to know the temperature of the back surface of the “wall,” then use

```
&DEVC XYZ=..., QUANTITY='BACK WALL TEMPERATURE', ID='Temp_b', IOR=3 /
```

Note that this quantity is only meaningful if the front or exposed surface of the “wall” has the attribute `BACKING='EXPOSED'` on the `SURF` line that defines it. The coordinates, `XYZ`, and orientation, `IOR`, refer to the front surface. To check that the heat conduction calculation is being done properly, you can add the additional line

```
&DEVC XYZ=..., QUANTITY='WALL TEMPERATURE', ID='Temp_f', IOR=-3 /
```

where now `XYZ` and `IOR` refer to the coordinates and orientation of the back side of the wall. These two wall temperatures ought to be the same. Remember that the “wall” in this case can only be at most one mesh cell thick, and its `THICKNESS` need not be the same as the mesh cell width. Rather, the `THICKNESS` ought to be the actual thickness of the “wall” through which FDS performs a 1-D heat conduction calculation.

22.2.3 Spatially-Integrated Outputs

A useful feature of a device (`DEVC`) is to specify an output quantity along with a desired statistic. For example,

```
&DEVC XB=..., QUANTITY='TEMPERATURE', ID='maxT', SPATIAL_STATISTIC='MAX' /
```

causes FDS to write out the maximum gas phase temperature over the volume bounded by `XB`. Other `SPATIAL_STATISTIC`'s are discussed below. Some are appropriate for gas phase output quantities, some for solid phase, and some for both. Note that if `XB` is used for a point device without a `SPATIAL_STATISTIC`, then FDS will define `XYZ` to be the center of the volume defined by `XB`. A `SPATIAL_STATISTIC` can only be used for a gas phase `QUANTITY` in Table 22.12 with both D and S in the file type column or for a solid phase `QUANTITY` in Table 22.13 with both D and B in the file type column.

For solid phase output quantities, like heat fluxes and surface temperatures, the specification of a `SURF_ID` along with the appropriate statistic limits the calculation to only those surfaces. You can further limit the search by using the sextuplet of coordinates `XB` to force FDS to only compute statistics for surface cells within the given volume. Be careful to account for the fact that the solid surface might shift to conform to the underlying numerical grid. Note that you do not (and should not) specify an orientation via the parameter `IOR` when using a spatial statistic. `IOR` is only needed to find a specific point on the solid surface.

Note that a shortcut for specifying the six `XB` values on the `DEVC` line is to instead use `DB='WHOLE DOMAIN'` (`DB` stands for “Domain Boundary”).

Linearly Interpolated Value

By default, FDS reports a given gas-phase `QUANTITY` at a given point `XYZ` to be the value computed at the center of the cell containing the point. However, in some instances, you might want to report a linearly-interpolated value over the nearest eight grid cells:

```
&DEVC XYZ=..., QUANTITY='TEMPERATURE', ID='T', SPATIAL_STATISTIC='INTERPOLATION' /
```

Be careful when using this option near a thin obstruction or a collection of stair-stepped obstructions because the average value might include data from the opposite side of the obstructions. A notable example would be a sprinkler “device” located under a stair-stepped ceiling.

Minimum or Maximum Value

For a scalar quantity defined at the center of gas or solid phase cells, ϕ_{ijk} , set `SPATIAL_STATISTIC` to 'MIN' or 'MAX' to compute the minimum or maximum value, respectively, over the cells that are included in the specified volume bounded by `XB`:

$$\min_{ijk} \phi_{ijk} \quad ; \quad \max_{ijk} \phi_{ijk} \quad (22.1)$$

Note also that you must specify a volume to sum over via the coordinate parameters, `XB`, which can extend into multiple meshes.

If you want to know where the specified `QUANTITY` achieves its minimum or maximum value over the specified volume `XB`, set `SPATIAL_STATISTIC` to 'MINLOC X' or 'MAXLOC X' for the *x* coordinate. Substitute *Y* or *Z* if you desire the *y* or *z* coordinate, in m.

Average Value

For a gas phase scalar quantity defined at the center of each grid cell, ϕ_{ijk} , the `SPATIAL_STATISTIC` 'MEAN' computes the average value,

$$\frac{1}{N} \sum_{ijk} \phi_{ijk} \quad (22.2)$$

over the cells that are included in the specified volume bounded by `XB`. Note that this statistic is only appropriate for gas phase quantities. Note also that you must specify a volume to sum over via the coordinate parameters, `XB`, which can extend into multiple meshes.

If you are interested in the average value over a plane, give the `XB` volume a small thickness to either side of the plane location encompassing the storage locations of the values of interest. For example, scalars like temperatures live at the cell center, so locate the plane at the cell center and add a small delta to either side of that plane for the `XB`.

Volume-Weighted Mean

For a gas phase scalar quantity, $\phi(x,y,z)$, `SPATIAL_STATISTIC='VOLUME MEAN'` produces the discrete analog of

$$\frac{1}{V} \iiint \phi(x,y,z) \, dx \, dy \, dz \quad (22.3)$$

which is very similar to 'MEAN', but it weights the values according to the relative size of the mesh cell. Note that this statistic is only appropriate for gas phase quantities. Note also that you must specify a volume to sum over via the coordinate parameters, `XB`, which can extend into multiple meshes.

Mass-Weighted Mean

For a gas phase scalar quantity, $\phi(x,y,z)$, `SPATIAL_STATISTIC='MASS MEAN'` produces the discrete analog of

$$\frac{\iiint \rho(x,y,z) \phi(x,y,z) \, dx \, dy \, dz}{\iiint \rho \, dx \, dy \, dz} \quad (22.4)$$

which is similar to `'VOLUME MEAN'`, but it weights the values according to the relative mass of the mesh cell. Note that this statistic is only appropriate for gas phase quantities. Note also that you must specify a volume to sum over via the coordinate parameters, `XB`, which can extend into multiple meshes.

Centroids

For a gas phase scalar quantity, $\phi(x,y,z)$, `SPATIAL_STATISTIC='CENTROID X'` produces the discrete analog of

$$\frac{\iiint x \phi(x,y,z) \, dx \, dy \, dz}{\iiint \phi(x,y,z) \, dx \, dy \, dz} \quad (22.5)$$

Similar calculations are performed for `'CENTROID Y'` and `'CENTROID Z'`. Note that this statistic is only appropriate for gas phase quantities, and you must specify a volume to sum over via the coordinate parameters, `XB`.

Volume Integral

For a gas phase scalar quantity, $\phi(x,y,z)$, `SPATIAL_STATISTIC='VOLUME INTEGRAL'` produces the discrete analog of

$$\iiint \phi(x,y,z) \, dx \, dy \, dz \quad (22.6)$$

This statistic is only appropriate for gas phase quantities, in particular those whose units involve m^{-3} . For example, heat release rate per unit volume is an appropriate output quantity. This statistic can also be used on a `DEVC` line with an output quantity of `DENSITY` to output the total mass within the volume bound by `XB`. In all cases, you must specify a volume to sum over via the coordinate parameters, `XB`, which can extend into multiple meshes.

Mass Integral

For a given gas phase output quantity, $\phi(x,y,z)$, `SPATIAL_STATISTIC='MASS INTEGRAL'` produces the discrete analog of

$$\iiint \rho(x,y,z) \phi(x,y,z) \, dx \, dy \, dz \quad (22.7)$$

Note that this statistic is only appropriate for gas phase quantities. Note also that you must specify a volume to sum over via the coordinate parameters, `XB`, which can extend into multiple meshes.

Area Integral

For a gas phase scalar quantity, $\phi(x,y,z)$, `SPATIAL_STATISTIC='AREA INTEGRAL'` produces the discrete analog of

$$\int \phi(x,y,z) \, dA \quad (22.8)$$

where dA depends on the coordinates you specify for `XB`. Note that this statistic is only appropriate for gas phase quantities, in particular those whose units involve m^{-2} . For example, the quantity `'MASS FLUX X'`

along with `SPEC_ID='my gas'` is an appropriate output quantity if you want to know the mass flux of the gas species that you have named 'my gas' through an area normal to the x direction. Note also that you must specify an area to sum over via the coordinate parameters, `XB`, which can extend into multiple meshes.

Area

This feature is usually used in conjunction with `SPATIAL_STATISTIC='AREA INTEGRAL'` above (use two separate `DEVC` lines, each with the same `QUANTITY`). When performing precise mass and energy balances, it is useful to know exactly the area of `VENT` or `SURF` of interest. Note that this may differ slightly from what is specified in the input file due to grid snapping for Cartesian geometries or due to the approximations used to represent curvilinear geometries. Using `SPATIAL_STATISTIC='AREA'` produces the discrete analog of

$$\int dA \quad (22.9)$$

where dA depends on the coordinates you specify for `XB`. To compute the surface area of a solid, use '`SURFACE AREA`', see below.

Surface Integral

For a solid phase scalar quantity, ϕ , `SPATIAL_STATISTIC='SURFACE INTEGRAL'` produces the discrete analog of

$$\int \phi dA \quad (22.10)$$

Note that this statistic is only appropriate for solid phase quantities, in particular those whose units involve m^{-2} . For example, the various heat and mass fluxes are appropriate output quantities.

Surface Area

The solid surface analog of '`AREA`' is '`SURFACE AREA`'. Knowing this value is useful for reconstructing the average flux on a surface, for example. On the `DEVC` line supply an `XB` that encompasses the surface of interest, use an arbitrary `QUANTITY`, and assign a `SURF_ID`.

```
&DEVC XB=..., QUANTITY=..., SURF_ID='my surf', SPATIAL_STATISTIC='SURFACE AREA',
      ID='surf area'/
```

Limiting the Integration

The input parameter `QUANTITY_RANGE` can be used to limit the region of integration for '`AREA INTEGRAL`', '`VOLUME INTEGRAL`', '`MASS INTEGRAL`', and '`SURFACE INTEGRAL`'. If `QUANTITY_RANGE` is set, the integration will only be performed if the value of the `QUANTITY` lies within the `QUANTITY_RANGE` where `QUANTITY_RANGE(1)` is the lower bound and `QUANTITY_RANGE(2)` is the upper bound. For example:

```
&DEVC XB=..., QUANTITY='MASS FRACTION', SPEC_ID='METHANE', SPATIAL_STATISTIC='MASS
      INTEGRAL', QUANTITY_RANGE=0.03,0.15/
```

would output the total mass of methane in the volume `XB` where the mass fraction was between 0.03 and 0.15.

A set of additional `SPATIAL_STATISTIC`'s are available for use with `QUANTITY_RANGE`: '`MASS`', '`VOLUME`', and '`SURFACE AREA`'. These output the mass, volume, or surface area (for a solid phase quantity) where the `QUANTITY` lies within the `QUANTITY_RANGE`. For example:

```
&DEVC XB=..., QUANTITY='TOTAL HEAT FLUX', SPATIAL_STATISTIC='SURFACE AREA', QUANTITY_
RANGE(1)=10./
```

would output the total surface area in the volume XB where the total heat flux exceeds 10 kW/m².

22.2.4 Temporally-Integrated Outputs

In addition to the spatial statistics, temporal statistics can be applied to DEVC output via the parameter TEMPORAL_STATISTIC. Both SPATIAL_STATISTIC and TEMPORAL_STATISTIC can be specified simultaneously on the DEVC line, assuming the combination makes sense. The SPATIAL_STATISTIC is applied first, say an integration of the QUANTITY over a given volume XB, followed by the TEMPORAL_STATISTIC, like a time integral.

Time Average

The TEMPORAL_STATISTIC='TIME AVERAGE' is typically applied by default, unless it is inappropriate for the chosen QUANTITY. With this statistic, FDS outputs the average value of the QUANTITY over the time period of DT_DEVC s just prior to the time of output. The parameter DT_DEVC is specified on the DUMP line.

Running Average

The TEMPORAL_STATISTIC='RUNNING AVERAGE' outputs the running average value of the QUANTITY over the time period starting at STATISTICS_START and ending at STATISTICS_END. This is the default behavior of “line” devices discussed in Sec. 22.2.5.

Favre Average

The TEMPORAL_STATISTIC='FAVRE AVERAGE' outputs the Favre (density weighted) running average value of the QUANTITY over the time period starting at STATISTICS_START and ending at STATISTICS_END. If the scalar quantity of interest is ϕ and the density is ρ and the running average is defined by an overbar (that is, $\bar{\phi}$ is the running average of the scalar), then the Favre average is denoted with a tilde and computed as $\tilde{\phi} = \bar{\rho\phi}/\bar{\rho}$. This output is not compatible with INITIAL_VALUE.

Instantaneous Value

For various reasons you might want to output the value of the QUANTITY that has not been time-averaged or processed in any way. Sometimes this is important for device and control functions. To output the instantaneous value, set TEMPORAL_STATISTIC to 'INSTANT VALUE'.

Smoothed Value

Control functions that use devices as inputs use the value of the devices determined by the SMOOTHING_FACTOR for each device. To output this smoothed value of the device, set TEMPORAL_STATISTIC to 'SMOOTHED'. This statistic outputs the instantaneous smoothed value, and it does not time-average the smoothed value over the output interval.

Time Integral

TEMPORAL_STATISTIC='TIME INTEGRAL' produces a discrete analog of the time integral:

$$\int_{t_0}^t \phi(\tau) d\tau \quad (22.11)$$

Minimum and Maximum Values over Time

Set TEMPORAL_STATISTIC to 'MIN' or 'MAX' to determine the minimum or maximum value of the QUANTITY over a time interval:

$$\min_{t_0 \leq t \leq t_1} \phi(t) \quad ; \quad \max_{t_0 \leq t \leq t_1} \phi(t) \quad (22.12)$$

where t_0 is STATISTICS_START and t_1 is STATISTICS_END.

For some applications you might need to predict a minimum or maximum value of the QUANTITY over a time interval that is longer than the simulation time. For example, in wind engineering the so-called 50 year maximum wind speed can be estimated using maximum yearly values collected over, say, 10 years. The idea is that extreme values conform to certain statistical distributions from which one can extrapolate extreme values over long time periods. In FDS, you can do something similar. If you add the parameter TIME_PERIOD to the DEVC line where you have also set the TEMPORAL_STATISTIC to 'MIN' or 'MAX', the time interval bounded by t_0 and t_1 is subdivided into $N = N_INTERVALS$ (default 10), and a minimum or maximum value is stored for each interval. At the end of the simulation, the set of maximum values (minimum values are handled similarly) are ranked in ascending order, $m = 1, N$. Next, the cumulative statistical distribution is introduced [81]:

$$F(\phi) = 1 - \exp[-\exp(\phi)] \quad ; \quad \phi = \ln[-\ln(1 - F)] \quad (22.13)$$

which represents the probability¹ of the value ϕ not being exceeded in the given time interval, $(t_1 - t_0)/N$. The probabilities for each of the N ascending maximum values are assumed to be $m/(N + 1)$. Plotting ϕ versus the expression on the right yields an approximate straight line from which we can estimate the value of ϕ appropriate for the longer TIME_PERIOD; that is, the value for which the probability of not being exceeded in the relatively short time interval $(t_1 - t_0)/N$ is very close to 1. Returning to the example from wind engineering, the probability of not exceeding the 50 year wind speed in a given year is $1 - 1/50 = 0.98$.

Time to Reach Minimum and Maximum Value

The time at which the given QUANTITY achieves its maximum or minimum value is specified by the TEMPORAL_STATISTIC 'MAX TIME' or 'MIN TIME', respectively.

Root Mean Square (RMS)

TEMPORAL_STATISTIC='RMS' produces the unbiased estimate of the *root mean square* of the QUANTITY ϕ :

$$\phi_{\text{rms}} = \sqrt{\frac{\sum_{i=1}^n (\phi_i - \bar{\phi})^2}{n - 1}} \quad (22.14)$$

The computation starts at STATISTICS_START and ends at STATISTICS_END.

¹Note that the parameterless form of the distribution is intended for extrapolation only.

Favre Root Mean Square

TEMPORAL_STATISTIC='FAVRE RMS' produces the unbiased estimate of the density-weighted *root mean square* of the QUANTITY ϕ :

$$\tilde{\phi}_{\text{rms}} = \sqrt{\frac{\sum_{i=1}^n (\phi_i - \tilde{\phi})^2}{n-1}} \quad ; \quad \tilde{\phi} = \overline{\rho\phi} / \bar{\rho} \quad (22.15)$$

The computation starts at STATISTICS_START and ends at STATISTICS_END. This output is not compatible with INITIAL_VALUE.

Covariance

If $u = U - \bar{U}$ and $v = V - \bar{V}$ are the deviations for two random variables, U and V , then an unbiased estimate of the *covariance* is given by

$$\overline{uv} = \frac{\sum_{i=1}^n (U_i - \bar{U})(V_i - \bar{V})}{n-1} \quad (22.16)$$

To output this statistic you must add a QUANTITY2 to the DEVC line and set TEMPORAL_STATISTIC='COV'. The following lines would create a line of devices and a single point device equivalent to the first point in the line:

```
&DEVC XB=X1,X2,Y1,Y2,Z1,Z2, QUANTITY='CELL W', QUANTITY2='TEMPERATURE',
      TEMPORAL_STATISTIC='COV', ID='wT-cov_line', POINTS=20 /
&DEVC XYZ=X1,Y1,Z1, QUANTITY='CELL W', QUANTITY2='TEMPERATURE',
      TEMPORAL_STATISTIC='COV', ID='wT-cov_point', STATISTICS_START=20. /
```

Correlation Coefficient

Setting TEMPORAL_STATISTIC='CORRcoef' outputs the *correlation coefficient* given by

$$\rho_{uv} = \frac{\overline{uv}}{u_{\text{rms}} v_{\text{rms}}} \quad (22.17)$$

Here again you must add a QUANTITY2 to the device line.

22.2.5 Linear Array of Point Devices

You can use a single DEVC line to specify a linear array of devices. By adding the parameter POINTS and using the sextuple coordinate array XBP², you can direct FDS to create a line of devices from (x_1, y_1, z_1) to (x_2, y_2, z_2) . There are two options.

Steady-State Profile

Sometimes it is convenient to calculate a steady-state profile. For example, the vertical velocity profile along the centerline of a doorway can be recorded with the following line of input:

```
&DEVC XBP=X1,X2,Y1,Y2,Z1,Z2, QUANTITY='U-VELOCITY', ID='vel', POINTS=20,
      STATISTICS_START=20., STATISTICS_END=40. /
```

²Earlier versions of FDS made use of the array XB to specify the extents of a linear array of point devices. However, XB might be needed to specify a particular type of SPATIAL_STATISTIC associated with each device.

In a file called `CHID_line.csv`, there will be between 1 and 4 columns of data associated with this single `DEVC` line. If x_1 is different than x_2 , there will be a column of x coordinates associated with the linear array of points. The same holds for the y and z coordinates. The last column contains the 20 temperature points time-averaged over the interval between `STATISTICS_START` and `STATISTICS_END`. The default value of the latter is `T_END` and the former is half the total simulation time. This is a convenient way to output a time-averaged linear profile of a quantity, like an array of thermocouples. Note that the statistics output to the `_line.csv` file start being averaged at `T=STATISTICS_START`. Prior to this time, the values are zero. This prevents initial transient from biasing the final average value or other temporal statistics.

By default, the points are uniformly spaced between (x_1, y_1, z_1) and (x_2, y_2, z_2) . To change this, you can add arrays of coordinate values `POINTS_ARRAY_X`, `POINTS_ARRAY_Y`, and/or `POINTS_ARRAY_Z`. For example, the line

```
&DEVC XBP=0,0,0,0,0,1, QUANTITY='TEMPERATURE', ID='T', POINTS=4,
      POINTS_ARRAY_Z=0.1,0.2,0.3,0.7 /
```

creates an array of four points in the vertical direction that are not uniformly spaced. The original values of 0 and 1 in the array `XBP` are ignored. Using all three `POINTS_ARRAYS`, you can create a curved line of points. The upper dimension of these arrays is 100.

A single “line” file can hold more than a single line of data. By default, the coordinate columns are labeled using the `ID` of the `DEVC` appended with either `-x`, `-y`, or `-z`. To change these labels, use `X_ID`, `Y_ID`, and/or `Z_ID`. To suppress the coordinate columns altogether, add `HIDE_COORDINATES=T` to the `DEVC` line. This is convenient if you have multiple arrays of data that use the same coordinates. If you want the data plotted as a function of the distance from the origin, $r = \sqrt{x^2 + y^2 + z^2}$, provide the label `R_ID`. If you want the data plotted as a function of the distance from the first point on the line, (x_1, y_1, z_1) , provide the label `D_ID`.

You can convert the spatial coordinate of a steady-state profile by setting `COORD_FACTOR` on the `DEVC` line. For example, to convert from the default unit of meters to centimeters, set `COORD_FACTOR` to 100. Also set `XYZ_UNITS` to the appropriate character string, in this case `XYZ_UNITS='cm'`.

Time-Varying Profile

If you do not want a steady-state profile, but rather you just want to specify an array of evenly spaced devices, you can use a similar input line, except with the additional attribute `TIME_HISTORY`.

```
&DEVC XBP=X1,X2,Y1,Y2,Z1,Z2, QUANTITY='U-VELOCITY', ID='vel', POINTS=20,
      TIME_HISTORY=T /
```

This directs FDS to just add 20 devices to the on-going list, saving you from having to write 20 `DEVC` lines. The `ID` for each device will be `'vel-01'`, `'vel-02'`, etc.

Other Statistics for a Linear Array of Devices

By default, when you specify multiple `POINTS` on a `DEVC` line and you do not specify `TIME_HISTORY=T`, a running average of the specified `QUANTITY` is saved at each point location and written to the file with suffix `_line.csv` with the accumulated values at the time `STATISTICS_END`. However, you can apply any other `TEMPORAL_STATISTIC` and compute, for example, the root-mean-square (`'RMS'`), minimum (`'MIN'`), or maximum (`'MAX'`) value over the specified time interval.

You may also specify a `SPATIAL_STATISTIC` for each device in the line as follows:


```
&DEVC XBP=1,10,5,5,8,8, QUANTITY='TEMPERATURE', ID='Tmean', POINTS=20,
      STATISTICS_START=20., STATISTICS_END=40.
      SPATIAL_STATISTIC='MEAN', DX=0.1, DY=5.0, DZ=3.0 /
```

This input line specifies a linear array of 20 points starting at $\mathbf{x} = (1, 5, 8)$ and ending at $\mathbf{x} = (10, 5, 8)$. For each point in the array, (x_i, y_i, z_i) , FDS calculates the average temperature over the volume $x_i - \delta x \leq x \leq x_i + \delta x$; $y_i - \delta y \leq y \leq y_i + \delta y$; $z_i - \delta z \leq z \leq z_i + \delta z$. These average temperatures are then time-averaged over the time interval between 20 s and 40 s. In general, the spatial averaging is done first; that is, FDS reports the time-average of the spatially-averaged temperatures for each point in the linear array, not the spatial-average of the temporally-averaged temperatures.

Moving a Linear Array of Devices

It is possible to translate and/or rotate a linear array of devices using parameters on a `MOVE` line (see Sec. 11.4). To specify these parameters, use the character string `MOVE_ID` on the `DEVC` line. Note that if a `SPATIAL_STATISTIC` is specified, the spatial bounds, `DX`, `DY`, and/or `DZ`, are not transformed.

22.3 In-Depth Profiles within Solids: The PROF Namelist Group

FDS uses a fine one-dimensional grid at each boundary cell to calculate the heat conduction and reactions within a solid. The solid can be a wall cell or a Lagrangian particle. In either case, use the PROF output to record the properties of the solid in depth at discrete intervals of time. A PROF can use any solid phase output QUANTITY that takes a DEPTH. These are identified with the file type “Pr” in Table 22.13.

The parameters for a PROF line are listed in Table 23.24. For a wall cell, the parameters are similar to those used to specify a surface quantity in the DEVC group. XYZ designates the triplet of coordinates, IOR the orientation, and ID an identifying character string. Here is an example of how you would use this feature to get a time history of temperature profiles within a wall cell:

```
&PROF ID='T-1', XYZ=..., QUANTITY='INSIDE WALL TEMPERATURE', IOR=3 /
```

For a particle, there are two options. The first is to reference the INIT line used to place the particle within the domain:

```
&INIT ID='my particle', XYZ=..., N_PARTICLES=1, PART_ID='...' /  
&PROF ID='T-2', INIT_ID='my particle', QUANTITY='INSIDE WALL TEMPERATURE' /
```

The second option is to specify the particle using its unique identifying tag³. Once you have determined the tag, add the following line to the input file:

```
&PROF LP_TAG=..., PART_ID='...', QUANTITY='INSIDE WALL TEMPERATURE' /
```

Note that you must specify the PART_ID so that the appropriate inner noding structure can be determined. With either option for designating a particle, do not specify XYZ or IOR on the PROF line.

The QUANTITY is the physical quantity to record. Each PROF line creates a separate file. The first number in each row is the time at which the profile is extracted. The second number is the number of “nodes,” n , which is the number of solid phase cells plus 1. The next n values are the node coordinates, running from 0 at the surface to the full THICKNESS at the end. The next n numbers are the values of the QUANTITY at the node points. The first of these values is located at the surface. If you specify CELL_CENTERED=T, the QUANTITY values will be recorded at n interior cell centers rather than cell boundaries. The ID is an optional input which, if provided, is used in the file header.

There is a second optional format. If you specify FORMAT_INDEX=2 on the PROF line, the resulting file will contain columns containing only the final set of node coordinates and quantity values. This is handy for displaying a steady-state temperature profile.

³Each droplet or particle has a unique integer identifier. To find it, open the case in Smokeview, load the particles, right click, “Files/Data/Coloring,” “Particles,” “Settings,” and click the box labeled “Show selected particle tag.” Then click on the desired particle.

22.4 Animated Planar Slices: The SLCF Namelist Group

The `SLCF` (“slice file”) namelist group parameters (Table 23.30) allows you to record various gas phase quantities at more than a single point. A “slice” refers to a subset of the whole domain. It can be a line, plane, or volume, depending on the values of `XB`. The sextuplet `XB` indicates the boundaries of the “slice” plane. `XB` is prescribed as in the `OBST` or `VENT` groups, with the possibility that 0, 2, or 4 out of the 6 values be the same to indicate a volume, plane or line, respectively. A handy trick is to specify, for example, `PBY=5.3` instead of `XB` if it is desired that the entire plane $y = 5.3$ slicing through the domain be saved. `PBX` and `PBZ` control planes perpendicular to the x and z axes, respectively. An alternative way to specify a slice plane through the middle of the domain is using `DB` with values of `'XMID'`, `'YMID'`, or `'ZMID'`.

By default, 1-D and 2-D slice files are saved `NFRAMES` times per simulation. You can control the frequency of output with `DT_SLCF` on the `DUMP` line. If the “slice” is a 3-D volume, then its output frequency is controlled by the parameter `DT_SL3D`. You may specify a value of `DT_SL3D` on `DUMP` or provide a series of discrete times. Note that 3-D slice files can become extremely large if `DT_SL3D` is small.

Animated vectors can be created in Smokeview if a given `SLCF` line has the attribute `VECTOR=T`. If two `SLCF` entries are in the same plane, then only one of the lines needs to have `VECTOR=T`. Otherwise, a redundant set of velocity component slices will be created.

Normally, FDS averages slice file data at cell corners. For example, gas temperatures are computed at cell centers, but they are linearly interpolated to cell corners and output to a file that is read by Smokeview. To prevent this from happening, set `CELL_CENTERED=T`. This forces FDS to output the actual cell-centered data with no averaging. Note that this feature is mainly useful for diagnostics because it enables you to visualize the values that FDS actually computes. If `CELL_CENTERED=T` is combined with `VECTOR=T` then the staggered velocity components will be displayed. For example,

```
&SLCF PBX=0, QUANTITY='VELOCITY', VECTOR=T, CELL_CENTERED=T /
```

will show the staggered velocity components as cell face normal vectors in Smokeview.

Slice file information is recorded in files (See Sec. 27.9) labeled `CHID_$.sf`, where n is the index of the slice file. A short Fortran program `fds2ascii.f90` produces a text file from a line, plane or volume of data. See Sec. 22.11 for more details.

By default, Smokeview will blank slice file data inside obstructions. However, this is expensive to load at startup in Smokeview for large cases. If you wish Smokeview not to store this blanking array, set `IBLANK_SMV=F` on `MISC`. Another option is to run Smokeview from the command line and to add `-noblack` as an option.

You may add a name via `ID` on the `SLCF` as an identifier sent to Smokeview. It is only optional.

22.5 Animated Boundary Quantities: The BNDF Namelist Group

The BNDF (“boundary file”) namelist group parameters allows you to record surface quantities at all solid obstructions. As with the SLCF group, each quantity is prescribed with a separate BNDF line, and the output files are of the form CHID_\$n\$.bf. No physical coordinates need be specified, however, just QUANTITY. See Table 22.5. For certain output quantities, additional parameters need to be specified via the PROP namelist group. In such cases, add the character string, PROP_ID, to the BNDF line to tell FDS where to find the necessary extra information.

BNDF files (Sec. 27.11) can become very large, so be careful in prescribing the time interval, DT_BNDF, or discrete times, RAMP_BNDF, on the DUMP line. One way to reduce the size of the output file is to turn off the drawing of boundary information on desired obstructions. On any given OBST line, if the string BNDF_OBST=F is included, the obstruction is not colored. To turn off all boundary drawing, set BNDF_DEFAULT=F on the MISC line. Then individual obstructions can be turned back on with BNDF_OBST=T on the appropriate OBST line. Individual faces of a given obstruction can be controlled via BNDF_FACE(IOR), where IOR is the index of orientation (+1 for the positive x direction, -1 for negative, and so on). Another way is to disable BNDF output for specific meshes by setting BNDF_MESH=F on the MESH lines where no BNDF output is desired. Disabling output for a MESH includes all wall cells at the exterior boundary of that MESH.

Normally, FDS averages boundary file data at cell corners. For example, surface temperatures are computed at the center of each surface cell, but they are linearly interpolated to cell corners and output to a file that is read by Smokeview. To prevent this from happening, set CELL_CENTERED=T on the BNDF line. This forces FDS to output the actual cell-centered data with no averaging. Note that this feature is mainly useful for diagnostics.

Applying boundary files to complex geometries can be controlled with the BNDF_GEOM logical on the GEOM line. Note that BNDF output for GEOM uses face centered values only (node centered is not available; the user need not set CELL_CENTERED=T).

Sometimes it is useful to render the QUANTITY integrated over time. For example, a heat flux in units of kW/m^2 can be integrated in time producing the total energy absorbed by the surface in units of kJ/m^2 . To do this, set TEMPORAL_STATISTIC equal to 'TIME INTEGRAL' on the BNDF line. Note that there are no other options for TEMPORAL_STATISTIC on a BNDF line.

22.6 Animated Isosurfaces: The ISOF Namelist Group

The ISOF (“ISOsurface File”) namelist group creates three-dimensional animated contours of gas phase scalar quantities. For example, a 300 °C temperature isosurface is a 3-D surface on which the gas temperature is 300 °C. Three different values of the temperature can be saved via the line:

```
&ISOF QUANTITY='TEMPERATURE', VALUE(1)=50., VALUE(2)=200., VALUE(3)=500. /
```

where the values are in °C. Note that the isosurface output files CHID_\$n\$.iso can become very large, so experiment with different sampling rates, (DT_ISOF, or discrete times, RAMP_ISOF, either of which is specified on the DUMP line). The parameter QUANTITY2 can be used to color the isosurface. For example, the line:

```
&ISOF QUANTITY='MIXTURE FRACTION' , VALUE(1)=0.05, QUANTITY2='TEMPERATURE' /
```

draws a surface where the mixture fraction has a value of 0.05 kg/kg and colors the surface this using temperature. The parameter SKIP=n (default: 1) can also be used to reduce data by skipping n-1 values in each direction. The parameter DELTA=val can also be used to reduce data by forcing each triangle to have all sides larger than val.

Any gas phase quantity can be animated via iso-surfaces, but use caution. To render an iso-surface, the desired quantity must be computed in every mesh cell at every output time step. For quantities like 'TEMPERATURE', this is not a problem, as FDS computes it and saves it anyway. However, species volume fractions demand substantial amounts of time to compute at each mesh cell. Remember to include the SPEC_ID corresponding to the given QUANTITY if necessary.

22.7 Plot3D Static Data Dumps

Data stored in Plot3D [82] files use a format developed by NASA that is used by many CFD programs for representing simulation results. See Sec. 27.10 for a description of the file structure. Plot3D data is visualized in three ways: as 2-D contours, vector plots and iso-surfaces. Vector plots may be viewed if one or more of the u , v and w velocity components are stored in the Plot3D file. The vector length and direction show the direction and relative speed of the fluid flow. The vector colors show a scalar fluid quantity such as temperature. Five quantities are written out to a file at one instant in time. The default specification is:

```
&DUMP ..., PLOT3D_QUANTITY(1:5)='TEMPERATURE',  
        'U-VELOCITY','V-VELOCITY','W-VELOCITY','HRRPUV' /
```

It's best to leave the velocity components as is, because Smokeview uses them to draw velocity vectors. If any of the specified quantities require the additional specification of a particular species, use `PLOT3D_SPEC_ID(n)` to provide the `SPEC_ID` for `PLOT3D_QUANTITY(n)`.

Plot3D data are stored in files with extension `.q`. There is an optional file that can be output with coordinate information if another visualization package is being used to render the files. If you write `WRITE_XYZ=T` on the `DUMP` line, a file with suffix `.xyz` is written out. Smokeview does not require this file because the coordinate information can be obtained elsewhere.

Past versions of FDS (1-5) output Plot3D files by default. Now, you must specify the time interval between dumps using `DT_PL3D` on the `DUMP` line.

22.8 SMOKE3D: Realistic Smoke and Fire

For any simulation involving combustion, FDS automatically creates three output files that are rendered by Smokeview as realistic looking smoke, fire, and/or hot gas. By default, the output quantities are the 'DENSITY' of 'SOOT', the 'HRRPUV' (Heat Release Rate Per Unit Volume), and the `TEMPERATURE` of the gases. These quantities are rendered as semi-transparent slices that give you the illusion of looking through smoke or fire. The greater the value of the quantity, the darker the slices will appear en masse. For `HRRPUV`, the lowest value rendered is:

$$\dot{q}_{\min}'' = \min(200, 20/\delta x) \text{ kW/m}^3 \quad (22.18)$$

where δx is the grid cell size.

You have the option of rendering the `DENSITY` of any other species besides 'SOOT', so long as the `MASS_EXTINCTION_COEFFICIENT` on the `SPEC` line is appropriate in describing the attenuation of visible light by the specified gas species. Here is an example of how to change the smoke species. Normally, you do not need to do this as the "smoke" is an assumed part of the default combustion model when a non-zero `SOOT_YIELD` is defined on the `REAC` line.

```
&SPEC ID='MY SMOKE', MW=29., MASS_EXTINCTION_COEFFICIENT=8700. /  
&SM3D QUANTITY='DENSITY', SPEC_ID='MY SMOKE' /
```

The `MASS_EXTINCTION_COEFFICIENT` is passed to Smokeview to be used for visualization.

FDS outputs 3D smoke quantities as 8 bit integers compressed using run length encoding. Soot density, `HRRPUV` or temperatures are first scaled to 8 bit integers (soot density is converted to an opacity first). Repeated integers are replaced by nI where n is the number of repeats and I is the value repeated.

You can change the scale of `HRRPUV` via the parameter `HRRPUV_MAX_SMV` (default 1200 kW/m³) on the `DUMP` line. You can change the `TEMPERATURE` bounds via `TEMP_MIN_SMV` and `TEMP_MAX_SMV` (default 20 °C and 2000 °C).

22.9 Particle Output Quantities

This section discusses output options for Lagrangian particles.

22.9.1 Liquid Droplets that are Attached to Solid Surfaces

Liquid droplets (as opposed to solid particles) “stick” to solid surfaces unless directed otherwise. There are various quantities that describe these populations. For example, 'MPUA' is the Mass Per Unit Area of the droplets⁴ defined by PART_ID. Likewise, 'AMPUA' is the Accumulated Mass Per Unit Area. Both of these are given in units of kg/m². Think of these outputs as measures of the instantaneous mass density per unit area, and the accumulated total, respectively. These quantities are not identical measures. The quantity 'AMPUA' is analogous to a “bucket test,” where the droplets are collected in buckets and the total mass determined at the end of a given time period. In this case each grid cell on the floor is considered its own bucket. Each droplet is counted only once when it reaches the floor⁵. MPUA counts a droplet whenever it is on any solid surface, including the walls. If the droplet moves from one solid wall cell to another, it will be counted again. The cooling of a solid surface by droplets of a given type is given by 'CPUA', the Cooling Per Unit Area in units of kW/m². Since a typical sprinkler simulation only tracks a small fraction of the droplets emitted from a sprinkler, both MPUA and CPGA also perform an exponential smoothing. This avoids having spotted distributions on surfaces due to the infrequent arrival of droplets that likely have a high weighting factor.

Each of the output quantities mentioned above has a variant in which the quantity is summed by species rather than particle type. For example, the quantity 'AMPUA_Z' along with a specified SPEC_ID rather than a PART_ID will sum the given output quantity over all particle classes with the given SPEC_ID.

As an example of how to use these kinds of output quantities, the test case bucket_test_1 describes a single sprinkler mounted 10 cm below a 5 m ceiling. Water flows for 30 s at a constant rate of 180 L/min (ramped up and down in 1 s). The simulation continues for another 10 s to allow water drops time to reach the floor. The total mass of water discharged is

$$180 \frac{\text{L}}{\text{min}} \times 1 \frac{\text{kg}}{\text{L}} \times \frac{1}{60} \frac{\text{min}}{\text{s}} \times 30 \text{ s} = 90 \text{ kg} \quad (22.19)$$

In the simulation, the quantity 'AMPUA' with SPATIAL_STATISTIC='SURFACE INTEGRAL' is specified on the DEVC line. This results in FDS summing 'AMPUA' over each grid cell in the volume defined by XB, in this case the entire floor, analogous to if there were an single bucket present that was the same size as the area specified with XB. Summing the values of 'AMPUA' over the entire floor yields a total of 90 kg (Fig. 22.1). Note that there really is no need to time-average the results. The quantity is inherently accumulating.

22.9.2 Solid Particles on Solid Surfaces

If you want to monitor the accumulation of solid particles that have fallen on a solid surface, use a device as follows:

```
&DEVC ID=..., XB=..., QUANTITY='MPUV', PART_ID='rods', SPATIAL_STATISTIC='VOLUME  
INTEGRAL' /
```

The volume over which to integrate, XB, should be at least one grid cell thick above the surface.

⁴The output quantity 'MPUA' can be used for both liquid and solid particles. However, 'AMPUA' is appropriate only for liquid droplets.

⁵Be aware of the fact that the default behavior for liquid droplets hitting the “floor,” that is, the plane $z = ZMIN$, is to disappear (POROUS_FLOOR=T on the MISC line). In this case, 'MPUA' will be zero, but 'AMPUA' will not. FDS stores the droplet mass just before removing the droplet from the simulation for the purpose of saving CPU time.

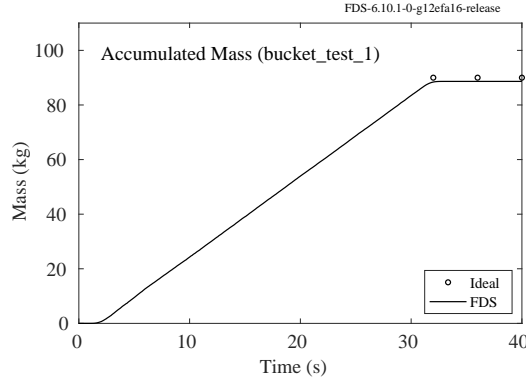


Figure 22.1: Accumulated water collected at the floor in the `bucket_test_1` case.

22.9.3 Droplet and Particle Densities and Fluxes in the Gas Phase

Away from solid surfaces, 'MPUV' is the Mass Per Unit Volume of particles or droplets of type (PART_ID) in units of kg/m^3 . The average volume fraction of droplets in the cell is 'DROPLET VOLUME FRACTION', which is equal to the MPUV divided by the liquid density of the droplets. 'MPUV_Z' provides the same information integrated over all droplets of a single species, (SPEC_ID).

The quantities 'PARTICLE FLUX X', 'PARTICLE FLUX Y', and 'PARTICLE FLUX Z' produce slice and Plot3D colored contours of the mass flux of particles in the x , y , and z directions, respectively, in units of $\text{kg/m}^2/\text{s}$. You can also apply these quantities to a device. For example, in the case called `bucket_test_4.fds`, the input line

```
&DEVC XB=..., ID='flux', QUANTITY='PARTICLE FLUX Z', SPATIAL_STATISTIC='AREA
INTEGRAL' /
```

records the integrated mass flux of *all* particles passing through the given horizontal plane. Figure 22.2 presents the results of this simple test case in which water spraying at a rate of 0.0005 kg/s for 55 s passes through a measurement plane and onto the floor. The total water accumulated is 0.0275 kg .

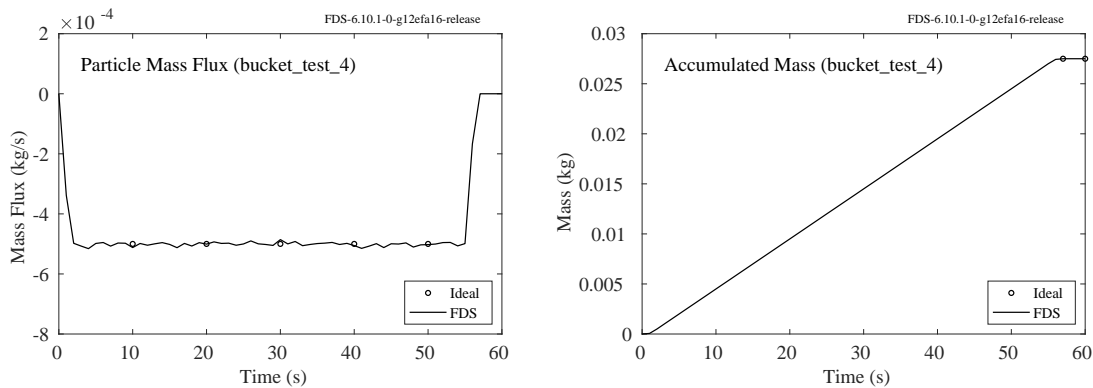


Figure 22.2: Mass flux and accumulated water collected at the floor in the `bucket_test_4` case.

The quantity 'PARTICLE RADIATION LOSS' reports the net energy emitted and absorbed from a given grid cell by all particles within it, in units of kW/m^3 . A positive value indicates that emission is greater than absorption.

22.9.4 Coloring Particles and Droplets in Smokeview

The parameter `QUANTITIES` on the `PART` line is an array of character strings indicating which scalar quantities should be used to color particles and droplets in Smokeview. The choices are

```
'PARTICLE AGE' (s)
'PARTICLE DIAMETER' (μm)
'PARTICLE TEMPERATURE' (°C)
'PARTICLE MASS' (kg)
'PARTICLE PHASE'
'PARTICLE VELOCITY' (m/s)
'PARTICLE WEIGHTING FACTOR'
'PARTICLE U', 'PARTICLE V', 'PARTICLE W' (m/s)
'PARTICLE X', 'PARTICLE Y', 'PARTICLE Z' (m)
'PARTICLE ACCEL X', 'PARTICLE ACCEL Y', 'PARTICLE ACCEL Z' (m/s2)
'PARTICLE DRAG FORCE X', 'PARTICLE DRAG FORCE Y', 'PARTICLE DRAG FORCE Z' (N)
'PARTICLE DRAG COEFFICIENT'
'PARTICLE BULK DENSITY' (kg/m3)
'PARTICLE HEAT TRANSFER COEFFICIENT' (W/m2/K)
'PARTICLE RADIATIVE HEAT FLUX' (kW/m2)
'PARTICLE CONVECTIVE HEAT FLUX' (kW/m2)
'PARTICLE TOTAL HEAT FLUX' (kW/m2)
```

If no `QUANTITIES` are specified and none are selected in Smokeview, then Smokeview will display particles with a single color determined by the color of the `SURF_ID` assigned to the `PART` class, with the exception of water droplets and liquid fuel droplets, which are colored blue and yellow, respectively. If no color is specified on `SURF` then the default solid particle color will revert to black. You may override the `SURF` color by specifying either `RGB` or `COLOR` on the `PART` line.

The `'PARTICLE TEMPERATURE'` is the surface temperature. For thermally thin particles, like droplets, this temperature represents the whole particle, but for thermally thick particles it is just the surface temperature. For in-depth temperature of thermally thick particles you need to use a profile `PROF`, as discussed in Sec. 22.3. In practice, you should designate only a small subset of particles (use a separate `PART` line) for monitoring internal temperatures, as the amount of data can become large, and there is no way to visualize this in Smokeview.

The `PARTICLE WEIGHTING FACTOR` describes how many actual particles each of the computational particles represent. The `PARTICLE BULK DENSITY` gives the weighted (total) mass per grid cell volume, which is an important parameter when using particles to represent porous media such as vegetation.

For solid particles with a specified `SURF_ID`, you may specify any of the solid phase output quantities listed in Table 22.7. If the specified quantity is associated with a species, use the parameter `QUANTITIES_SPEC_ID(N)` to specify the species. Here `N` refers to the order of the specified output quantities on the `PART` line.

22.9.5 Detailed Properties of Solid Particles

You may output properties of a single solid particle using a `DEVC` (device) line. For example, the lines:

```
&INIT ID='my particle', PART_ID='...', XB=..., N_PARTICLES=1 /
&DEVC ID='...', INIT_ID='my particle', QUANTITY='WALL TEMPERATURE' /
```

adds a column to the `CHID_devc.csv` file for the surface temperature of a single particle that has been introduced into the simulation via an `INIT` line.

If the `INIT` line has a `MULT_ID`; that is, an array of particles is introduced via this single `INIT` line, then the `INIT_ID` for each takes the form `'[ID]-00308'`, where `ID` is that of the `INIT` line, and 308 refers to the 308-th `INIT` line generated by the multiplicative sequence. The `INIT` lines are generated by looping over the x indices, then y , then z . For example, the following input lines

```
&INIT ID='P', PART_ID='CRIB1', XYZ=0.005,0.005,0.005 ,  
      N_PARTICLES=1, CELL_CENTERED=T, MULT_ID='array' /  
&MULT ID='array', DX=0.01, DY=0.01, DZ=0.01, I_LOWER=0, I_UPPER=9,  
      J_LOWER=0, J_UPPER=9, K_LOWER=0, K_UPPER=3 /  
&DEVC ID='T307', INIT_ID='P-00307', QUANTITY='WALL TEMPERATURE' /
```

specify an array of 10 by 10 by 4 particles. We want to record the surface temperature of the 307-th particle. See Sec. 7.6 for details of how the `MULT` line is interpreted.

If you know the unique identifying tag⁶ of a particular particle, you can output a solid phase quantity via the following line:

```
&DEVC ID='...', LP_TAG=..., QUANTITY='WALL TEMPERATURE' /
```

The `QUANTITY` in the case of a solid (i.e. non-liquid) particle would be taken from Table 22.5. Note that some of the listed quantities are not appropriate for particles.

Solid particles can be used as surrogate targets. For example, the following lines of input create a massless particle that can be used to record the heat flux at a given location away from a solid wall. In this case, the mock heat flux gauge is pointing in the $-x$ direction:

```
&DEVC ID='flux', INIT_ID='f1', QUANTITY='RADIATIVE HEAT FLUX' /  
&INIT ID='f1', XYZ=..., N_PARTICLES=1, PART_ID='rad gauge' /  
&PART ID='rad gauge', STATIC=T, ORIENTATION=-1,0,0, SURF_ID='target' /  
&SURF ID='target', RADIUS=0.001, GEOMETRY='SPHERICAL', EMISSIVITY=1. /
```

Note that the `DEVC` line does not contain device coordinates, but rather a reference to the `INIT` line that positions the single surrogate particle at the point `XYZ`. The `INIT` line references the `PART` line, which provides information about the particle, in particular the orientation of the heat flux gauge. The reference to the `SURF` line is mainly for consistency—FDS needs to know something about the particle’s geometry even though it is really just a “target.” Its volume is meaningless. The functionality of surrogate particles can be extended to model an array of devices. Instead of one heat flux gauge, we can create a line of them:

```
&DEVC ID='flux', INIT_ID='f1', POINTS=34, QUANTITY='RADIATIVE HEAT FLUX', X_ID='x' /  
&INIT ID='f1', XYZ=..., N_PARTICLES=34, DX=0.05, PART_ID='rad gauge' /
```

Note that the parameter `DX` on the `INIT` line creates a line of particles starting at the point `XYZ` and repeating every 0.05 m. For more information about specifying arrays of devices via the parameter `POINTS`, see Sec. 22.2.5.

⁶Each droplet or particle has a unique integer identifier. To find it, open the case in Smokeview, load the particles, right click, “Files/Data/Coloring,” “Particles,” “Settings,” and click the box labeled “Show selected particle tag.” Then click on the desired particle.

22.10 Special Output Features

This section lists a variety of output quantities that are useful for studying thermally-driven flows, combustion, pyrolysis, and so forth. Note that some of the output quantities can be produced in a variety of ways.

22.10.1 Heat Release Rate and Energy Conservation

Quantities associated with the overall energy budget are reported in the comma delimited file `CHID_hrr.csv`. This file is automatically generated; the only input parameters associated with it are `DT_HRR` and `RAMP_HRR` on the `DUMP` line. The columns in this file record the time history of the integrals of the terms in the enthalpy transport equation. The columns are defined as follows:

$$\begin{aligned}
 \underbrace{\frac{\partial}{\partial t} \int \rho h_s dV}_{Q_ENTH} = & \underbrace{\int \dot{q}''' dV}_{HRR} + \underbrace{\left(\dot{q}_{p,r} - \int \nabla \cdot \dot{\mathbf{q}}_r'' dV \right)}_{Q_RADI} + \underbrace{\sum_{\alpha} \dot{m}_{p,\alpha} h_{s,\alpha} - \int \rho \mathbf{u} h_s \cdot d\mathbf{S}}_{Q_CONV} \\
 & + \underbrace{\left(\dot{q}_{p,w} - \int \dot{q}_c'' dA \right)}_{Q_COND} + \underbrace{\sum_{\alpha} \int h_{s,\alpha} \rho D_{\alpha} \nabla Y_{\alpha} \cdot d\mathbf{S}}_{Q_DIFF} + \underbrace{\int \frac{d\bar{p}}{dt} dV}_{Q_PRES} \quad (22.20) \\
 & + \underbrace{\left(-\dot{q}_{p,r} - \dot{q}_{p,c} - \dot{q}_{p,w} \right)}_{Q_PART}
 \end{aligned}$$

- **Q_ENTH** The change in the sensible enthalpy of the gas. ρ is the density of the gas (kg/m^3). h_s is the sensible enthalpy of the gas (kJ/kg). The volume integral is over the entire domain.
- **HRR** The heat release rate of the fire (kW) resulting from gas phase combustion.
- **HRR_OX** The heat release rate (kW) of any solid phase oxidation reactions. This should be summed with the **HRR** column when comparing results to heat release measurements obtained from oxygen consumption calorimetry, as the additional oxygen sink from solid phase reactions will be lumped into the calorimetry measurement. This term does not appear in Eq. (22.20) because the effect of solid phase reactions on the gas phase occurs implicitly via sensible enthalpy transport (**Q_CONV**) and heat transfer (**Q_COND**, **Q_RADI**) at the solid surfaces.
- **Q_RADI** The thermal radiation *into* the domain from the exterior boundary or particles. $\dot{\mathbf{q}}_r''$ is the radiation heat flux vector (kW/m^2). Its divergence represents the net radiative emission from a volume of gas. Typically, **Q_RADI** has a negative value, meaning that a fire or hot gases radiate energy out of the domain. $\dot{q}_{p,r}$ is the radiation absorbed by a droplet or particle (kW). This term is added to **Q_RADI** and subtracted from **Q_PART** because it is implicitly included in $\nabla \cdot \dot{\mathbf{q}}_r''$ and needs to be separated off for the purpose of explicitly accounting for it in the energy budget.
- **Q_CONV** The flow of sensible enthalpy *into* the computational domain. $\dot{m}_{p,\alpha}$ is the production rate of gas species α from a solid particle or liquid droplet (kg/s). $h_{s,\alpha}$ is the sensible enthalpy of gas species α (kJ/kg). ρ is the gas density (kg/m^3), \mathbf{u} is the velocity vector (m/s). h_s is the sensible enthalpy of the gas. If the gas is flowing out of the domain, $\mathbf{u} \cdot d\mathbf{S}$ is positive.
- **Q_COND** The convective heat flux *into* the computational domain. \dot{q}_c'' is the heat convected from the gas to a surface. If the gas is relatively hot and the surfaces/particles/droplets relatively cool, **Q_COND**

is negative. At `OPEN` boundaries, Q_{COND} is $\int k \nabla T \cdot d\mathbf{S}$, where k (kW/(m·K)) is the turbulent thermal conductivity of the gas and ∇T is the temperature gradient across the open boundary. $\dot{q}_{p,w}$ is the energy transferred from a solid surface (`wall`) to a droplet or particle adhering to it. Notice that it is subtracted off in Q_{PART} because it makes no contribution to the energy of the gas.

- Q_{PRES} The work done by volumetric expansion (kW).
- Q_{PART} The rate of energy gained by the gas from liquid droplets or solid particles. $\dot{q}_{p,r}$ is the radiation absorbed by a droplet or particle (kW). $\dot{q}_{p,c}$ is the energy transferred via convection from the gas to a droplet or particle (kW). $\dot{q}_{p,w}$ is the energy transferred from a solid surface (`wall`) to a droplet or particle adhering to it.

An additional column, Q_{TOTAL} , includes the sum of the terms on the right hand side of the equation. Note that this summation does not include `HRR_OX`, as explained above. Ideally, this sum should equal the term on the left, Q_{ENTH} . All terms are reported in units of kW.

The other columns in the `CHID_hrr.csv` file contain the total burning rate of fuel, in units of kg/s, and the zone pressures. Note that the reported value of the burning rate is not adjusted to account for the possibility that each individual material might have a different heat of combustion. For this reason, it is not always the case that the reported total burning rate multiplied by the gas phase heat of combustion is equal to the reported heat release rate.

Note that the volume integrations in Eq. (22.20) are performed over the entire domain. The differential, dV , is the product of the local grid cell dimensions, $dx dy dz$. For the special case of two-dimensional cylindrical coordinates, $dV = r dr d\theta dz$, where $r = x$, $dr = dx$, and $d\theta = dy$. In the 2D Cartesian case (1 cell in the y direction, `CYLINDRICAL=.FALSE.`), the cell volumes and areas are divided by dy so that the column output is a value per unit length (e.g., kW/m). In the 2D cylindrical case, the cell volumes are divided by dy (which represents radians in this case) and multiplied by 2π to account for complete integration in the θ direction. The resulting value is nominally the same as the corresponding 3D problem. An example is shown in Fig. 22.3.

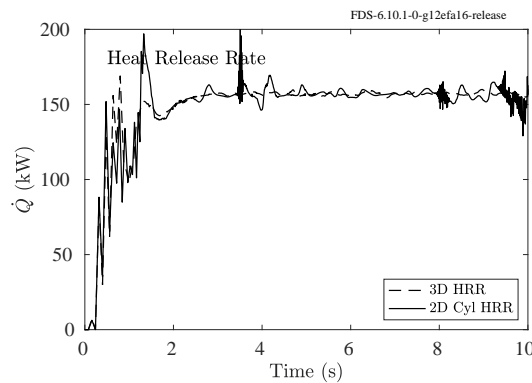


Figure 22.3: Heat release rate for the `test_hrr_2d_cyl` test case compared with the corresponding 3D problem.

As a test of the energy balance, a sample case called `Pressure_Solver/hallways.fds` simulates a fire near the end of five connected hallways. The other end of the hallways is open. As is seen in Fig. 22.4, the quantities Q_{ENTH} and Q_{TOTAL} are very closely matched, indicating that the sources of energy loss and gain are properly added and subtracted from the energy equation. As expected, the net energy gain/loss eventually goes to zero as the compartment reaches a quasi-steady state.

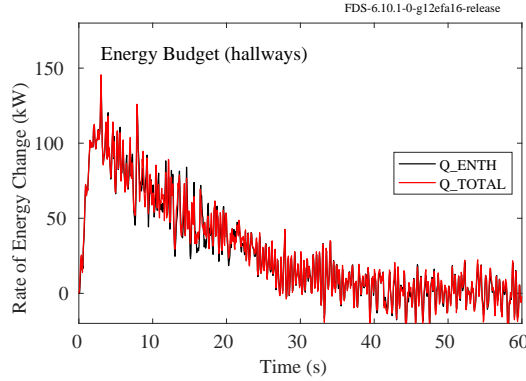


Figure 22.4: The energy budget for the `hallways` test case.

22.10.2 Gas Species Mass

If you set `MASS_FILE=T` on the `DUMP` line, a file called `CHID_mass.csv` is produced that lists the total mass of all gas species as a function of time. This flag is `F` by default because the calculation of all gas species in all mesh cells is time-consuming. The parameter `DT_MASS` or `RAMP_MASS` controls the frequency of output.

22.10.3 Mass Loss Rates

The rate at which gas species are generated at solid surfaces, solid particles, or liquid droplets is recorded in the file `CHID_hrr.csv`. This file is automatically generated; the only input parameters associated with it are `DT_HRR` and `RAMP_HRR` on the `DUMP` line. This file contains quantities related to the energy budget (Sec. 22.10.1), and the pressures of user-designated pressure zones (Sec. 22.10.4). The generation rates of all lumped gas species are given in units of `kg/s`.

22.10.4 Zone Pressures

If you specify pressure `ZONES`; that is, specific areas of the domain where the background pressure rises above ambient, the pressure of each zone is listed in the file `CHID_hrr.csv`.

22.10.5 Visibility and Obscuration

If you are performing a fire calculation using the simple chemistry approach, the smoke is tracked along with all other major products of combustion. The most useful quantity for assessing visibility in a space is the *light extinction coefficient*, K [83]. The intensity of monochromatic light passing a distance L through smoke is attenuated according to

$$I/I_0 = e^{-KL} \quad (22.21)$$

The light extinction coefficient, K , is a product of a mass specific extinction coefficient, K_m , which is fuel dependent, and the density of smoke particulate, ρY_S

$$K = K_m \rho Y_S \quad (22.22)$$

Devices that output a % obscuration such as a `DEVC` with a `QUANTITY` of `'ASPIRATION'`, `'CHAMBER_OBSCURATION'` (smoke detector), or `'PATH_OBSCURATION'` (beam detector) are discussed respectively in Sec. 18.3.7, Sec. 18.3.5, and Sec. 18.3.6.

Estimates of visibility through smoke can be made by using the equation

$$S = C/K \quad (22.23)$$

where C is a non-dimensional constant characteristic of the type of object being viewed through the smoke, i.e., $C = 8$ for a light-emitting sign and $C = 3$ for a light-reflecting sign [83]. Since K varies from point to point in the domain, the visibility S does as well.

Three parameters control smoke production and visibility. The first is the `SOOT_YIELD` on the `REAC` line, defined as the fraction of fuel mass that is converted to soot if the simple chemistry approach is being used. The second parameter, `MASS_EXTINCTION_COEFFICIENT`, is the K_m in Eq. (22.22). It is defined on one or more of the `SPEC` lines for the various light absorbing gas species. When using the simple chemistry combustion model, you can change the default mass extinction coefficient by adding a line to the input file of the form: `&SPEC ID='SOOT', MASS_EXTINCTION_COEFFICIENT=..., LUMPED_COMPONENT_ONLY=T /`. Its default value is 8700 m²/kg, a value suggested for flaming combustion of wood and plastics⁷. The third parameter, `VISIBILITY_FACTOR` on the `MISC` line, is the constant C in Eq. (22.23). It is 3 by default.

The gas phase output quantity '`EXTINCTION_COEFFICIENT`' is K . A similar quantity is the '`OPTICAL_DENSITY`', $D = K/2.3$, the result of using \log_{10} in the definition

$$D \equiv -\frac{1}{L} \log_{10} \left(\frac{I}{I_0} \right) = K \log_{10} e \quad (22.24)$$

The visibility S is output via the `QUANTITY` called '`VISIBILITY`'. Note that, by default, the visibility is associated with the smoke that is implicitly defined by the simple chemistry model. However, this quantity can also be associated with an explicitly defined species via the inclusion of a `SPEC_ID`. In other words, you can specify the output quantity '`VISIBILITY`' along with a `SPEC_ID`. This does not require that you do a simple chemistry calculation; only that you have specified the given species via a separate `SPEC` line. You can specify a unique `MASS_EXTINCTION_COEFFICIENT` on the `SPEC` line as well.

Note that FDS cannot report a visibility of infinity, but rather reports a `MAXIMUM_VISIBILITY` that you can control via the `MISC` line. The default is 30 m.

22.10.6 Flame Height and Flame Tilt

The “flame height,” and to some extent the “flame tilt,” are common quantities used to assess the hazard from a large fire. FDS does not output these quantities directly, but there is a combination of devices and controls that yield these values. This is all best explained with an example. Suppose you are simulating a fire in a domain that is 10 m high with a grid resolution of 0.2 m. The fire is centered about the point $(x,y) = (0,0)$ at $z = 0$. The following input lines compute the flame height and tilt:

```
&DEVC ID='HRR', Z_ID='z', XBP=0.0,0.0,0.0,0.0,0.1,9.9, QUANTITY='HRRPUV',
      SPATIAL_STATISTIC='VOLUME INTEGRAL', DX=5, DY=5, DZ=0.1, POINTS=100,
      STATISTICS_START=10. /
&CTRL ID='H', FUNCTION_TYPE='PERCENTILE', INPUT_ID='HRR', PERCENTILE=0.97 /
&DEVC ID='x_max', XB=-5,5,-5,5,9.0,9.2, QUANTITY='TEMPERATURE',
      SPATIAL_STATISTIC='MAXLOC X' /
&DEVC ID='y_max', XB=-5,5,-5,5,9.0,9.2, QUANTITY='TEMPERATURE',
      SPATIAL_STATISTIC='MAXLOC Y' /
&CTRL ID='x2', FUNCTION_TYPE='POWER', INPUT_ID='x_max', 'CONSTANT', CONSTANT=2 /
&CTRL ID='y2', FUNCTION_TYPE='POWER', INPUT_ID='y_max', 'CONSTANT', CONSTANT=2 /
&CTRL ID='r2', FUNCTION_TYPE='SUM', INPUT_ID='x2', 'y2' /
&CTRL ID='r', FUNCTION_TYPE='POWER', INPUT_ID='r2', 'CONSTANT', CONSTANT=0.5 /
```

⁷For most flaming fuels, a suggested value for K_m is 8700 m²/kg \pm 1100 m²/kg at a wavelength of 633 nm [84]

```

&CTRL ID='r/9', FUNCTION_TYPE='DIVIDE', INPUT_ID='d','CONSTANT', CONSTANT=9 /
&CTRL ID='tilt_rad', FUNCTION_TYPE='ATAN', INPUT_ID='r/9' /
&CTRL ID='tilt_deg', FUNCTION_TYPE='MULTIPLY', INPUT_ID='tilt_rad','CONSTANT',
    CONSTANT=57.296 /
&CTRL ID='cos_theta', FUNCTION_TYPE='COS', INPUT_ID='tilt_rad' /
&CTRL ID='L', FUNCTION_TYPE='DIVIDE', INPUT_ID='H','cos_theta' /
&DEVC ID='L_F', CTRL_ID='L', XYZ=0,0,0, QUANTITY='CONTROL VALUE',
    TEMPORAL_STATISTIC='RUNNING AVERAGE', STATISTICS_START=10, UNITS='m' /
&DEVC ID='tilt', CTRL_ID='tilt_deg', XYZ=0,0,0, QUANTITY='CONTROL VALUE',
    TEMPORAL_STATISTIC='RUNNING AVERAGE', STATISTICS_START=10, UNITS='deg' /

```

- ID='HRR' Compute a vertical array of heat release rates over horizontal slices that are one cell thick.
- ID='H' Take the function ('z','HRR') and compute the height below which 97 % of the fire's energy is released.
- ID='x_max' Find the x coordinate of the point of maximum temperature within the volume defined by XB. Do the same for the y coordinate.
- ID='x2','y2','r2','r','r/9' Determine the radial distance from the origin to the point of maximum temperature. Divide this distance by the height of the slice within which you located the maximum temperature, $z = 9$.
- ID='tilt_rad','tilt_deg' Calculate the arctangent of 'r/9' to determine the tilt angle in radians. Then convert this value to degrees.
- ID='cos_theta' Calculate the cosine of the tilt angle.
- ID='L' Divide the vertical flame height by the cosine of the tilt angle to arrive at the length of the flame.
- ID='L_F','tilt' Print out the flame length and tilt angle to the device file using a running average to smooth the noisy data.

To better understand the control logic, see Sec. 18.4. Pay particular attention to the averaging periods, start times, etc. You do not want to begin a running average until the fire plume has stabilized. Practice with some simple cases before implementing this in a complex simulation.

22.10.7 Layer Height and the Average Upper and Lower Layer Temperatures

Fire protection engineers often need to estimate the location of the interface between the hot, smoke-laden upper layer and the cooler lower layer in a burning compartment. Relatively simple fire models, often referred to as *two-zone models*, compute this quantity directly, along with the average temperature of the upper and lower layers. In a computational fluid dynamics (CFD) model like FDS, there are not two distinct zones, but rather a continuous profile of temperature. Nevertheless, there are methods that have been developed to estimate layer height and average temperatures from a continuous vertical profile of temperature. One such method [85] is as follows: Consider a continuous function $T(z)$ defining temperature T as a function of height above the floor z , where $z = 0$ is the floor and $z = H$ is the ceiling. Define T_u as the upper layer temperature, T_l as the lower layer temperature, and z_{int} as the interface height. Compute the quantities:

$$(H - z_{\text{int}}) T_u + z_{\text{int}} T_l = \int_0^H T(z) \, dz = I_1$$

$$(H - z_{\text{int}}) \frac{1}{T_u} + z_{\text{int}} \frac{1}{T_1} = \int_0^H \frac{1}{T(z)} dz = I_2$$

Solve for z_{int} :

$$z_{\text{int}} = \frac{T_1 (I_1 I_2 - H^2)}{I_1 + I_2 T_1^2 - 2 T_1 H} \quad (22.25)$$

Let T_1 be the temperature in the lowest mesh cell and perform the numerical integration of I_1 and I_2 . T_u is defined as the average upper layer temperature via

$$(H - z_{\text{int}}) T_u = \int_{z_{\text{int}}}^H T(z) dz \quad (22.26)$$

Further discussion of similar procedures can be found in Ref. [86].

The quantities 'LAYER HEIGHT', 'UPPER TEMPERATURE' and 'LOWER TEMPERATURE' can be designated via DEVC lines in the input file. For example, the line:

```
&DEVC XB=2.0,2.0,3.0,3.0,0.0,3.0, QUANTITY='LAYER HEIGHT', ID='whatever' /
```

produces a time history of the smoke layer height at $x = 2$ and $y = 3$ between $z = 0$ and $z = 3$.

22.10.8 Thermocouples

The output quantity THERMOCOUPLE is the temperature of a modeled thermocouple. The thermocouple temperature lags the true gas temperature by an amount determined mainly by its bead size. It is found by solving the following equation for the thermocouple temperature, T_{TC} [87]

$$\frac{D_{\text{TC}}}{6} \rho_{\text{TC}} c_{\text{TC}} \frac{dT_{\text{TC}}}{dt} = \epsilon_{\text{TC}} (U/4 - \sigma T_{\text{TC}}^4) + h(T_g - T_{\text{TC}}) \quad (22.27)$$

where ϵ_{TC} is the emissivity of the thermocouple, U is the integrated radiative intensity, T_g is the true gas temperature, and h is the heat transfer coefficient to a small sphere, $h = k\text{Nu}/D_{\text{TC}}$. The bead DIAMETER, EMISSIVITY, DENSITY, and SPECIFIC_HEAT are given on the associated PROP line. To over-ride the calculated value of the heat transfer coefficient, set HEAT_TRANSFER_COEFFICIENT on the PROP line (W/(mK)). The default value for the bead diameter is 0.001 m. The default emissivity is 0.85. The default values for the bead density and specific heat are that of nickel; 8908 kg/m³ and 0.44 kJ/kg/K, respectively. See the discussion on heat transfer to a water droplet in the Technical Reference Guide for details of the convective heat transfer to a small sphere.

The above discussion is appropriate for a so-called “bare bead” thermocouple, but often thermocouples are shielded or sheathed in various ways to mitigate the effect of thermal radiation. In such cases, there is no obvious bead diameter and there may be multiple metals and air gaps in the construction. Usually, the manufacturer provides a time constant, which is defined as the time required for the sensor to respond to 63.2 % of its total output signal when suddenly plunged into a warm air stream flowing at 20 m/s. The analysis and testing is typically done at relatively low temperature, in which case the radiation term in Eq. (22.27) can be neglected and the time constant, τ , can be defined in terms of effective thermal properties and an effective diameter:

$$\tau = \frac{D_{\text{eff}} \rho_{\text{eff}} c_{\text{eff}}}{6h} ; \quad h = \frac{k\text{Nu}}{D_{\text{eff}}} ; \quad \text{Nu} = 2 + 0.6\text{Re}^{1/2}\text{Pr}^{1/3} ; \quad \text{Re} = \frac{\rho \|\mathbf{u}\| D_{\text{eff}}}{\mu} \quad (22.28)$$

For a given value of the time constant, τ , and effective thermal properties, Eq. (22.28) can be solved implicitly for the effective diameter. The TIME_CONSTANT is specified on a PROP line which is identified by the DEV

line using a `PROP_ID`. If you specify the `TIME_CONSTANT`, you can still specify the effective `EMISSIVITY`, `DENSITY`, `SPECIFIC_HEAT`, and `HEAT_TRANSFER_COEFFICIENT` as well, but not the `DIAMETER` because this will be calculated automatically. In most cases, it is sufficient to simply specify the `TIME_CONSTANT`.

Figure 22.5 shows the results of a simple test case called `thermocouple_time_constant` whose input file is in the `Heat_Transfer` samples folder. Three thermocouples with given time constants of 0.5 s, 3.0 s, and 8.0 s are suddenly subjected to a 20 m/s air stream of 30 °C. It is expected that each TC should reach a temperature of 26.32 °C at their given time constants.

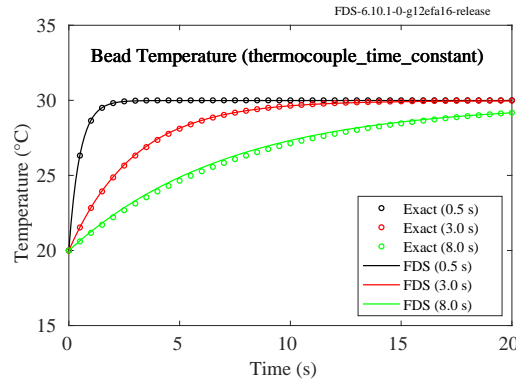


Figure 22.5: Temperature response of thermocouples with time constants of 0.5 s, 3.0 s, and 8.0 s compared with the exact solution of Eq. (22.27).

22.10.9 Volume Flow

A common quantity related to heating and ventilation systems is *volume flow*, typically denoted \dot{V} and expressed in units of m³/s or ft³/min. The way volume flow output is specified depends on whether it is at a boundary or completely within the gas phase.

Volume Flow in the Gas Phase

Consider a flow of gases through a window located at $x = x_0$ and bounded by $y_1 \leq y \leq y_2$ and $z_1 \leq z \leq z_2$. Its volume flow is given by:

$$\dot{V}(t) = \int_{z_1}^{z_2} \int_{y_1}^{y_2} u(x_0, y, z) \, dy \, dz \approx \sum_{k=k_1}^{k_2} \sum_{j=j_1}^{j_2} u_{ijk} \, \delta y \, \delta z \quad (22.29)$$

To specify volume flow as an output, use an input line like the following:

```
&DEVC ID='Vdot', XB=X0,X0,Y1,Y2,Z1,Z2, QUANTITY='U-VELOCITY',
      SPATIAL_STATISTIC='AREA INTEGRAL' /
```

This produces a column in the `CHID_devic.csv` containing the time history of the volume flow (m³/s) through the window. Note that if the gas is flowing in the positive coordinate direction, the volume flow is positive. You can reverse the sign by adding `CONVERSION_FACTOR=-1` to the `DEVC` line. If you want to distinguish the inflow from the outflow through the same window, use the parameters `QUANTITY_RANGE(1:2)` to set upper or lower bounds. For example, if you only want to record the volume flow in the negative coordinate direction, set `QUANTITY_RANGE(2)=0`, meaning that positive values of velocity are not counted in the summation.

Volume Flow at a Boundary

To record the volume flow at a solid surface or vent, add a line like the following:

```
&DEVC ID='Vdot', XB=..., QUANTITY='NORMAL VELOCITY', SURF_ID='...',  
      SPATIAL_STATISTIC='SURFACE INTEGRAL' /
```

This line instructs FDS to sum the normal component of velocity over any surface with the given `SURF_ID` within the volume bounded by `XB`. Note that a flow entering into the computational domain is negative.

Even though you may only be interested in the volume flow through a plane, provide an actual volume using `XB` to ensure that even if the vent or obstruction is snapped to the nearest grid cell, the area over which the integral is summed will not fall outside of your specified volume limits.

22.10.10 Mass Flow

There are several output options for mass flux and mass flow depending on whether a gas or solid phase quantity is desired, and the level of precision desired.

Mass Flow in the Gas Phase

The mass flow of gas species α through a window located at $x = x_0$ and bounded by $y_1 \leq y \leq y_2$ and $z_1 \leq z \leq z_2$ is given by:

$$\dot{m}_\alpha(t) = \int_{z_1}^{z_2} \int_{y_1}^{y_2} \rho Y_\alpha u \, dy \, dz \approx \sum_{k=k_1}^{k_2} \sum_{j=j_1}^{j_2} \frac{\rho_{ijk} Y_{ijk} + \rho_{i+1,jk} Y_{i+1,jk}}{2} u_{ijk} \delta y \delta z \quad (22.30)$$

To instruct FDS to output this quantity, use the input line:

```
&DEVC ID='m-dot', XB=x0,x0,y1,y2,z1,z2, QUANTITY='MASS FLUX X', SPEC_ID='ALPHA',  
      SPATIAL_STATISTIC='AREA INTEGRAL' /
```

To limit the integration to only flow in the positive or negative coordinate direction, set `QUANTITY_RANGE(1)` or `QUANTITY_RANGE(2)` to zero. If you want the total mass flow, omit the `SPEC_ID`.

'MASS FLUX X' and its y and z counterparts are only estimates of the actual computed mass flux for two reasons.

1. These quantities use an estimate of the flux-limiting mass flux terms that are computed by FDS. Notice in Eq. (22.30) that the density ρ and mass fraction Y_α are approximated at the forward x face of cell ijk with a simple average. The actual discrete approximation is more complicated than this, but these flux-limiting mass flux terms are normally not saved outside of the routine where they are computed.
2. If a `SPEC_ID` is specified, Eq. (22.30) represents only the *advective* component of the total mass flux. In turbulent flows, the diffusive component, which includes turbulent diffusion, can be significant.

If you want more precise output that includes the diffusive component of the mass flux and the exact flux limiter for the advective component, specify 'TOTAL MASS FLUX X' rather than 'MASS FLUX X' (or the y or z coordinate analogs). If you want to get a breakdown of the components, specify 'ADVECTIVE MASS FLUX X' and 'DIFFUSIVE MASS FLUX X' as well. Take care to specify `XB` in unambiguous cell locations (i.e., not directly on a cell face or boundary). With a `DEVC` in cell (I, J, K) the flux will be reported from the $(I+1/2, J, K)$ face, etc., on the staggered grid.

Mass Flow at a Boundary

There are three quantities that may be used for recording the mass flux of a particular gas species at a boundary: 'MASS FLUX', 'MASS FLUX WALL', and 'TOTAL MASS FLUX WALL'. Each takes a SPEC_ID if a particular gas species is desired; otherwise the sum of all gas species is given. The difference between the outputs is that 'MASS FLUX' refers to the generation rate of a gas species or solid material component at a solid boundary, whether it be user-specified or the result of a pyrolysis model. 'MASS FLUX WALL', on the other hand, is the actual computed mass flux which might be slightly different than 'MASS FLUX' due to small numerical error in the normal component of velocity at a solid wall. That is, 'MASS FLUX WALL' is a direct computation of

$$\dot{m}_\alpha = \rho Y_\alpha u_n - \rho D_\alpha \partial Y_\alpha / \partial n \quad (22.31)$$

where \mathbf{n} is the normal direction of the surface. The third mass flux output option is called 'TOTAL MASS FLUX WALL'. This quantity is very similar to 'MASS FLUX WALL' except that it combines the predicted and corrected mass flux over a single time step. It is the boundary analog of the gas phase quantities 'TOTAL MASS FLUX X/Y/Z'.

For each form of the boundary mass flux, a positive value means that mass is entering the computational domain. These quantities may be applied at a solid surface or vent. 'MASS FLUX WALL' and 'TOTAL MASS FLUX WALL' can also be applied at an OPEN boundary.

To demonstrate the difference between these various forms of mass flux output, consider Fig. 22.6. These plots show the spatially integrated mass flux of gas from a solid surface, where the mass flux has been ramped up linearly for 10 s, held steady for 20 s, and ramped down to zero linearly for 10 s. The 'MASS FLUX' output merely mimics this specified curve. The 'MASS FLUX WALL' and 'TOTAL MASS FLUX WALL' output exhibit fluctuations caused by the fact that the mass flux is not perfectly applied at the solid surface—there are small fluctuations in the normal component of velocity caused by the inexact velocity-pressure coupling at the boundary. These fluctuations are minor and average out close to the specified value, but sometimes it is useful to use the more exact output quantities for diagnostic purposes.

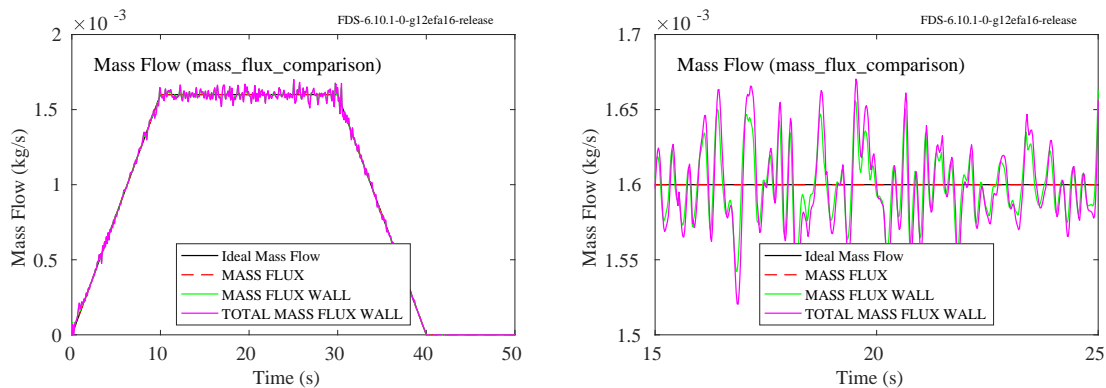


Figure 22.6: (Left) Mass flow rate of a gas from a solid boundary. (Right) Close-up view.

22.10.11 Enthalpy Flow

Enthalpy Flow in the Gas Phase

The flow of enthalpy through a window located at $x = x_0$ and bounded by $y_1 \leq y \leq y_2$ and $z_1 \leq z \leq z_2$ is given by:

$$\dot{q}(t) = \int_{z_1}^{z_2} \int_{y_1}^{y_2} \left[\rho(h_s(T) - h_s(T_\infty))u - k \frac{\partial T}{\partial x} \right] dy dz \approx \sum_{k=k_1}^{k_2} \sum_{j=j_1}^{j_2} \left[\rho_{i+\frac{1}{2},jk} \left(h_s(T_{i+\frac{1}{2},jk}) - h_s(T_\infty) \right) u_{ijk} - k_{ijk} \frac{T_{i+1,jk} - T_{ijk}}{\delta x} \right] \delta y \delta z \quad (22.32)$$

To instruct FDS to output this quantity, use the input line:

```
&DEVC ID='q-dot', XB=x0,x0,y1,y2,z1,z2, QUANTITY='ENTHALPY FLUX X',
      SPATIAL_STATISTIC='AREA INTEGRAL' /
```

Enthalpy Flow at a Boundary

'ENTHALPY FLUX WALL' is the equivalent of 'ENTHALPY FLUX X/Y/Z' at a boundary. It is the rate of energy advected *into* the domain at a vent or open boundary:

$$\dot{q}_w'' = -\rho_w h_s(T_w) \mathbf{u} \cdot \mathbf{n} - k \frac{\partial T}{\partial n} \quad (22.33)$$

where n is the coordinate direction pointing into the boundary; that is, out of the computational domain. Note that the enthalpy flux at a boundary is defined using the density, temperature, and sensible enthalpy of the gas mixture at the boundary, and is not relative to ambient conditions like its gas phase counterparts. For this reason, this quantity is often used in energy budget calculations to track the flow of enthalpy into and out of the computational domain. The quantity is positive for gas flow into the domain.

22.10.12 Heat Flux

There are various output quantities related to the thermal exposure of solid surfaces.

- 'TOTAL HEAT FLUX' The rate of radiative and convective energy absorbed at a solid surface:

$$\dot{q}_{\text{tot}}'' = \epsilon_s (\dot{q}_{\text{inc}}'' - \sigma T_s^4) + h_c (T_g - T_s) \quad (22.34)$$

where \dot{q}_{inc}'' is the *incident* radiative heat flux, ϵ_s is the surface emissivity, h_c is the convective heat transfer coefficient, T_s is the surface temperature, and T_g is the gas temperature in the vicinity of the surface. The convective heat transfer coefficient is calculated by FDS using the specified surface properties and the calculated near-boundary flow characteristics.

- 'RADIATIVE HEAT FLUX' The net radiative component of Eq. (22.34), $\dot{q}_r'' = \epsilon_s (\dot{q}_{\text{inc}}'' - \sigma T_s^4)$.
- 'CONVECTIVE HEAT FLUX' The convective component of Eq. (22.34), $\dot{q}_c'' = h_c (T_g - T_s)$.
- 'CONVECTIVE HEAT FLUX GAUGE' The convective component of Eq. (22.35), $\dot{q}_{c,\text{gauge}}'' = h_c (T_g - T_{\text{gauge}})$.
- 'INCIDENT HEAT FLUX' The term \dot{q}_{inc}'' in Eq. (22.34).

- 'GAUGE HEAT FLUX' This quantity simulates a measurement made with a water-cooled heat flux gauge:

$$\dot{q}_{\text{gauge}}'' = \epsilon_{\text{gauge}} (\dot{q}_{\text{inc}}'' - \sigma T_{\text{gauge}}^4) + h_c (T_g - T_{\text{gauge}}) \quad (22.35)$$

If the heat flux gauge used in an experiment has a temperature other than ambient or an emissivity other than 1, specify `GAUGE_TEMPERATURE` (T_{gauge} , °C) and `GAUGE_EMISSIVITY` (ϵ_{gauge}) on the `PROP` line associated with the device:

```
&DEVC ID='hf', XYZ=..., IOR=-2, QUANTITY='GAUGE HEAT FLUX', PROP_ID='hfp' /
&PROP ID='hfp', GAUGE_TEMPERATURE=80., GAUGE_EMISSIVITY=0.9 /
```

By default, the heat transfer coefficient, h_c , in Eq. (22.35) is calculated at the solid surface to which the device is attached, based on the specified surface properties and characteristics of the surrounding flow field. However, you may specify a fixed `HEAT_TRANSFER_COEFFICIENT` (W/(m²·K)) for the gauge on the `PROP` line.

- 'GAUGE HEAT FLUX GAS' The same as 'GAUGE HEAT FLUX', except that it can be located anywhere within the computational domain and not just at a solid surface. It also has an arbitrary `ORIENTATION` vector that points in any desired direction. The `ORIENTATION` vector need not be normalized, as in the following:

```
&DEVC ID='hf', QUANTITY='GAUGE HEAT FLUX GAS', XYZ=..., ORIENTATION=-1,1,0,
PROP_ID='my gauge' /
&PROP ID='my gauge', GAUGE_EMISSIVITY=0.85, HEAT_TRANSFER_COEFFICIENT=15. /
```

The `GAUGE_TEMPERATURE`, T_{gauge} , `GAUGE_EMISSIVITY`, ϵ_{gauge} , and `HEAT_TRANSFER_COEFFICIENT`, h_c (W/(m² K)), can be specified using a `PROP` line that is referenced by the `DEVC` line. Their default values are `TMPA`, 1, and 10, respectively.

Note that the parameter `SPATIAL_STATISTIC` is not appropriate for this quantity, meaning that you cannot integrate this quantity over a plane or volume. However, you can use the parameter `POINTS` to create a one-dimensional array of these devices (see Sec. 22.2.5).

- 'RADIOMETER' Similar to a water-cooled heat flux gauge, except that this quantity measures only the net radiative component:

$$\dot{q}_{\text{radiometer}}'' = \epsilon_{\text{gauge}} (\dot{q}_{\text{inc}}'' - \sigma T_{\text{gauge}}^4) \quad (22.36)$$

The `GAUGE_TEMPERATURE` (T_{gauge} , °C) and `GAUGE_EMISSIVITY` (ϵ_{gauge}) can be set on the `PROP` line associated with the device if their values are different than ambient and 1, respectively.

The quantity `RADIOMETER` is based on the *ellipsoidal radiometer* concept first proposed by Nils-Erik Gunnars and described in the Ref. [88]. The intent of such a device is to eliminate the contribution of convection from the measurement.

- 'RADIOMETER GAS' The same as 'RADIOMETER' described above, except it can be located anywhere within the computational domain and not just at a solid surface. It also has an arbitrary `ORIENTATION` vector that points in any desired direction, much like a heat flux gauge. The `ORIENTATION` vector need not be normalized, as in the following:

```
&DEVC ID='hf', QUANTITY='RADIOMETER GAS', XYZ=..., ORIENTATION=-1,1,0/
```

Note that the parameter `SPATIAL_STATISTIC` is not appropriate for this quantity, meaning that you cannot integrate this quantity over a plane or volume. However, you can use the parameter `POINTS` to create a one-dimensional array of these devices (see Sec. 22.2.5).

- 'RADIATIVE HEAT FLUX GAS' This output quantity is the same as 'RADIATIVE HEAT FLUX' described above, except it can be located anywhere within the computational domain and not just at a solid surface. It also has an arbitrary `ORIENTATION` vector that points in any desired direction, much like a heat flux gauge. The `ORIENTATION` vector need not be normalized, as in the following:

```
&DEVC ID='hf', QUANTITY='RADIATIVE HEAT FLUX GAS', XYZ=..., ORIENTATION=-1,1,0/
```

Note that the parameter `SPATIAL_STATISTIC` is not appropriate for this quantity, meaning that you cannot integrate this quantity over a plane or volume. However, you can use the parameter `POINTS` to create a one-dimensional array of these devices (see Sec. 22.2.5).

- 'RADIANCE' The radiation intensity at the point, \mathbf{x} , and direction angle, \mathbf{s} , where the summation is over all spectral bands:

$$I(\mathbf{x}, \mathbf{s}) = \sum_{n=1}^N I_n(\mathbf{x}, \mathbf{s}) \quad (22.37)$$

This output quantity is specified in a similar way as 'RADIATIVE HEAT FLUX GAS' in that a surrogate target particle is placed at the point `XYZ` with an `ORIENTATION` vector pointing in any desired direction.

```
&DEVC ID='rad2', QUANTITY='RADIANCE', XYZ=..., ORIENTATION=-1,0,0 /
```

The units for 'RADIANCE' are $\text{kW}/\text{m}^2/\text{sr}$.

For `QUANTITY` types ending in `GAS`, a view angle for the radiation component of the device can be set using `VIEW_ANGLE` on `PROP`. Wall devices can only have a 180° view angle.

22.10.13 Adiabatic Surface Temperature

FDS includes a calculation of the adiabatic surface temperature (AST), a way of expressing the thermal exposure of a solid surface. Following the idea proposed by Ulf Wickström [89], T_{AST} is the surface temperature for which the total heat flux defined by Eq. (22.34) is zero. The following equation can be solved using an analytical solution given by Malendowski [90]:

$$\epsilon_s (\dot{q}_{\text{inc}}'' - \sigma T_{\text{AST}}^4) + h_c (T_g - T_{\text{AST}}) = 0 \quad (22.38)$$

where, \dot{q}_{inc}'' is the *incident* radiative heat flux onto the surface, ϵ_s is the surface emissivity, h_c is the convective heat transfer coefficient, and T_g is the surrounding gas temperature.

To output the AST at a single point on a solid surface, include a device as follows:

```
&DEVC ID='AST', XYZ=..., IOR=..., QUANTITY='ADIABATIC SURFACE TEMPERATURE' /
```

Note that `IOR` specifies the orientation of the surface. To produce a contour plot of AST on all solid surfaces, include a boundary file as follows:

```
&BNDF QUANTITY='ADIABATIC SURFACE TEMPERATURE' /
```

The usefulness of the AST is that it represents an effective exposure temperature that can be passed on to a more detailed model of the solid object. It provides the gas phase thermal boundary condition in a single quantity, and it is not affected by the uncertainty associated with the solid phase heat conduction model within FDS. Obviously, the objective in passing information to a more detailed model is to get a better prediction of the solid temperature (and ultimately its mechanical response) than FDS can provide. To reinforce this notion, you can output the adiabatic surface temperature even when there is no actual solid surface in your model using the following lines:

```
&DEVC ID='AST', XYZ=..., QUANTITY='ADIABATIC SURFACE TEMPERATURE GAS',
      ORIENTATION=0.707,0.0,0.707, PROP_ID='props' /
&PROP ID='props', EMISSIVITY=0.9, HEAT_TRANSFER_COEFFICIENT=10. /
```

This output indicates the maximum achievable solid surface temperature at the given location `XYZ` that is *not* actually in the vicinity of any solid surface, facing in any direction as indicated by the `ORIENTATION` vector. Note that you *must* set the `EMISSIVITY` and `HEAT_TRANSFER_COEFFICIENT` ($\text{W}/(\text{m}^2\cdot\text{K})$) on the `PROP` line because there is no actual solid surface from which to infer these values. In fact, FDS creates a Lagrangian particle that records the AST at the desired location. FDS creates an `INIT` line to position the particle. If you want to output other quantities at this point that are related to the AST, you can create another `DEVC` at the same location and use `INIT_ID` to identify the `DEVC` line for the `ADIABATIC SURFACE TEMPERATURE`. For example, you can output the heat transfer coefficient used in calculating the AST as follows:

```
&DEVC ID='AST', XYZ=..., QUANTITY='ADIABATIC SURFACE TEMPERATURE GAS', ... /
&DEVC ID='HTC', XYZ=..., QUANTITY='HEAT TRANSFER COEFFICIENT', INIT_ID='AST' /
```

Example: AST vs. Surface Temperature

The test case called `adiabatic_surface_temperature.fds` in the `Radiation` folder simulates a 0.1 mm steel plate being heated by a thermal plume. The plate is perfectly insulated (`BACKING='INSULATED'` on the `SURF` line) and its steady-state temperature should be equivalent to its adiabatic surface temperature or AST. The left plot of Fig. 22.7 shows the plate temperature rising toward the AST, much as an actual plate thermometer would. The right plot simply shows the AST calculated using the `QUANTITY 'ADIABATIC SURFACE TEMPERATURE'` applied via a `DEVC` at the plate surface, and the AST calculated using the `QUANTITY 'ADIABATIC SURFACE TEMPERATURE GAS'` applied via a `DEVC` positioned just in front of the plate. The idea of the latter device would be to record the AST even if the plate were not actually represented in the simulation. For these two recordings of the AST to be identical, the plate `SURFACE` conditions and the `AST-Gas` `PROPERTIES` must both include the same explicitly-defined `HEAT_TRANSFER_COEFFICIENT` and `EMISSIVITY`.

22.10.14 Extracting Detailed Radiation Data

FDS solves the radiation transport equation (RTE) using a finite volume method in which the radiation intensity is computed over a discrete number of solid angles. Because there are approximately 100 angles by default, the angle-specific intensity values are not saved in the gas phase cells—only at boundaries. However, there is a way to save these values into a file, which might be useful for transferring data to another model or diagnosing problems with the RTE. This form of output is only appropriate when the default gray gas radiation model is used, not the narrow-band model.

Data is saved within rectangular blocks of cells using one or more input lines of the form:

```
&RADF XB=..., I_STEP=2, J_STEP=3, K_STEP=1 /
```

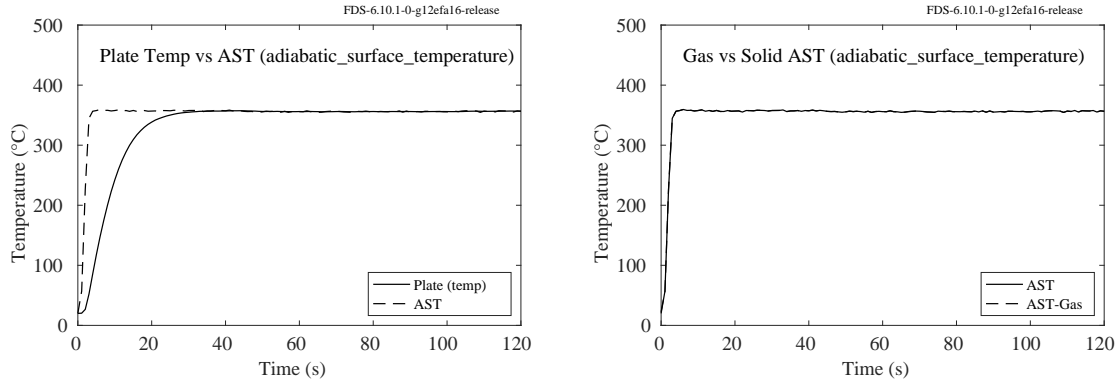


Figure 22.7: (Left) Surface temperature vs. adiabatic surface temperature of an insulated plate. (Right) AST recorded with a device positioned on the plate surface (AST) and one just off the surface (AST-Gas).

This line directs FDS to write out the radiation intensity, I_{ijk}^l (W/m²/sr), for each cell with indices ijk that is bounded by XB and for each solid angle l . You can skip cells using the parameters I_STEP, J_STEP, and K_STEP, each of which default to 1. The print outs are done every DT_RADF s. This parameter is listed on the DUMP line. The time interval, DT_RADF, has no default value and must be set.

Each output file is in text, not binary, format (for example, CHID_radf_n.txt). The format of the file is shown in Fig. 22.8. NSTEPS is the number of TIMES that data is written out. NP is the number of points within the block designated by XB on the RADF line of the input file. XYZ_INTENSITIES are the coordinates of the NP points. NI is the number of solid angles, which are listed under XYZ_DIRECTIONS. A direction vector is a normalized form of the discretized solid angle vector, (D_x^l, D_y^l, D_z^l) , that is described in the Radiation chapter of the FDS Technical Reference Guide, Volume 1 [3].

Each TIME the intensities are written out, each cell in the designated block will list its temperature in degrees K, a placeholder value of 0, and then the intensities at the NI solid angles in units of W/m²/sr.

22.10.15 Flame Temperature

The output quantity 'TEMPERATURE' reports the average gas temperature within a gas phase grid cell. In a typical fire simulation, combustion within the grid cell occurs over a flame sheet that is not resolvable on the grid. Because the source term in the radiation transport equation is a function of the gas temperature to the fourth power, the temperature in cells where combustion occurs is given an effective value so that the user-specified RADIATIVE_FRACTION is achieved. The output quantity 'EFFECTIVE FLAME TEMPERATURE' displays this modified temperature.

This quantity should not be taken as the adiabatic flame temperature. More details are given in Section 14.1.3.

22.10.16 Detailed Spray Properties

Detailed experimental measurements of sprays⁸ using Phase Doppler Particle Analysis (PDPA) provide information on the droplet size distribution, speed and concentration. A special device type is defined via a DEVC line to simulate the PDPA measurement. The actual quantity to measure, and the details of the measurement are defined using an associated PROP line.

⁸The 'PDPA' output quantities generally apply to liquid droplets. However, they also can be applied to solid particles.


```

NSTEPS
  2

TIMES
  0.00
 10.00

NP
 3150

XYZ_INTENSITIES
  2.350  1.350  0.050
  .
  .
  .

NI
 104

XYZ_DIRECTIONS
  0.138  0.138  0.981
 -0.138  0.138  0.981
  .
  .
  .

TIME
  0.00

INTENSITIES
293.15  0.00      133.17      133.17      133.17      133.17      133.17  ...
  .
  .
  .

```

Figure 22.8: Format of RADF output file.

By default, the PDPA device output at time t is computed as a time integral between t_s (PDPA_START) and t_e (PDPA_END):

$$F(t) = \frac{1}{t_e - t_s} \int_{t_s}^{t_e} f(t') dt' \quad (22.39)$$

but instantaneous values can be obtained by setting PDPA_INTEGRATE=.FALSE. on the corresponding PROP line, in which case

$$F(t) \equiv f(t) \quad (22.40)$$

The function $f(t)$ has two forms:

$$f_1(t) = \left(\frac{\sum_i n_i D_i^m \phi}{\sum_i n_i D_i^n} \right)^{\frac{1}{m-n}} ; \quad f_2(t) = \frac{\sum_i n_i \phi}{V} \quad (22.41)$$

where n_i is the number of real particles represented by the single simulated particle, D_i is the particle diameter, and ϕ is the QUANTITY to be measured. In each case, the summation goes over all the particles within a sphere with radius PDPA_RADIUS centered at the location given by XYZ on the DEVC line.

The first function, $f_1(t)$, is used for the computation of various mean diameters, with associated properties defined using the following keywords on the PROP line:

- PDPA_M is the exponent m in the numerator.
- PDPA_N is the exponent n in denominator.

If $m = n$, the exponent $1/(m - n)$ is removed from the formula.

The second function, $f_2(t)$, is used to compute mass and energy-related quantities without the diameter weighting. The concentrations are based on the sampling volume, V , defined by PDPA_RADIUS. The options for the QUANTITY, ϕ , are listed in Table 22.2.

Table 22.2: Output quantities available for PDPA.

QUANTITY	ϕ	f	Units of f
'DIAMETER' (default)	1	f_1	μm
'ENTHALPY'	$(4/3)\pi\rho_i r_i^3 (c_{p,i}(T_i)T_i - c_{p,i}(T_m)T_m)$	f_2	kJ/m^3
'PARTICLE FLUX X'	$(4/3)\pi\rho_i r_i^3 u_i$	f_2	$\text{kg}/(\text{m}^2 \text{s})$
'PARTICLE FLUX Y'	$(4/3)\pi\rho_i r_i^3 v_i$	f_2	$\text{kg}/(\text{m}^2 \text{s})$
'PARTICLE FLUX Z'	$(4/3)\pi\rho_i r_i^3 w_i$	f_2	$\text{kg}/(\text{m}^2 \text{s})$
'U-VELOCITY'	u_i	f_1	m/s
'V-VELOCITY'	v_i	f_1	m/s
'W-VELOCITY'	w_i	f_1	m/s
'VELOCITY'	$(u_i^2 + v_i^2 + w_i^2)^{\frac{1}{2}}$	f_1	m/s
'TEMPERATURE'	T_i	f_1	$^{\circ}\text{C}$
'MASS CONCENTRATION'	$(4/3)\pi\rho_i r_i^3$	f_2	kg/m^3
'NUMBER CONCENTRATION'	1	f_2	particles/m^3

* T_m is the melting temperature of the associated species.

It is also possible to output histograms of PDPA output quantities. When HISTOGRAM is set to T on the PROP line, normalized histogram bin counts are output to a comma-separated value (.csv) file from all devices (DEVC) associated with this PROP line. The number of bins and the limits of the histogram are

controlled by parameters on the `PROP` line. The value used in creating the histogram is $D_i^m \phi$. Note that when the specified histogram `QUANTITY` is 'DIAMETER', the `HISTOGRAM_LIMITS` must be given in meters, not microns. Values falling outside the histogram limits are included in the counts of the first and last bins. Cumulative distributions can be output by setting `HISTOGRAM_CUMULATIVE=T` on the `PROP` line. To output unnormalized counts or cumulative distribution, set `HISTOGRAM_NORMALIZE` to `F`. Note, however, that the counts correspond to the super droplets/particles, not the numerical ones. Due to the stratified sampling technique used (see Sec. 15.3.3), the counts are not necessarily integers.

The histogram output file contains two-columns for each device. The first column provides the values of the bin center-points, and the second column provides the relative frequency. Volume and area based distributions can be output by setting the `PDPA_N` parameter on the `PROP` line to 3 and 2 respectively. Notice that this works differently from the mean diameter computation where the weighting is based on the `PDPA_M` parameter.

The properties of the PDPA device are defined using the following parameters on the `PROP` line:

- `PART_ID` Name of the particle group to limit the computation to. Do not specify to account for all particles.
- `PDPA_START` t_s , starting time of time integration in seconds. PDPA output is always a running average over time. As the spray simulation may contain some initial transient phase, it may be useful to specify the starting time of data collection.
- `PDPA_END` t_e , ending time of time integration in seconds.
- `PDPA_INTEGRATE` A logical parameter for choosing between time integrated or instantaneous values. `T` by default.
- `PDPA_RADIUS` Radius (m) of the sphere, centered at the device location, inside which the particles are monitored.
- `PDPA_NORMALIZE` Can be set `F` to force $V = 1$ in the formula for $f_2(t)$.
- `QUANTITY` Specified on `PROP` line for choosing the variable ϕ .
- `HISTOGRAM_NBINS` Number of bins used for the histogram.

The following example is used to measure the Sauter mean diameter, D_{32} , of the particle type 'water drops', starting from time 5 s.

```
&PROP ID='pdpa_d32'
      PART_ID='water drops'
      PDPA_M=3
      PDPA_N=2
      PDPA_RADIUS=0.01
      PDPA_START=5. /
&DEVC XYZ=0.0,0.0,1.0, QUANTITY='PDPA', PROP_ID='pdpa_d32' /
```

The following example is used to write out a histogram of droplet size using 20 equally sized bins between 0 and 2000 μm .

```
&PROP ID='pdpa_d'
      PART_ID='water drops'
      QUANTITY='DIAMETER'
      PDPA_RADIUS=0.01
```

```

PDPA_START=0.0
PDPA_M=1
HISTOGRAM=T
HISTOGRAM_NBINS=20
HISTOGRAM_LIMITS=0,2000E-6 /
&DEVC XYZ=0.0,0.0,1.0, QUANTITY='PDPA', PROP_ID='pdpa_d' /

```

The following example provides a normalized distribution of droplet volumes:

```

&PROP ID='Droplet volumetric distribution'
PART_ID='water-drops'
QUANTITY='DIAMETER'
PDPA_RADIUS=0.2
PDPA_START=2.0
PDPA_M=1
PDPA_N=3
HISTOGRAM=T
HISTOGRAM_NBINS=30
HISTOGRAM_CUMULATIVE=.FALSE.
HISTOGRAM_NORMALIZE=.TRUE.
HISTOGRAM_LIMITS=0,500e-6 /

```

22.10.17 Output Associated with Thermogravimetric Analysis (TGA)

In addition to the profile (PROF) output, there are various additional quantities that are useful for monitoring reacting surfaces. For example, 'WALL THICKNESS' gives the overall thickness of the solid surface element. 'SURFACE DENSITY' gives the overall mass per unit area for the solid surface element, computed as an integral of material density over wall thickness. Both quantities are available both as DEVC and BNDF. For the quantity 'SURFACE DENSITY', you can add a MATL_ID to get the surface density of a single component of the solid material.

Thermogravimetric Analysis (TGA) Output

Thermogravimetric Analysis or TGA is a bench-scale measurement in which a very small solid material sample is heated up at a constant rate. The results of a TGA measurement are presented in the form of a normalized mass and normalized mass loss rate. Analogous quantities can be output from FDS:

$$\begin{aligned}
 \text{'NORMALIZED MASS'} &= \frac{\sum_{\alpha} m''_{\alpha}(t)}{\sum_{\alpha} m''_{\alpha}(0)} \quad (\text{dimensionless}) \\
 \text{'NORMALIZED MASS LOSS RATE'} &= \frac{\sum_{\alpha} \dot{m}''_{\alpha}(t)}{\sum_{\alpha} m''_{\alpha}(0)} \quad (1/\text{s})
 \end{aligned}$$

For both of these quantities, you can add a MATL_ID to get the normalized mass or mass loss rate of a single component of the solid material, $m''_{\alpha}(t)/\sum_{\alpha} m''_{\alpha}(0)$.

Micro-Combustion Calorimetry (MCC) Output

Micro-Combustion Calorimetry or MCC is similar to TGA, except the vaporized gas is burned. The result is a normalized heat release rate:

$$\text{'NORMALIZED HEAT RELEASE RATE'} = \dot{m}''_f(t) \Delta H / \sum_{\alpha} m''_{\alpha}(0) \quad (\text{W/g})$$

Note that \dot{m}''_f is the mass flux of fuel and ΔH is the heat of combustion.

Differential Scanning Calorimetry (DSC) Output

Differential Scanning Calorimetry or DSC is a measurement of the rate of heat absorption by a small material sample under constant heating. The result is a normalized heat absorption rate:

$$\text{'NORMALIZED HEATING RATE'} = \dot{q}_c''(t) / \sum_{\alpha} m_{\alpha}''(0) \quad (\text{W/g})$$

Note that it is assumed that the sample is heated purely by convection, in which case \dot{q}_c'' is the convective heat flux.

22.10.18 Fractional Effective Dose (FED) and Fractional Irritant Concentration (FIC)

The Fractional Effective Dose index (FED), developed by Purser [91], is a commonly used measure of human incapacitation due to the combustion gases. The FED value is calculated as

$$\text{FED}_{\text{tot}} = (\text{FED}_{\text{CO}} + \text{FED}_{\text{CN}} + \text{FED}_{\text{NO}_x} + \text{FLD}_{\text{irr}}) \times \text{HV}_{\text{CO}_2} + \text{FED}_{\text{O}_2} \quad (22.42)$$

The fraction of an incapacitating dose of CO is calculated as

$$\text{FED}_{\text{CO}} = \int_0^t 2.764 \times 10^{-5} (C_{\text{CO}}(t))^{1.036} dt \quad (22.43)$$

where t is time in minutes and C_{CO} is the CO concentration (ppm). The fraction of an incapacitating dose of CN is calculated as

$$\text{FED}_{\text{CN}} = \int_0^t \left(\frac{1}{220} \exp\left(\frac{C_{\text{CN}}(t)}{43}\right) - 0.0045 \right) dt \quad (22.44)$$

where t is time in minutes and C_{CN} is the concentration (ppm) of HCN corrected for the protective effect of NO_2 . C_{CN} is calculated as

$$C_{\text{CN}} = C_{\text{HCN}} - C_{\text{NO}_2} - C_{\text{NO}} \quad (22.45)$$

The fraction of an incapacitating dose of NO_x is calculated as

$$\text{FED}_{\text{NO}_x} = \int_0^t \frac{C_{\text{NO}_x}(t)}{1500} dt \quad (22.46)$$

where t is time in minutes and C_{NO_x} is the sum of NO and NO_2 concentrations (ppm).

The Fractional Lethal Dose (FLD) of irritants is calculated as

$$\text{FLD}_{\text{irr}} = \int_0^t \left(\frac{C_{\text{HCl}}(t)}{F_{\text{FLD,HCl}}} + \frac{C_{\text{HBr}}(t)}{F_{\text{FLD,HBr}}} + \frac{C_{\text{HF}}(t)}{F_{\text{FLD,HF}}} + \frac{C_{\text{SO}_2}(t)}{F_{\text{FLD,SO}_2}} + \frac{C_{\text{NO}_2}(t)}{F_{\text{FLD,NO}_2}} + \frac{C_{\text{C}_3\text{H}_4\text{O}}(t)}{F_{\text{FLD,C}_3\text{H}_4\text{O}}} + \frac{C_{\text{CH}_2\text{O}}(t)}{F_{\text{FLD,CH}_2\text{O}}} \right) dt \quad (22.47)$$

where t is time in minutes, the nominators are the instantaneous concentrations (ppm) of each irritant and the denominators are the exposure doses of respective irritants predicted to be lethal to half the population. The lethal exposure doses [91] are given in Table 22.3. To include the effect of an irritant gas not listed in the table, you should specify F_{FLD} in $\text{ppm} \times \text{min}$ using the `FLD_LETHAL_DOSE` property of the corresponding SPEC line.

The fraction of an incapacitating dose of low O_2 hypoxia is calculated as

$$\text{FED}_{\text{O}_2} = \int_0^t \frac{dt}{\exp[8.13 - 0.54(20.9 - C_{\text{O}_2}(t))]} \quad (22.48)$$

Table 22.3: Coefficients used for the computation of irritant effects of gases.

	HCl	HBr	HF	SO ₂	NO ₂	C ₃ H ₄ O	CH ₂ O
F _{FLD} (ppm × min)	114000	114000	87000	12000	1900	4500	22500
F _{FIC} (ppm)	900	900	900	120	350	20	30

where t is time in minutes and C_{O_2} is the O₂ concentration (volume percent). The hyperventilation factor induced by carbon dioxide is calculated as

$$HV_{CO_2} = \frac{\exp(0.1903 C_{CO_2}(t) + 2.0004)}{7.1} \quad (22.49)$$

where t is time in minutes and C_{CO_2} is the CO₂ concentration (percent).

The Fractional Irritant Concentration (FIC), also developed by Purser [91], represents the toxic effect which depends upon the immediate concentrations of irritants. The overall irritant concentration FIC is calculated as

$$FIC_{irr} = \frac{C_{HCl}(t)}{F_{FIC,HCl}} + \frac{C_{HBr}(t)}{F_{FIC,HBr}} + \frac{C_{HF}(t)}{F_{FIC,HF}} + \frac{C_{SO_2}(t)}{F_{FIC,SO_2}} + \frac{C_{NO_2}(t)}{F_{FIC,NO_2}} + \frac{C_{C_3H_4O}(t)}{F_{FIC,C_3H_4O}} + \frac{C_{CH_2O}(t)}{F_{FIC,CH_2O}} \quad (22.50)$$

where the nominators are the instantaneous concentrations of each irritant and the denominators are the concentrations of respective irritants expected to cause incapacitation in half the population. The incapacitating concentrations [91] are given in Table 22.3. To include the irritant effect of a gas not listed in the table, you should specify F_{FIC} in ppm using the FIC_CONCENTRATION property on the corresponding SPEC line.

The output quantities FED and FIC may be displayed either at a point via a device:

```
&DEVC ID='whatever', XYZ=..., QUANTITY='FED', PROP_ID='Activity Level' /
```

You can also visualize FIC as a contour slice. The output quantity FED cannot be directly output as a contour slice, but it can be visualized as a slice in Smokeview if the slices for O₂, CO₂, and CO are defined in the same plane:

```
&SLCF PBY=1.0, QUANTITY='VOLUME FRACTION', SPEC_ID='CARBON DIOXIDE' /
&SLCF PBY=1.0, QUANTITY='VOLUME FRACTION', SPEC_ID='CARBON MONOXIDE' /
&SLCF PBY=1.0, QUANTITY='VOLUME FRACTION', SPEC_ID='OXYGEN' /
```

Notice that all slices are defined on the same plane. If these slices are present, Smokeview can compute the FED and display it as a contour slice.

Also note that device output for FED can make use of an optional activity level defined on a property line:

```
&PROP ID='Activity Level', FED_ACTIVITY=3 /
```

The parameter FED_ACTIVITY is an integer denoting that the person is at rest (1), doing light work (2), or doing heavy work (3). The default value is 2.

22.10.19 Histograms

It is sometimes useful to compile probability distribution functions (PDFs) or histograms of various output quantities. Suppose, for example, you are monitoring the temperature at two locations, and in addition to

plots of the time histories, you want a PDF as well. Do something like following:

```
&DEVC XYZ=..., QUANTITY='TEMPERATURE', ID='T_1', PROP_ID='hist', HIDE_COORDINATES=F /
&DEVC XYZ=..., QUANTITY='TEMPERATURE', ID='T_2', PROP_ID='hist', HIDE_COORDINATES=T /
&PROP ID='hist'
    HISTOGRAM=T
    HISTOGRAM_NBINS=200
    HISTOGRAM_LIMITS=0,2000e-6
    HISTOGRAM_CUMULATIVE=F
    HISTOGRAM_NORMALIZE=T /
```

When `HISTOGRAM` is set to `T` on the `PROP` line, normalized histogram bin counts are output to a comma-separated value file (`CHID_hist.csv`). The parameter `HISTOGRAM_NBINS` is the number of bins dividing the quantity range `HISTOGRAM_LIMITS`. Values falling outside the histogram limits are included in the counts of the first and last bins. Cumulative distributions can be output by setting `HISTOGRAM_CUMULATIVE=T`. To output unnormalized counts or a cumulative distribution, set `HISTOGRAM_NORMALIZE=F`.

The histogram output file contains two-columns for each device. The first column gives the bin center-points, and the second column gives the relative frequency. The parameter `HIDE_COORDINATES` is handy for suppressing the repeated entry of the first column when multiple devices use the same set of histogram parameters.

22.10.20 Complex Terrain and Related Quantities

Complex terrain for wind and wildland fire simulations can be generated either by using an immersed boundary `GEOMETRY`. Visualizing output data on or just above complex terrain can be done in a few different ways.

The first method is via a “slice” file that conforms to the terrain. The following line illustrates how to do it:

```
&SLCF AGL_SLICE=10., QUANTITY='TEMPERATURE' /
```

Instead of specifying a plane on which to draw contours of gas temperature, you specify that the contour is to be located 10 m Above Ground Level (AGL). Adding `VECTOR=T` adds the option of showing velocity vectors.

The second method of visualizing data on complex terrain is via a boundary (`BNDF`) file. Specify a solid phase output quantity as you normally would, for example:

```
&BNDF QUANTITY='WALL TEMPERATURE' /
```

You may also specify an image file (.png) to overlay on the terrain via `TERRAIN_IMAGE` on the `MISC` line.

22.10.21 Wind and the Pressure Coefficient

In the field of wind engineering, a commonly used quantity is known as the `PRESSURE_COEFFICIENT`:

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty U^2} \quad (22.51)$$

p_∞ is the ambient, or “free stream” pressure, and ρ_∞ is the ambient density. The parameter U is the free-stream wind speed, given as `CHARACTERISTIC_VELOCITY` on the `PROP` line

```
&BNDF QUANTITY='PRESSURE COEFFICIENT', PROP_ID='U' /
&DEVC ID='Cp', XYZ=..., IOR=2, QUANTITY='PRESSURE COEFFICIENT', PROP_ID='U' /
```

```
&PROP ID='U', CHARACTERISTIC_VELOCITY=3.4 /
```

Thus, you can either paint values of C_p at all surface points, or create a single time history of C_p using a single device at a single point.

Wall pressure, viscous stresses and integrated forces

If you desire to output the pressure on a point in the wall or viscous stress along the stream-wise direction next to it, you can set devices in the form:

```
&DEVC ID='WP', XYZ=..., IOR=..., QUANTITY='WALL PRESSURE', SURF_ID='MySurf' /
&DEVC ID='WS', XYZ=..., IOR=..., QUANTITY='VISCOUS STRESS WALL ', SURF_ID='MySurf' /
```

The corresponding distributed forces at said point, projected in a specified direction, can be obtained adding the vector triplet `FORCE_DIRECTION`. You can also integrate these distributed forces to obtain total pressure and viscous forces on a surface. As an example, if you want to compute total forces on faces with `SURF_ID='MySurf'`, within a volume `XB` in the x direction, add:

```
&DEVC ID='PFx', XB=..., QUANTITY='WALL PRESSURE', SURF_ID='MySurf',
    SPATIAL_STATISTIC='SURFACE INTEGRAL', FORCE_DIRECTION=1.,0.,0. /
&DEVC ID='VFx', XB=..., QUANTITY='VISCOUS STRESS WALL ',
    SURF_ID='MySurf', SPATIAL_STATISTIC='SURFACE INTEGRAL',
    FORCE_DIRECTION=1.,0.,0. /
```

22.10.22 Dry Volume and Mass Fractions

During actual experiments, species such as CO and CO₂ are typically measured “dry”; that is, the water vapor is removed from the gas sample prior to analysis. For easier comparison of FDS predictions with measured data, you can specify the logical parameter `DRY` on a `DEVC` line that reports the 'MASS FRACTION' or 'VOLUME FRACTION' of a species. For example, the first line reports the actual volume fraction of CO, and the second line reports the output of a gas analyzer in a typical experiment.

```
&DEVC ID='wet CO', XYZ=..., QUANTITY='VOLUME FRACTION', SPEC_ID='CARBON MONOXIDE' /
&DEVC ID='dry CO', XYZ=..., QUANTITY='VOLUME FRACTION', SPEC_ID='CARBON MONOXIDE',
    DRY=T /
```

22.10.23 Aerosol and Soot Concentration

Currently there are three different device options for outputting aerosol concentration (e.g., soot concentration) from FDS. It is important to recognize what each device is outputting so that the proper selection can be made.

```
&DEVC ID='MF_SOOT', XYZ=..., QUANTITY='MASS FRACTION', SPEC_ID='SOOT' /
&DEVC ID='VF_SOOT', XYZ=..., QUANTITY='VOLUME FRACTION', SPEC_ID='SOOT' /
&DEVC ID='SOOT_VF', XYZ=..., QUANTITY='AEROSOL VOLUME FRACTION', SPEC_ID='SOOT' /
```

Specifying a `DEVC` with a 'MASS FRACTION' and a `SPEC_ID` of `SOOT` will output the mass fraction of soot in the gas phase. The quantity 'VOLUME FRACTION' and a `SPEC_ID` of `SOOT` will output the volume fraction of soot in the gas phase treating the soot as if it were an ideal gas. The quantity 'AEROSOL VOLUME FRACTION'

and a `SPEC_ID` of 'SOOT' will output the volume fraction of soot as if it were a solid particle in the computational cell based on the following equation,

$$f_v = \rho Y_a / \rho_a \quad (22.52)$$

where ρ is the local density, Y_a is the local mass fraction of the aerosol, and ρ_a is density of the aerosol defined using the `SPEC` input `DENSITY_SOLID`. The default value for `DENSITY_SOLID` is `SOOT_DENSITY` on `MISC`, which defaults to 1800 kg/m³ for soot [92].

22.10.24 Gas Velocity

The gas velocity components, u , v , and w , can be output as slice (`SLCF`), point device (`DEVC`), isosurface (`ISO`), or Plot3D quantities using the names 'U-VELOCITY', 'V-VELOCITY', and 'W-VELOCITY'. The total velocity is specified as just 'VELOCITY'. Normally, the velocity is always positive, but you can use the parameter `VELO_INDEX` with a value of 1, 2 or 3 to indicate that the velocity ought to have the same sign as u , v , or w , respectively. This is handy if you are comparing velocity predictions with measurements. For Plot3D files, add `PLOT3D_VELO_INDEX(N)=...` to the `DUMP` line, where N refers to the Plot3D quantity 1, 2, 3, 4 or 5.

22.10.25 Enthalpy

There are several outputs that report the enthalpy of the gas mixture. First, the 'SPECIFIC ENTHALPY' and the 'SPECIFIC SENSIBLE ENTHALPY' are defined:

$$h(T) = \Delta h_f^\circ + \int_{T_{\text{ref}}}^T c_p(T') dT' \quad ; \quad h_s(T) = \int_{T_{\text{ref}}}^T c_p(T') dT' \quad (22.53)$$

Both have units of kJ/kg. The quantities 'ENTHALPY' and 'SENSIBLE ENTHALPY' are ρh and ρh_s , respectively, in units of kJ/m³.

22.10.26 Computer Performance

There are several useful `DEVC QUANTITY`'s that can help monitor the performance of your computer:

- 'ACTUATED SPRINKLERS' Number of activated sprinklers.
- 'CFL MAX' The maximum value of the CFL (Courant-Friedrichs-Lewy) number, the primary constraint on the time step, for the mesh in which the device is located. By default, the time step is chosen so that the CFL number remains within the range of 0.8 to 1.0. If you want to see the CFL number in each grid cell, use a slice (`SLCF`) file with `QUANTITY='CFL'` and `CELL_CENTERED=T`.
- 'CPU TIME' Elapsed CPU time since the start of the simulation, in seconds.
- 'ITERATION' Number of time steps completed at the given time of the simulation.
- 'NUMBER OF PARTICLES' Number of Lagrangian particles for the `MESH` in which the `DEVC` is located.
- 'TIME STEP' Duration of a simulation time step, δt , in seconds.
- 'VN MAX' The maximum value of the VN (Von Neumann) number, a secondary constraint on the time step, for the mesh in which the device is located. By default, the time step is chosen so that the VN number remains below 1. If you want to see the VN number in each grid cell, use a slice (`SLCF`) file with `QUANTITY='VN'` and `CELL_CENTERED=T`.

- 'WALL CLOCK TIME' Elapsed wall clock time since the start of the simulation, in seconds.
- 'WALL CLOCK TIME ITERATIONS' Elapsed wall clock time since the start of the time stepping loop, in seconds.
- 'RAM' (Linux only) The memory used by the process that controls the mesh in which this device is located, in units of MB. This value is the equivalent of the value reported under the heading `RES` when doing a `top` command at the command prompt. Usually `RES` is reported in kB, but it is converted to MB here. More precisely, 'RAM' is 1/1000 of the value of `VmRSS` in the system file called `/proc/<PID>/status` where `<PID>` is the process ID.

In addition, the following flags can be useful in monitoring the performance of an MPI calculation. They are typically used for debugging.

- `VELOCITY_ERROR_FILE` If set to `T` on the `DUMP` line, this parameter will cause FDS to create a file with a time history of the maximum error associated with the normal component of velocity at solid or interpolated boundaries. See Sec. 21.3 for a description of this file.

22.10.27 Output File Precision

There are several different output files that have the format of a comma-separated value (.csv) file. These files consist of real numbers in columns separated by commas. By default, the real numbers are formatted

```
-1.2345678E+123
```

To change the precision of the numbers, use `SIG_FIGS` on the `DUMP` line to indicate the number of significant figures in the mantissa (default is 8). Use `SIG_FIGS_EXP` to change the number of digits in the exponent (default is 3). Keep in mind that the precision of real numbers used internally in an FDS calculation is approximately 12, equivalent to 8 byte or double precision following conventional Fortran rules.

22.10.28 A Posteriori Mesh Quality Metrics

The quality of a particular simulation is most directly tied to grid resolution. Three output quantities are discussed here for measuring errors in the velocity and scalar fields. It should be noted that the link between these metrics and true simulation quality is still in the research phase. In other words, a good quality score is not sufficient to assure a good simulation (at the present time).

Measure of Turbulence Resolution

A scalar quantity referred to as the *measure of turbulence resolution* [93] is defined locally as:

$$M(\mathbf{x}) = \frac{\langle k_{sgs} \rangle}{\langle TKE \rangle + \langle k_{sgs} \rangle} \quad (22.54)$$

Angled brackets denote suitable time-averages.

The turbulent kinetic energy (TKE) must be post processed because we cannot compute the fluctuation until we know the mean. Use the following `DEVCS` to output the three velocity components:

```
&DEVC ..., QUANTITY='U-VELOCITY' /
&DEVC ..., QUANTITY='V-VELOCITY' /
&DEVC ..., QUANTITY='W-VELOCITY' /
```

TKE is then computed by

$$\text{TKE} = \frac{1}{2} ((\tilde{u} - \langle \tilde{u} \rangle)^2 + (\tilde{v} - \langle \tilde{v} \rangle)^2 + (\tilde{w} - \langle \tilde{w} \rangle)^2) \quad (22.55)$$

The subgrid kinetic energy is estimated from Deardorff's eddy viscosity model (see [3])

$$k_{\text{sgs}} \approx (\mu_t / (\rho C_v \Delta))^2 \quad (22.56)$$

To output an estimate of the subgrid kinetic energy per unit mass use

```
&DEVC ..., QUANTITY='SUBGRID KINETIC ENERGY' /
```

You should then average TKE and k_{sgs} for use in (22.54).

The concept behind the measure of turbulence resolution is illustrated in Figure 22.9. Notice that on the left the difference between the grid signal and the test signal is very small. On the right, the grid signal is highly turbulent and the corresponding test signal is much smoother. We infer then that the flow is under-resolved.

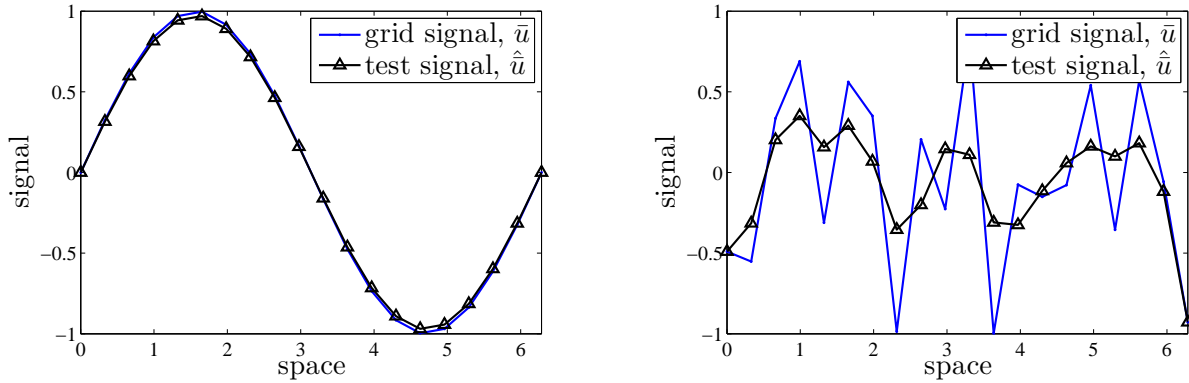


Figure 22.9: (Left) Resolved signal, M is small. (Right) Unresolved signal, M is close to unity.

For the canonical case of isotropic turbulence Pope actually defines LES such that $M < 0.2$. That is, LES requires resolution of 80% of the kinetic energy in the flow field (because this puts the grid Nyquist limit within the inertial subrange). The question remains as to whether this critical value is sufficient or necessary for a given engineering problem. As shown in Ref. [94], maintaining mean values of M near 0.2 indeed provides satisfactory results (simulation results within experimental error bounds) for mean velocities and species concentrations in non-reacting, buoyant plumes.

Wavelet Error Measure

A resolution metric that we call the *wavelet error measure* or WEM may be output using, for example,

```
&SLCF PBX=0, QUANTITY='WAVELET ERROR', QUANTITY2='HRRPUV' /
```

We begin by providing background on the simple Haar wavelet [95]. For a thorough and more sophisticated review of wavelet methods, the reader is referred to Schneider and Vasilyev [96].

Suppose the scalar function $f(r)$ is sampled at discrete points r_j , separated by a distance h , giving values s_j . Defining the *unit step function* over the interval $[r_1, r_2]$ by

$$\phi_{[r_1, r_2]} = \begin{cases} 1 & \text{if } r_1 \leq r < r_2 \\ 0 & \text{otherwise} \end{cases} \quad (22.57)$$

the simplest possible reconstruction of the signal is the step function approximation

$$f(r) \approx \sum_j s_j \phi_{[r_j, r_j+h]}(r) \quad (22.58)$$

By “viewing” the signal at a coarser resolution, say $2h$, an identical reconstruction of the function f over the interval $[r_j, r_j + 2h]$ may be obtained from

$$f_{[r_j, r_j+2h]}(r) = \underbrace{\frac{s_j + s_{j+1}}{2}}_a \phi_{[r_j, r_j+2h]}(r) + \underbrace{\frac{s_j - s_{j+1}}{2}}_c \psi_{[r_j, r_j+2h]}(r) \quad (22.59)$$

where a is as the *average* coefficient and c is as the *wavelet* coefficient. The Haar *mother wavelet* (Figure 22.10) is identified as

$$\psi_{[r_1, r_2]}(r) = \begin{cases} 1 & \text{if } r_1 \leq r < \frac{1}{2}(r_1 + r_2) \\ -1 & \text{if } \frac{1}{2}(r_1 + r_2) \leq r < r_2 \end{cases} \quad (22.60)$$

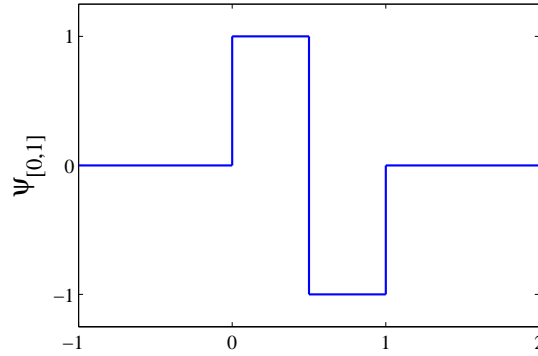


Figure 22.10: Haar mother wavelet on the interval $[0,1]$.

The decomposition of the signal shown in Eq. (22.59) may be repeated at ever coarser resolutions. The result is a *wavelet transform*. The procedure is entirely analogous to the Fourier transform, but with compact support. If we look at a 1D signal with 2^m points, the repeated application of (22.59) results in an $m \times m$ matrix of averages \mathbf{a} with components a_{ij} and an $m \times m$ wavelet coefficient matrix \mathbf{c} with components c_{ij} . Each row i of \mathbf{a} may be reconstructed from the $i+1$ row of \mathbf{a} and \mathbf{c} . Because of this and because small values of the wavelet coefficient matrix may be discarded, dramatic compression of the signal may be obtained.

Here we are interested in using the wavelet analysis to say something about the local level of error due to grid resolution. Very simply, we ask what can be discerned from a sample of four data points along a line. Roughly speaking we might expect to see one of the four scenarios depicted in Figure 22.11. Within each plot window we also show the results of a Haar wavelet transform for that signal. Looking first at the two

top plots, on the left we have a step function and on the right we have a straight line. Intuitively, we expect large error for the step function and small error for the line. The following error measure achieves this goal:

$$\text{WEM}(\mathbf{x}, t) = \max_{x,y,z} \left(\frac{|c_{11} + c_{12}| - |c_{21}|}{|a_{21}|} \right) \quad (22.61)$$

Note that we have arbitrarily scaled the measure so that a step function leads to WEM of unity. In practice the transform is performed in all coordinate directions and the max value is reported. The scalar value may be output to Smokeview at the desired time interval.

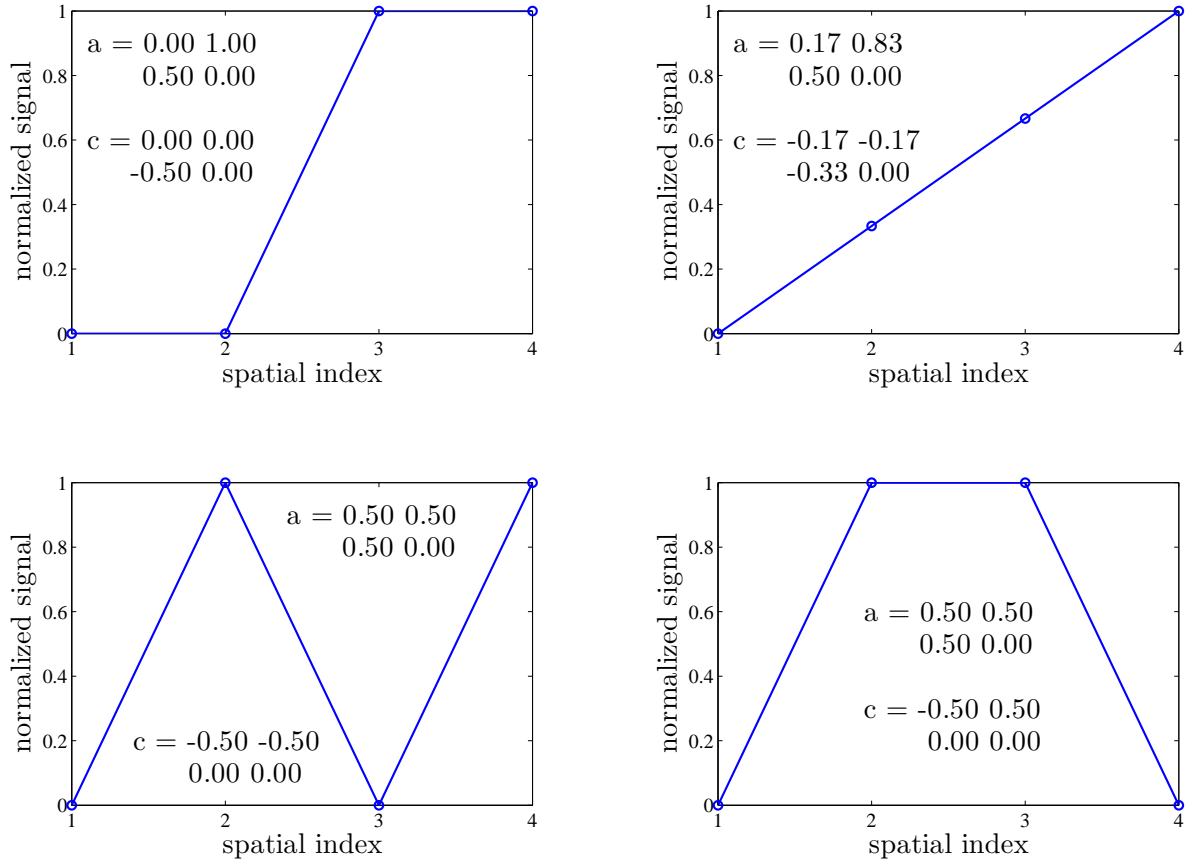


Figure 22.11: Averages and coefficients for local Haar wavelet transforms on four typical signals.

Looking now at the two plots on the bottom of Figure 22.11, the signal on the left, which may indicate spurious oscillations or unresolved turbulent motion, leads to $\text{WEM} = 2$. Our measure therefore views this situation as the worst case in a sense. The signal to the lower right is indicative of an extremum, which actually is easily resolved by most centered spatial schemes and results again in $\text{WEM} = 0$.

In [94], the time average of WEM was reported for LES of a non-reacting buoyant plume at three grid resolutions. From this study, the best advice currently is to maintain average values of WEM less than 0.5.

Local Cell Reynolds Number

Additionally, we provide an estimate of the *local cell Reynolds number* given by the ratio of the cell size (LES filter width, Δ) to an estimate of the local Kolmogorov scale, η (see [69]). For a DNS, Δ/η should be

less than or equal to one. The Kolmogorov scale is computed from its definition:

$$\eta \equiv \left(\frac{(\mu/\rho)^3}{\varepsilon} \right)^{1/4} \quad (22.62)$$

where μ is the molecular dynamic viscosity, ρ is the density, and ε is the kinetic energy dissipation rate, which requires modeling. In FDS, we assume the dissipation rate is locally equivalent to the production of subgrid-scale kinetic energy. This implies

$$\varepsilon = (\mu_t/\rho)|\tilde{S}|^2 \quad (22.63)$$

where μ_t is the turbulent viscosity and $|\tilde{S}|$ is the filtered strain invariant (see FDS Tech Guide).

```
&SLCF PBX=0, QUANTITY='CELL REYNOLDS NUMBER' /
```

Near-Wall Grid Resolution

Large-eddy simulations of boundary layer flows fall into two general categories: LES with near-wall resolution and LES with near-wall modeling (wall functions). FDS employs the latter. The wall models used in FDS are law of the wall [69]. For the wall models to function properly, the grid resolution near the wall should fall within a certain range of y^+ , the non-dimensional distance from the wall expressed in viscous units. To check this, you may add a boundary file output as follows:

```
&BNDF QUANTITY='VISCIOUS WALL UNITS' /
```

The value of y^+ reported is half (since the velocity lives at the cell face center) the wall-normal cell dimension (δn) divided by the maximum between the local viscous length scale, δ_v [69] or the sand grain roughness, s , for rough walls:

$$y^+ = \frac{\delta n/2}{\max(\delta_v, s)}; \quad \delta_v = \frac{\mu/\rho}{u_\tau}; \quad u_\tau = \sqrt{\tau_w/\rho}, \quad (22.64)$$

where $\tau_w = \mu \partial|\mathbf{u}|/\partial n$ is the viscous stress evaluated at the wall (τ_w is computed by the wall function, $|\mathbf{u}|$ is taken as an estimate of the stream-wise velocity component near the wall); the quantity u_τ is the *friction velocity*. The friction velocity may also be output in a boundary file or via a device attached to a wall. For example:

```
&DEVC XYZ=1,0,0, QUANTITY='FRICTION VELOCITY', IOR=3, ID='u_tau' /
```

Wall functions for LES are still under development, but as a general guideline for efficiency reasons it is recommended that the first grid cell fall within the log layer. There is no penalty—other than computational time—for being more highly resolved. The viscous sublayer lies within $0 < y^+ < 5$. The transition region lies roughly between $5 < y^+ < 30$, a value $y^+ = 30$ would be considered highly resolved. The upper limit of the log region for statistically stationary boundary layers depends on the Reynolds number, and there are no hard rules for transient flows. Beyond $y^+ = 1000$ the first grid cell is likely to fall in the wake region of the boundary layer and may produce unreliable results. A reasonable target for practical engineering LES is $y^+ = \mathcal{O}(100)$.

22.10.29 Extinction

In combustion, knowing if, when, or where chemical reactions have been extinguished is important. The output quantity `EXTINCTION` tells the user whether or not combustion has been prevented by the extinction routine. By default, `EXTINCTION = 0`, which means that the FDS extinction routine has not prevented combustion. An `EXTINCTION` value of 1 means that the routine has prevented combustion. The criteria for an `EXTINCTION` value of 1 is the presence of fuel and oxidizer without any energy release. An `EXTINCTION` value of -1 means that there is either no fuel or no oxidizer present.

22.10.30 Fire spread over a surface

In some cases it can be useful to output timings related to the spread of fire over a surface. For example, when modeling wildland fires the shape and spread of the fire front can be very important. For this reason, two specialized output quantities are available as boundary files (Sec. 22.5) or as measurements from devices places on a solid boundary (Sec. 22.2). These are `FIRE ARRIVAL TIME` and `FIRE RESIDENCE TIME`. The `FIRE ARRIVAL TIME` quantity outputs the time at which the gas-phase cell adjacent to the solid exceeds a threshold for heat release rate per volume (`HRRPUV`) and the `FIRE RESIDENCE TIME` gives the cumulative time over which the threshold is exceeded during the simulation. The chosen threshold is the same as used by smokeview for rendering `HRRPUV`, as described in Sec. 22.8. In the case of a level set simulation (Sec. 17.5), the `FIRE ARRIVAL TIME` can be computed directly from the level set value and the `FIRE RESIDENCE TIME` comes from the spread-rate adjusted burning duration of the fuel, as described in Sec. 17.5.2. These two quantities are cumulatively populated over time such that a full picture of the fire spread can be obtained from relatively infrequent outputs - theoretically only one snapshot at the end of the simulation is required.

For level set simulations another output is available, called `LS SPREAD RATE`. This output stores the magnitude of the local spread rate calculated as the fire reaches a given point along the surface. The dependence on slope and local wind is related to the mode of level set which is activated (Sec. 17.5).

22.11 Extracting Numbers from the Output Data Files

As part of the standard FDS-SMV distribution package, there is a short Fortran program called `fds2ascii`. The source code is located in the GitHub repository, `firemodels/fds`, in the directory, `Utilities/fds2ascii`. To run the program, just type:

```
fds2ascii
```

at the command prompt. You will be asked a series of questions about which type of output file to process, what time interval to time average the data, and so forth. A single file is produced with the name `CHID_fds2ascii.csv`. A typical command line session looks like this:

```
>> fds2ascii
  Enter Job ID string (CHID):
bucket_test_1
  What type of file to parse?
  PL3D file? Enter 1
  SLCF file? Enter 2
  BNDF file? Enter 3
3
  Enter Sampling Factor for Data?
  (1 for all data, 2 for every other point, etc.)
1
  Limit the domain size? (y or n)
y
  Enter min/max x, y and z
-5 5 -5 5 0 1
  1 MESH 1, WALL TEMPERATURE
  Enter starting and ending time for averaging (s)
35 36
  Enter orientation: (plus or minus 1, 2 or 3)
3
  Enter number of variables
1
  Enter boundary file index for variable 1
1
  Enter output file name:
bucket_test_1_fds2ascii.csv
  Writing to file...      bucket_test_1_fds2ascii.csv
```

These commands tell `fds2ascii` that you want to convert (binary) boundary file data into a text file. You want to sample every data point within the specified volume, you want only those surfaces that point upward (+3 orientation), you only want 1 variable (only one is listed anyway and its index is 1 – that is just the number used to list the available files). The data will be time-averaged, and it will be output to a file listed at the end of the session. Here is a more detailed explanation of the questions:

Enter Job ID string (CHID): Enter the name of the job that you entered into the FDS input file under the parameter `CHID`. Do not include the `.fds` suffix.

What type of file to parse? The program can only read Plot3D (`PL3D`), slice (`SLCF`), or boundary (`BNDF`) files.

Enter Sampling Factor for Data: If you want to print out every point, enter 1; every other point, enter 2; and so on.

Domain selection: The answer is a one or two letter code. If the first letter is y or Y , this means you want to print out a subset of the data for the entire domain. If n or N , you do not want to limit the data. The answer of z or Z is only for cases where you have an outdoor fire scenario over rough terrain, and you want to offset the z coordinate so that it denotes height off the ground. If the second letter of the two letter code is a or A , this means that you want the program to automatically select the appropriate files. If the first letter of the two letter code is y or Y , you will be prompted for 6 numbers denoting the minimum and maximum values of x , y , and z .

Enter starting and ending time for averaging (s): The slice and boundary file data will be time-averaged over the designated time interval.

How many variables to read: Enter the number of different output quantities that you want to include in your output file.

Enter orientation: (plus or minus 1, 2 or 3): For a boundary file, you must designate the orientation of the surfaces you want to print out. +1 means surfaces facing the positive x direction, -2 means negative y , and so on.

22.12 Gas Phase Output Quantities

Table 22.4, spread over the following pages, lists gas phase output quantities. The column “File Type” lists the allowed output files for the quantities: “D” is for Device (DEVC), “I” is for Iso-surface (ISOF), “P” is for Plot3D, “S” is for Slice (SLCF).

For those output quantities that require a species name via SPEC_ID, the species implicitly defined when using the simple chemistry combustion model are 'OXYGEN', 'NITROGEN', 'WATER VAPOR', and 'CARBON DIOXIDE'. If CO_YIELD, SOOT_YIELD, and/or HCN_YIELD are specified on the REAC line, then 'CARBON MONOXIDE', 'SOOT', and/or 'HYDROGEN CYANIDE' are included as output species. The fuel species can be output via the FUEL specified on the REAC line. As an example of how to use the species names, suppose you want to calculate the integrated mass flux of carbon monoxide through a horizontal plane, like the total amount entrained in a fire plume. Use a “device” as follows

```
&DEVC ID='CO_flow', XB=-5,5,-5,5,2,2, QUANTITY='MASS FLUX Z',  
      SPEC_ID='CARBON MONOXIDE', SPATIAL_STATISTIC='AREA INTEGRAL' /
```

The ID is just a label in the output file. When an output quantity is related to a particular gas species or particle type, you must specify the appropriate SPEC_ID or PART_ID on the same input line. All output quantity names ought to be in single quotes.

Table 22.4: Gas phase output quantities.

QUANTITY	Explanation	Units	File Type
ABSOLUTE PRESSURE	Absolute pressure, $p = \bar{p} + \tilde{p}$	Pa	D,I,P,S
ABSORPTION COEFFICIENT	Section 14.3	1/m	D,I,P,S
ADVECTIVE MASS FLUX X ¹	Section 22.10.10	kg/s/m ²	D,I,P,S
ADVECTIVE MASS FLUX Y ¹	Section 22.10.10	kg/s/m ²	D,I,P,S
ADVECTIVE MASS FLUX Z ¹	Section 22.10.10	kg/s/m ²	D,I,P,S
AEROSOL VOLUME FRACTION ¹	Section 22.10.23	mol/mol	D,I,P,S
BACKGROUND PRESSURE	Background pressure, \bar{p}	Pa	D,I,P,S
BULK DENSITY	Section 9.2.9	kg/m ³	D,I,P,S
CELL REYNOLDS NUMBER	Section 22.10.28		D,I,P,S
CELL U	$(u_{i,j,k} + u_{i-1,j,k})/2$	m/s	D,I,P,S
CELL V	$(v_{i,j,k} + v_{i,j-1,k})/2$	m/s	D,I,P,S
CELL W	$(w_{i,j,k} + w_{i,j,k-1})/2$	m/s	D,I,P,S
CFL	Section 22.10.26		D,I,P,S
CFL MAX	Section 22.10.26		D
CHEMISTRY SUBITERATIONS	Section 13.3.7		D,S
CHI_R	Section 14.1		D,I,S
COMBUSTION EFFICIENCY	$\delta t / \tau_{\text{mix}}$		D,I,P,S
CONDUCTIVITY	Section 12.1.3	W/(m K)	D,I,P,S
C_SMAG	Smagorinsky coefficient		D,I,P,S
DENSITY ¹	Total or species density	kg/m ³	D,I,P,S
DIFFUSIVE MASS FLUX X ¹	Section 22.10.10	kg/s/m ²	D,I,P,S
DIFFUSIVE MASS FLUX Y ¹	Section 22.10.10	kg/s/m ²	D,I,P,S
DIFFUSIVE MASS FLUX Z ¹	Section 22.10.10	kg/s/m ²	D,I,P,S
DIFFUSIVITY ¹	Section 12.1.3	m ² /s	D,I,P,S
DISSIPATION RATE	$(\mu/\rho) S_{ij} S_{ij}$	m ² /s ³	D,I,P,S
DIVERGENCE	$\nabla \cdot \mathbf{u}$	1/s	D,I,P,S
EFFECTIVE FLAME TEMPERATURE	Section 22.10.15	°C	D,I,P,S
ENTHALPY	Section 12.1.3	kJ/m ³	D,I,P,S
ENTHALPY FLUX X	Section 22.10.11	kW/m ²	D,I,P,S
ENTHALPY FLUX Y	Section 22.10.11	kW/m ²	D,I,P,S
ENTHALPY FLUX Z	Section 22.10.11	kW/m ²	D,I,P,S
EXTINCTION	Section 22.10.29		D,S
EXTINCTION COEFFICIENT	Section 22.10.5	1/m	D,I,P,S
F_X, F_Y, F_Z	Momentum terms, F_x, F_y, F_z	m/s ²	D,I,P,S
H	$H = \mathbf{u} ^2/2 + \tilde{p}/\rho$	(m/s) ²	D,I,P,S
HRRPUV	\dot{q}'''	kW/m ³	D,I,P,S
HRRPUV REAC ⁶	\dot{q}''' for REAC_ID	kW/m ³	D,S
IDEAL GAS PRESSURE	$\bar{p} = \rho RT / \bar{W}$	Pa	D,I,P,S
INTEGRATED INTENSITY	$U = \int I \, ds$	kW/m ²	D,I,P,S
INTERNAL ENERGY	$\rho h - \bar{p}$	kJ/m ³	D,I,P,S
KINETIC ENERGY	Staggered $(u^2 + v^2 + w^2)/2$	(m/s) ²	D,I,P,S
KOLMOGOROV LENGTH SCALE	Section 22.10.28	m	D,I,P,S
MACH NUMBER	$ \mathbf{u} / \sqrt{(R/\bar{W})T\gamma}$		S,D

Table 22.4: Gas phase output quantities (continued).

QUANTITY	Explanation	Units	File Type
MASS FLUX x^1	Section 22.10.10	$\text{kg}/(\text{m}^2 \text{ s})$	D,I,P,S
MASS FLUX y^1	Section 22.10.10	$\text{kg}/(\text{m}^2 \text{ s})$	D,I,P,S
MASS FLUX z^1	Section 22.10.10	$\text{kg}/(\text{m}^2 \text{ s})$	D,I,P,S
MASS FRACTION ¹	Y_α	kg/kg	D,I,P,S
MAXIMUM VELOCITY ERROR	Section 21	m/s	D
MIXING TIME	Combustion mixing time, τ_{mix}	s	D,I,P,S
MIXTURE FRACTION	Z	kg/kg	D,I,P,S
MOLECULAR CONDUCTIVITY	Section 12.1.3	$\text{W}/(\text{m K})$	D,I,P,S
MOLECULAR VISCOSITY	Molecular viscosity, $\mu(\mathbf{Z}, T)$	$\text{kg}/(\text{m s})$	D,I,P,S
OPTICAL DENSITY	Section 22.10.5	$1/\text{m}$	D,I,P,S
ORIENTED VELOCITY ⁵	$(u, v, w) \cdot (n_x, n_y, n_z)$	m/s	D
PRESSURE	Perturbation pressure, \tilde{p}	Pa	D,I,P,S
PRESSURE ITERATIONS	Number of pressure iterations		D
PRESSURE ZONE	Section 10.3		D,S
Q CRITERION	$\frac{1}{2}[\text{trace}(\nabla \mathbf{u})^2 - \text{trace}((\nabla \mathbf{u})^2)]$	$1/\text{s}^2$	D,I,P,S
RADIAL VELOCITY	$(u, v) \cdot (x, y)/\sqrt{x^2 + y^2}$	m/s	D,I,P,S
RADIATION ABSORPTION	κU	kW/m^3	D,I,P,S
RADIATION EMISSION	$4\kappa\sigma T^4$	kW/m^3	D,I,P,S
RADIATION LOSS	$\nabla \cdot \mathbf{q}_r'' = \kappa(U - 4\sigma T^4)$	kW/m^3	D,I,P,S
RADIATIVE HEAT FLUX GAS	Section 22.10.12	kW/m^2	D
REAC SOURCE TERM ¹	\dot{m}'''_α	kg/m^3	D,I,P,S
RELATIVE HUMIDITY	Section 13.1.1	%	D,I,P,S
RESOLVED KINETIC ENERGY	$k_{\text{res}} = (\bar{u}^2 + \bar{v}^2 + \bar{w}^2)/2$	$(\text{m}/\text{s})^2$	D,I,P,S
RTE SOURCE CORRECTION FACTOR	Section 14.1.3		D
SENSIBLE ENTHALPY	Section 22.10.25	kJ/m^3	D,I,P,S
SPECIFIC ENTHALPY	Section 22.10.25	kJ/kg	D,I,P,S
SPECIFIC HEAT	c_p	$\text{kJ}/(\text{kg K})$	D,I,P,S
SPECIFIC INTERNAL ENERGY	$h - \bar{p}/\rho$	kJ/kg	D,I,P,S
SPECIFIC SENSIBLE ENTHALPY	Section 22.10.25	kJ/kg	D,I,P,S
STRAIN RATE	$2(S_{ij}S_{ij} - 1/3(\nabla \cdot \mathbf{u})^2)$	$1/\text{s}$	D,I,P,S
STRAIN RATE X	$\partial w/\partial y + \partial v/\partial z$	$1/\text{s}$	D,I,P,S
STRAIN RATE Y	$\partial u/\partial z + \partial w/\partial x$	$1/\text{s}$	D,I,P,S
STRAIN RATE Z	$\partial v/\partial x + \partial u/\partial y$	$1/\text{s}$	D,I,P,S
SUBGRID KINETIC ENERGY	Section 22.10.28	m^2/s^2	D,S
SUM LUMPED MASS FRACTIONS	$\sum_i Z_i$ (should be 1)		D,S
SUM PRIMITIVE MASS FRACTIONS	$\sum_\alpha Y_\alpha$ (should be 1)		D,S
TEMPERATURE	Section 22.10.15	$^\circ\text{C}$	D,I,P,S
TOTAL MASS FLUX x^1	Section 22.10.10	$\text{kg}/\text{s}/\text{m}^2$	D,I,P,S
TOTAL MASS FLUX y^1	Section 22.10.10	$\text{kg}/\text{s}/\text{m}^2$	D,I,P,S
TOTAL MASS FLUX z^1	Section 22.10.10	$\text{kg}/\text{s}/\text{m}^2$	D,I,P,S
U-VELOCITY	Gas velocity component, u	m/s	D,I,P,S
V-VELOCITY	Gas velocity component, v	m/s	D,I,P,S
W-VELOCITY	Gas velocity component, w	m/s	D,I,P,S

Table 22.4: Gas phase output quantities (continued).

QUANTITY	Explanation	Units	File Type
VELOCITY ³	Gas velocity	m/s	D,I,P,S
VISCOSITY	Effective viscosity, $\mu + \mu_t$	kg/(m s)	D,I,P,S
VISIBILITY	Section 22.10.5	m	D,I,P,S
VN	Section 22.10.26		D,I,P,S
VN MAX	Section 22.10.26		D
VORTICITY X	$\partial w / \partial y - \partial v / \partial z$	1/s	D,I,P,S
VORTICITY Y	$\partial u / \partial z - \partial w / \partial x$	1/s	D,I,P,S
VORTICITY Z	$\partial v / \partial x - \partial u / \partial y$	1/s	D,I,P,S
VOLUME FRACTION ¹	X_α	mol/mol	D,I,P,S
WAVELET ERROR	Section 22.10.28		S

¹ Requires the specification of a gas species using SPEC_ID

Omit SPEC_ID for total flux.

Do not use for MIXTURE FRACTION.

² Requires the specification of a particle name using PART_ID

³ Add VELO_INDEX=1 to the input line if you want to multiply the velocity by the sign of u .

Use the indices 2 and 3 for v and w , respectively.

⁴ Allows for an optional MATL_ID

⁵ Requires an ORIENTATION on the DEVC line.

⁶ Requires REAC_ID

22.13 Solid Phase Output Quantities

Table 22.5 below lists solid phase output quantities. Appropriate sections are cited. The definition of mathematical symbols can be found in Volume 1 of the FDS Technical Reference Guide [3]. The File Type “B” refers to a boundary file, “D” refers to the device file, CHID_devic.csv, and “pr” refers to the profile file, CHID_prof.csv.

Table 22.5: Solid phase output quantities.

QUANTITY	Explanation	Units	File Type
ADIABATIC SURFACE TEMPERATURE	Section 22.10.13	°C	B,D
AMPUA ²	Section 22.9.1	kg/m ²	B,D
AMPUA_ _z	Section 22.9.1	kg/m ²	B,D
BACK WALL TEMPERATURE	Section 22.2.2	°C	B,D
BLOWING CORRECTION	Section 8.2.2		B,D
BURNING RATE	Mass loss rate of fuel	kg/(m ² s)	B,D
CONDENSATION HEAT FLUX	Section 13.7	kW/m ²	B,D
CONVECTIVE HEAT FLUX	Section 22.10.12	kW/m ²	B,D
CONVECTIVE HEAT FLUX GAUGE	Section 22.10.12	kW/m ²	B,D
CONVECTIVE HEAT TRANSFER REGIME	Section 8.2.2		B,D
CPUA ²	Section 22.9.1	kW/m ²	B,D
CPUA_ _z	Section 22.9.1	kW/m ²	B,D
DEPOSITION VELOCITY	Section 13.4	m/s	B,D
FRICTION VELOCITY	Section 22.10.28	m/s	B,D
GAUGE HEAT FLUX	Section 22.10.12	kW/m ²	B,D
ENTHALPY FLUX WALL	Section 22.10.11	kW/m ²	B,D
TOTAL HEAT FLUX	Section 22.10.12	kW/m ²	B,D
EMISSIVITY	Surface emissivity (usually constant)		B,D
FIRE ARRIVAL TIME	Section 22.10.30	s	B,D
FIRE RESIDENCE TIME	Section 22.10.30	s	B,D
LS SPREAD RATE	Section 22.10.30	m/s	B,D
GAS DENSITY	Gas Density near wall	kg/m ³	B,D
GAS TEMPERATURE	Gas Temperature near wall	°C	B,D
HEAT TRANSFER COEFFICIENT	Section 8.2.2	W/(m ² K)	B,D
HRRPUA	\dot{q}''	kW/m ²	B,D
INCIDENT HEAT FLUX	Section 22.10.12	kW/m ²	B,D
INSIDE WALL TEMPERATURE	Section 22.2.2	°C	D,Pr
INSIDE WALL DEPTH	Section 22.2.2	m	D,Pr
LAYER DIVIDE DEPTH	Section 9.2.2	m	B,D
MASS FLUX ^{1,4}	Section 22.10.10	kg/(m ² s)	B,D
MASS FLUX WALL ¹	Section 22.10.10	kg/(m ² s)	B,D
MPUA ²	Section 22.9.1	kg/m ²	B,D
MPUA_ _z	Section 22.9.1	kg/m ²	B,D
NORMAL VELOCITY	Wall normal velocity	m/s	B,D
NORMALIZED HEATING RATE	Section 22.10.17	W/g	D
NORMALIZED HEAT RELEASE RATE	Section 22.10.17	W/g	D
NORMALIZED MASS ⁴	Section 22.10.17		D

Table 22.5: Solid phase output quantities (continued).

QUANTITY	Explanation	Units	File Type
NORMALIZED MASS LOSS RATE ⁴	Section 22.10.17	1/s	D
OXIDATIVE HRRPUA	Section 9.2.3	kW/m ²	B,D
PRESSURE COEFFICIENT	Section 22.10.21		B,D
RADIATIVE HEAT FLUX	Section 22.10.12	kW/m ²	B,D
RADIOMETER	Section 22.10.12	kW/m ²	B,D
REFERENCE_HEAT_FLUX	Section 9.1.4	kW/m ²	B,D
SOLID CONDUCTIVITY ⁴	Section 22.2.2	W/(m K)	D,Pr
SOLID DENSITY ⁴	Section 22.2.2	kg/m ³	D,Pr
SOLID ENTHALPY ⁴	Section 22.2.2	kJ/m ³	D,Pr
SOLID MASS FRACTION ⁵	Section 22.2.2	kg/kg	D,Pr
SOLID SPECIFIC HEAT ⁴	Section 22.2.2	kJ/(kg K)	D,Pr
SUBSTEPS	Section 8.3.8		B,D
SURFACE DENSITY ⁴	Section 22.10.17	kg/m ²	B,D
SURFACE DEPOSITION ¹	Section 13.4	kg/m ²	B,D
TOTAL MASS FLUX WALL ¹	Section 22.10.10	kg/s/m ²	B,D
VELOCITY ERROR	Section 21	m/s	B,D
VISCOUS STRESS WALL	Section 22.10.21	Pa	B,D
VISCOUS WALL UNITS	Section 22.10.28		B,D
WALL ENTHALPY	$\int \rho_s c_s T dV_s$	kJ	B,D
WALL PRESSURE	Section 22.10.21	Pa	B,D
WALL TEMPERATURE	Surface temperature	°C	B,D
WALL THICKNESS	Section 22.10.17	m	B,D
WALL VISCOSITY	Near-wall viscosity, μ_w	kg/(ms)	B,D

- ¹ Requires the specification of a gas species using SPEC_ID
² Requires the specification of a particle name using PART_ID
³ Requires specification of an additional scalar using QUANTITY2.
⁴ Allows for an optional MATL_ID
⁵ Requires a MATL_ID

22.14 Device, Control, and Other Miscellaneous Output Quantities

Table 22.6 below lists output quantities associated with devices and controls, as well as other miscellaneous output. The File Type “D” refers to the device output file `CHID_devc.csv`. The File Type “S” refers to a contour or “slice” file.

Table 22.6: Output quantities for devices, controls, and miscellaneous functions.

QUANTITY	Explanation	Units	File Type
ACTUATED SPRINKLERS	Section 22.10.26		D
ASPIRATION	Section 18.3.7	%/m	D
CHAMBER OBSCURATION	Section 18.3.5	%/m	D
CELL INDEX I	Mesh cell index in x		D,S
CELL INDEX J	Mesh cell index in y		D,S
CELL INDEX K	Mesh cell index in z		D,S
CONTROL	Section 18.5		D
CONTROL VALUE	Section 18.5		D
CPU TIME	Section 22.10.26	s	D
FED	Section 22.10.18		D
FIC	Section 22.10.18		D,S
FIRE DEPTH	Section 18.3.8	m	D
ITERATION	Section 22.10.26		D
LAYER HEIGHT	Section 22.10.7	m	D
LINK TEMPERATURE	Section 18.3.4	°C	D
LOWER TEMPERATURE	Section 22.10.7	°C	D
NUMBER OF PARTICLES	Section 22.10.26		D
OPEN NOZZLES	Section 22.10.26		D
PATH OBSCURATION	Section 18.3.6	%	D
RAM	Section 22.10.26	MB	D
RANDOM NUMBER	Uniform random variable over [0,1]		D
SPRINKLER LINK TEMPERATURE	Section 18.3.1	°C	D
THERMOCOUPLE	Section 22.10.8	°C	D
TIME	Section 22.2	s	D
TIME STEP	Section 22.10.26	s	D
TRANSMISSION	Section 18.3.6	%/m	D
UPPER TEMPERATURE	Section 22.10.7	°C	D
WALL CLOCK TIME	Section 22.10.26	s	D
WALL CLOCK TIME ITERATIONS	Section 22.10.26	s	D

22.15 Droplet and Particle Output Quantities

Table 22.7 below lists some less often used output quantities. These are mainly used for diagnostic purposes. Explanations for most can be found in Volume 1 of the FDS Technical Reference Guide [3].

Table 22.7: Particle and droplet output quantities.

QUANTITY	Explanation	Units	File Type
ADA ²	Avg. Droplet (cross sectional) Area	m ² /m ³	D,I,P,S
ADA_ _z	Avg. Droplet (cross sectional) Area	m ² /m ³	D,I,P,S
ADD ²	Avg. Droplet Diameter	μm	D,I,P,S
ADD_ _z	Avg. Droplet Diameter	μm	D,I,P,S
ADT ²	Avg. Droplet Temperature	°C	D,I,P,S
ADT_ _z	Avg. Droplet Temperature	°C	D,I,P,S
DROPLET VOLUME FRACTION ²	Section 22.9.3		D,P,S
MPUV ³	Section 22.9.3	kg/m ³	D,P,S
MPUV_ _z	Section 22.9.3	kg/m ³	D,P,S
PARTICLE ACCEL X ²	Section 22.9.4	m/s ²	PA
PARTICLE ACCEL Y ²	Section 22.9.4	m/s ²	PA
PARTICLE ACCEL Z ²	Section 22.9.4	m/s ²	PA
PARTICLE AGE	Section 22.9.4	s	PA
PARTICLE BULK DENSITY	Section 22.9.4	kg/(m ³)	PA
PARTICLE DIAMETER	Section 22.9.4	μm	PA
PARTICLE DRAG COEFFICIENT ²	Section 22.9.4		PA
PARTICLE DRAG FORCE X ²	Section 22.9.4	N	PA
PARTICLE DRAG FORCE Y ²	Section 22.9.4	N	PA
PARTICLE DRAG FORCE Z ²	Section 22.9.4	N	PA
PARTICLE FLUX X ²	Section 22.9.3	kg/(m ² s)	P,S
PARTICLE FLUX Y ²	Section 22.9.3	kg/(m ² s)	P,S
PARTICLE FLUX Z ²	Section 22.9.3	kg/(m ² s)	P,S
PARTICLE HEAT TRANSFER COEFFICIENT ²	Section 22.9.4	W/m ² /K	PA
PARTICLE MASS	Section 22.9.4	kg	PA
PARTICLE PHASE	Section 22.9.4		PA
PARTICLE RADIATION LOSS	Section 22.9.3	kW/m ³	D,I,P,S
PARTICLE TEMPERATURE	Section 22.9.4	°C	PA
PARTICLE U	Section 22.9.4	m/s	PA
PARTICLE V	Section 22.9.4	m/s	PA
PARTICLE VELOCITY	Section 22.9.4	m/s	PA
PARTICLE W	Section 22.9.4	m/s	PA
PARTICLE WEIGHTING FACTOR	Section 22.9.4		PA
PARTICLE X	Section 22.9.4	m	PA
PARTICLE Y	Section 22.9.4	m	PA
PARTICLE Z	Section 22.9.4	m	PA
PDPA	Droplet statistics, Section 22.10.16		D
QABS ²	Absorption efficiency of droplets		D,I,P,S
QABS_ _z	Absorption efficiency of droplets		D,I,P,S

Table 22.7: Particle and droplet output quantities (continued).

QUANTITY	Explanation	Units	File Type
QSCA ²	Scattering efficiency of droplets		D,I,P,S
QSCA_ _Z ¹	Scattering efficiency of droplets		D,I,P,S

¹ Requires the specification of a gas species using `SPEC_ID`

² Requires the specification of a particle class using `PART_ID`

³ Requires the specification of a non-zero mass particle class using `PART_ID`

22.16 Summary of HVAC Output Quantities

Table 22.8 summarizes the various output quantities for HVAC systems. In the file type column, D indicates a device output specified with a `DEVC` input, and H indicates a Smokeview output specified with an `HVAC` input.

All HVAC output quantities can be selected with a `DEVC` input. Do not specify an `XYZ` or `XB` for HVAC outputs. When using a `DEVC`, quantities for a duct require the specification of a `DUCT_ID`, quantities for a node require the specification of a `NODE_ID`, and quantities for a duct cell (when `HVAC_MASS_TRANSPORT_CELL_L` or `N_CELLS` are specified) require specification of both a `DUCT_ID` and a `CELL_L` (distance along the duct in meters from the first node where the desired cell is located). Mass and volume fraction outputs also require the specification of a `SPEC_ID`. Fan and aircoil outputs require the `DUCT_ID` of the duct they are located in. Filter outputs require the `NODE_ID` of the node they are located in. The quantity `DUCT ENTHALPY FLOW` applies Eq. 22.32 to the flow in the duct. To have the node output `NODE ENTHALPY` reflect the duct quantity of `DUCT ENTHALPY FLOW` set `RELATIVE=.TRUE` for the node output. The quantity `NODE PRESSURE DIFFERENCE` requires that one specify both elements of the array `NODE_ID`, and the pressure difference is calculated by subtracting the first node from the second.

HVAC output quantities beginning with `DUCT` that are not duct cell quantities and output quantities beginning with `NODE` except for `NODE PRESSURE DIFFERENCE` can also be output to the `CHID.hvac` file that Smokeview uses to visualize duct quantities (file format description in Section 27.21). These are identified in Table 22.8 as a File Type of H. Smokeview outputs are specified using the `HVAC` namelist. To select duct outputs create a single `HVAC` input with `TYPE_ID='DUCT QUANTITY LIST'`, and to select node outputs create a single `HVAC` input with `TYPE_ID='NODE QUANTITY LIST'`. Specify a list of up to 20 outputs using `QUANTITY`, and if any quantities require a species also specify `QUANTITY_SPEC_ID` for that quantities. For duct outputs, if the duct is divided into cells, then Smokeview outputs are written for each cell. The output interval can be set using `DT_HVAC` on `DUMP`. For example:

```
&HVAC TYPE_ID='DUCT QUANTITY LIST',
      QUANTITY='DUCT VELOCITY','DUCT LOSS','DUCT VOLUME FRACTION','DUCT VOLUME
      FRACTION',
      QUANTITY_SPEC_ID(3)='OXYGEN',QUANTITY_SPEC_ID(4)='SOOT' /
```

would output the velocity, flow loss, and the volume fractions of oxygen and soot in all ducts.

Table 22.8: HVAC output quantities.

QUANTITY	Explanation	Units	File Type
AIRCOIL HEAT EXCHANGE	Heat exchange rate for an aircoil	kW	D
DUCT DENSITY	Density of the flow in a duct	kg/m ³	D,H
DUCT CELL DENSITY	Gas density in a duct cell	kg/m ³	D
DUCT CELL MASS FRACTION	Mass fraction of a species in a duct cell	kg/kg	D
DUCT CELL TEMPERATURE	Gas temperature in a duct cell	°C	D
DUCT CELL VOLUME FRACTION	Volume fraction of a species in a duct cell	mol/mol	D
DUCT ENTHALPY FLOW	Enthalpy flow in a duct	kW	D,H
DUCT LOSS	Total flow loss coefficient for a duct		D,H
DUCT MASS FLOW	Mass flow in a duct	kg/s	D,H
DUCT MASS FRACTION	Mass fraction of a species in a duct	kg/kg	D,H
DUCT TEMPERATURE	Temperature of the flow in a duct	°C	D,H
DUCT VELOCITY	Velocity of a duct	m/s	D,H
DUCT VOLUME FLOW	Volumetric flow in a duct	m ³ /s	D,H
DUCT VOLUME FRACTION	Volume fraction of a species in a duct	mol/mol	D,H
FAN PRESSURE	Pressure output of a fan in a duct	Pa	D
FILTER LOADING	Loading of a species in a filter	kg	D
FILTER LOSS	Flow loss through a filter		D
NODE DENSITY	Density of the flow through a node	kg/m ³	D,H
NODE ENTHALPY	Enthalpy of a node	kJ/kg	D,H
NODE SENSIBLE ENTHALPY	Sensible Enthalpy of a node	kJ/kg	D,H
NODE MASS FRACTION	Mass fraction of a species in a node	kg/kg	D,H
NODE PRESSURE	Pressure of a node	Pa	D,H
NODE PRESSURE DIFFERENCE	Pressure difference between two nodes	Pa	D
NODE TEMPERATURE	Temperature of the flow though a node	°C	D,H
NODE VOLUME FRACTION	Volume fraction of a species in a node	mol/mol	D,H

Chapter 23

Alphabetical List of Input Parameters

This chapter lists all of the input parameters for FDS in separate tables grouped by namelist, these tables are in alphabetical order along with the parameters within them. This is intended to be used as a quick reference and does not replace reading the detailed description of the parameters in the main body of this guide. See Table 5.1 for a cross-reference of relevant sections and the tables in this chapter. The reason for this statement is that many of the listed parameters are mutually exclusive – specifying more than one can cause the program to either fail or run in an unpredictable manner. Also, some of the parameters trigger the code to work in a certain mode when specified. For example, specifying the thermal conductivity of a solid surface triggers the code to assume the material to be thermally-thick, mandating that other properties be specified as well. Simply prescribing as many properties as possible from a handbook is bad practice. Only prescribe those parameters which are necessary to describe the desired scenario. Note that you may use the character string `FYI` on any namelist line to make a note or comment.

23.1 BACK (Background species)

Table 23.1: For more information see Section 12.1.1.

BACK (Background Species Parameters)				
Keyword	Type	Description	Units	Default
SPEC_ID	Char. Array	Section 12.1.1		
MASS_FRACTION	Real Array	Section 12.1.1		

23.2 BNDF (Boundary File Parameters)

Table 23.2: For more information see Section 22.5.

BNDF (Boundary File Parameters)				
Keyword	Type	Description	Units	Default
CELL_CENTERED	Logical	Section 22.5		F
MATL_ID	Character	Section 22.13		
PART_ID	Character	Section 22.13		
PROP_ID	Character	Section 22.5		
QUANTITY	Character	Section 22.13		
SPEC_ID	Character	Section 22.13		
TEMPORAL_STATISTIC	Character	Section 22.5		

23.3 CATF (Concatenate Input Files Parameters)

Table 23.3: For more information see Section 5.4.

CATF (Concatenate Input Files Parameters)				
Keyword	Type	Description	Units	Default
OTHER_FILES	Character Array	Section 5.4		

23.4 CLIP (Clipping Parameters)

Table 23.4: For more information see Section 19.5.

CLIP (Specified Upper and Lower Limits)				
Keyword	Type	Description	Units	Default
CLIP_DT_RESTRICTIONS_MAX	Integer	Section 19.5.2		5
MAXIMUM_DENSITY	Real	Section 19.5.2	kg/m ³	
MAXIMUM_TEMPERATURE	Real	Section 19.5.1	°C	
MINIMUM_DENSITY	Real	Section 19.5.2	kg/m ³	
MINIMUM_TEMPERATURE	Real	Section 19.5.1	°C	

23.5 COMB (General Combustion Parameters)

Table 23.5: For more information see Chapter 13.

COMB (General combustion parameters)				
Keyword	Type	Description	Units	Default
CHECK_REALIZABILITY	Logical	Section 13.3.7		F
COMPUTE_ADIABATIC_FLAME_TEMPERATURE	Logical	Section 13.1.6		F
DO_CHEM_LOAD_BALANCE	Logical	Section 13.3.7		F
EQUIV_RATIO_CHECK	Logical	Section 13.3.7		T
EXTINCTION_MODEL	Character	Section 13.1.6		
FINITE_RATE_MIN_TEMP	Real	Section 13.3	°C	-273.15
FIXED_MIX_TIME	Real	Section 13.1.5	s	
FREE_BURN_TEMPERATURE	Real	Section 13.1.6	°C	600
INITIAL_UNMIXED_FRACTION	Real	Section 13.1.5		1.0
MAX_CHEMISTRY_SUBSTEPS	Integer	Section 13.3.7		20
MAX_EQUIV_RATIO	Real	Section 13.3.7		10.0
MIN_EQUIV_RATIO	Real	Section 13.3.7		0.2.
N_FIXED_CHEMISTRY_SUBSTEPS	Integer	Section 13.3.7		-1
ODE_SOLVER	Character	Section 13.3.7		
ODE_MIN_ATOL	Real	Section 13.3.7		
ODE_REL_ERROR	Real	Section 13.3.7		
SUPPRESSION	Logical	Section 13.1.6		T
TAU_CHEM	Real	Section 13.1.5		1.E-10
TAU_FLAME	Real	Section 13.1.5		1.E10
ZZ_MIN_GLOBAL	Real	Section 13.3		1.E-10

23.6 CSVF (Comma Separated Velocity Files)

Table 23.6: For more information see Section 20.5.

CSVF (Comma Delimited Initialization Files)				
Keyword	Type	Description	Units	Default
SPECFILE	Character	Section 20.5		
TMPFILE	Character	Section 20.5		
UVWFILE	Character	Section 20.5		

23.7 CTRL (Control Function Parameters)

Table 23.7: For more information see Section 18.5.

CTRL (Control Function Parameters)				
Keyword	Type	Description	Units	Default
CONSTANT	Real	Section 18.5.6		
CONTROL_FORCE (3)	Logical Array	Section 18.5.7		F
DELAY	Real	Section 18.5.10	s	0.
DIFFERENTIAL_GAIN	Real	Section 18.5.7		0.
FUNCTION_TYPE	Character	Section 18.4		
ID	Character	Section 18.5		
INITIAL_STATE	Logical	Section 18.4		F
INPUT_ID	Char. Array	Section 18.5		
INTEGRAL_GAIN	Real	Section 18.5.7		0.
LATCH	Logical	Section 18.4		T
N	Integer	Section 18.5		1
ON_BOUND	Character	Section 18.5.3		LOWER
PERCENTILE	Real	Section 18.5.8		
PROPORTIONAL_GAIN	Real	Section 18.5.7		0.
RAMP_ID	Character	Section 18.5.5		
SETPOINT (2)	Real	Section 18.4		
TARGET_VALUE	Real	Section 18.5.7		0.
TRIP_DIRECTION	Integer	Section 18.4		1

23.8 DEVC (Device Parameters)

Table 23.8: For more information see Section 18.1.

DEVC (Device Parameters)				
Keyword	Type	Description	Units	Default
ABSOLUTE_VALUE	Logical	Section 18.2		F
BYPASS_FLOWRATE	Real	Section 18.3.7	kg/s	0
CELL_L	Real	Section 22.16	m	
CONVERSION_ADDEND	Real	Section 18.2		0
CONVERSION_FACTOR	Real	Section 18.2		1
COORD_FACTOR	Real	Section 22.2.5		1
CTRL_ID	Character	Section 18.6.1		
DB	Character	Section 22.2.3		
DELAY	Real	Section 18.3.7	s	0
DEPTH	Real	Section 22.2.2	m	0
DEVC_ID	Character	Sections 18.3.7 and 18.6.1		
D_ID	Character	Section 22.2.5		

Table 23.8: Continued

DEVC (Device Parameters)				
Keyword	Type	Description	Units	Default
DRY	Logical	Section 22.10.22		F
DUCT_ID	Character	Section 10.2		
DX	Real	Section 22.2.5	m	0
DY	Real	Section 22.2.5	m	0
DZ	Real	Section 22.2.5	m	0
FLOWRATE	Real	Section 18.3.7	kg/s	0
FORCE_DIRECTION	Real(3)	Section 22.10.21		
HIDE_COORDINATES	Logical	Section 22.2.5		F
ID	Character	Section 18.1		
INITIAL_STATE	Logical	Section 18.4		F
INIT_ID	Character	Section 15.4		
IOR	Integer	Section 18.1		
LATCH	Logical	Section 18.4		T
LP_TAG	Integer	Section 22.9.5		
MATL_ID	Character	Section 22.10.17		
MOVE_ID	Character	Section 22.2.5		
N_INTERVALS	Integer	Section 22.2.4		10
NODE_ID	Character(2)	Section 10.2		
NO_UPDATE_CTRL_ID	Character	Section 18.6.3		
NO_UPDATE_DEVC_ID	Character	Section 18.6.3		
ORIENTATION	Real Triplet	Section 18.1		0,0,-1
OUTPUT	Logical	Section 18.2		T
PART_ID	Character	Section 22.15		
PIPE_INDEX	Integer	Section 18.3.1		1
POINTS	Integer	Section 22.2.5		1
POINTS_ARRAY_X	Real Array	Section 22.2.5	m	
POINTS_ARRAY_Y	Real Array	Section 22.2.5	m	
POINTS_ARRAY_Z	Real Array	Section 22.2.5	m	
PROP_ID	Character	Section 18.1		
QUANTITY	Character	Section 18.1		
QUANTITY2	Character	Section 22.2.5		
QUANTITY_RANGE	Real(2)	Section 22.2.3		-1.E50,1.E50
REAC_ID	Character	Section 22.12		
RELATIVE	Logical	Section 18.2		F
R_ID	Character	Section 22.2.5		
ROTATION	Real	Section 18.1	deg.	0
SETPOINT	Real	Section 18.4		
SMOOTHING_FACTOR	Real	Section 18.4		0
SMOOTHING_TIME	Real	Section 18.4	s	
SPATIAL_STATISTIC	Character	Section 22.2.3		
SPEC_ID	Character	Section 22.12		
STATISTICS_END	Real	Section 22.2.4	s	T_BEGIN

Table 23.8: Continued

DEVC (Device Parameters)				
Keyword	Type	Description	Units	Default
STATISTICS_START	Real	Section 22.2.4	s	T_BEGIN
SURF_ID	Character	Section 22.2.3		
TEMPORAL_STATISTIC	Character	Section 22.2.3		
TIME_AVERAGED	Logical	Section 18.2		
TIME_HISTORY	Logical	Section 22.2.5		
TIME_PERIOD	Real	Section 22.2.4	s	
TRIP_DIRECTION	Integer	Section 18.4		1
UNITS	Character	Section 18.2		
VELO_INDEX	Integer	Section 22.10.24		0
XB (6)	Real Sextuplet	Section 22.2.3	m	
XBP (6)	Real Sextuplet	Section 22.2.5	m	
XYZ (3)	Real Triplet	Section 18.1	m	
X_ID	Character	Section 22.2.5		ID-x
Y_ID	Character	Section 22.2.5		ID-y
Z_ID	Character	Section 22.2.5		ID-z
XYZ_UNITS	Character	Section 22.2.5		'm'

23.9 DUMP (Output Parameters)

Table 23.9: For more information see Chapter 22.

DUMP (Output Parameters)				
Keyword	Type	Description	Units	Default
CFL_FILE	Logical	Section 6.2.2		F
CLIP_RESTART_FILES	Logical	Section 5.6		T
COLUMN_DUMP_LIMIT	Logical	Section 18.2		F
CTRL_COLUMN_LIMIT	Integer	Section 18.2		254
DEVC_COLUMN_LIMIT	Integer	Section 18.2		254
DIAGNOSTICS_INTERVAL	Integer	Section 3.4		100
DT_BNDF	Real	Section 22.1	s	$\Delta t / \text{NFRAMES}$
DT_CPU	Real	Section 27.6	s	$2\Delta t$
DT_CTRL	Real	Section 22.1	s	$\Delta t / \text{NFRAMES}$
DT_DEVC	Real	Section 22.1	s	$\Delta t / \text{NFRAMES}$
DT_FLUSH	Real	Section 22.1	s	$\Delta t / \text{NFRAMES}$
DT_HRR	Real	Section 22.1	s	$\Delta t / \text{NFRAMES}$
DT_HVAC	Real	Section 22.1	s	$\Delta t / \text{NFRAMES}$
DT_ISOF	Real	Section 22.1	s	$\Delta t / \text{NFRAMES}$
DT_MASS	Real	Section 22.1	s	$\Delta t / \text{NFRAMES}$
DT_PART	Real	Section 22.1	s	$\Delta t / \text{NFRAMES}$

Table 23.9: Continued

DUMP (Output Parameters)				
Keyword	Type	Description	Units	Default
DT_PL3D	Real	Section 22.1	s	$2\Delta t$
DT_PROF	Real	Section 22.1	s	$\Delta t / \text{NFRAMES}$
DT_RADF	Real	Section 22.10.14	s	$2\Delta t$
DT_RESTART	Real	Section 22.1	s	$2\Delta t$
DT_SL3D	Real	Section 22.1	s	$2\Delta t$
DT_SLCF	Real	Section 22.1	s	$\Delta t / \text{NFRAMES}$
DT_SMOKE3D	Real	Section 22.1	s	$\Delta t / \text{NFRAMES}$
SMV_PARALLEL_WRITE	Logical	Section 22		T
DT_SPEC	Real	Section 20.5	s	
DT_TMP	Real	Section 20.5	s	
DT_UVW	Real	Section 20.5	s	
FLUSH_FILE_BUFFERS	Logical	Section 22		T
HRRPUV_MAX_SMV	Real	Section 22.8	kW/m^3	1200
MASS_FILE	Logical	Section 22		F
MAXIMUM_PARTICLES	Integer	Section 15.5		1000000
NFRAMES	Integer	Section 22		1000
PLOT3D_PART_ID(5)	Char. Quint	Section 22.7		
PLOT3D_QUANTITY(5)	Char. Quint	Section 22.7		
PLOT3D_SPEC_ID(5)	Char. Quint	Section 22.7		
PLOT3D_VELO_INDEX	Int. Quint	Section 22.10.24		0
RAMP_BNDF	Character	Section 22.1		
RAMP_CPU	Character	Section 27.6		
RAMP_CTRL	Character	Section 22.1		
RAMP_DEVC	Character	Section 22.1		
RAMP_FLUSH	Character	Section 22.1		
RAMP_HRR	Character	Section 22.1		
RAMP_HVAC	Character	Section 22.1		
RAMP_ISOFC	Character	Section 22.1		
RAMP_MASS	Character	Section 22.1		
RAMP_PART	Character	Section 22.1		
RAMP_PL3D	Character	Section 22.1		
RAMP_PROF	Character	Section 22.1		
RAMP_RADF	Character	Section 22.10.14		
RAMP_RESTART	Character	Section 22.1		
RAMP_SL3D	Character	Section 22.1		
RAMP_SLCF	Character	Section 22.1		
RAMP_SMOKE3D	Character	Section 22.1		
RAMP_SPEC	Character	Section 22.1		
RAMP_TMP	Character	Section 22.1		
RAMP_UVW	Character	Section 22.1		
RENDER_FILE	Character	Reference [2]		
RESULTS_DIR	Character	Section 22		

Table 23.9: Continued

DUMP (Output Parameters)				
Keyword	Type	Description	Units	Default
SIG_FIGS	Integer	Section 22.10.27		8
SIG_FIGS_EXP	Integer	Section 22.10.27		3
SMOKE3D	Logical	Section 22.8		T
STATUS_FILES	Logical	Section 22		F
SUPPRESS_DIAGNOSTICS	Logical	Section 3.4		F
TEMP_MAX_SMV	Real	Section 22.8	°C	2000
TEMP_MIN_SMV	Real	Section 22.8	°C	2000
VELOCITY_ERROR_FILE	Logical	Section 22.10.26		F
WRITE_XYZ	Logical	Section 22.7		F

$$\Delta t = T_{\text{END}} - T_{\text{BEGIN}}$$

23.10 GEOM (Unstructured Geometry Parameters)

Table 23.10: For more information see Section 7.3.

GEOM (Unstructured Geometry Parameters)				
Keyword	Type	Description	Units	Default
BINARY_FILE	Character	Section 7.3.9		'null'
BNDF_GEOM	Logical	Section 22.5		T
CELL_BLOCK_IOR	Integer	Section 7.3.10		0
CELL_BLOCK_ORIENTATION	Real Triplet	Section 7.3.10		0.0,0.0,0.0
COLOR	Character	Section 7.5		'null'
CYLINDER_LENGTH	Real	Section 7.3.6	m	
CYLINDER_RADIUS	Real	Section 7.3.6	m	
CYLINDER_ORIGIN	Real Triplet	Section 7.3.6	m	
CYLINDER_AXIS	Real Triplet	Section 7.3.6		
CYLINDER_NSEG_AXIS	Integer	Section 7.3.6		
CYLINDER_NSEG_THETA	Integer	Section 7.3.6		
EXTEND_TERRAIN	Logical	Section 7.3.6		F
EXTRUDE	Real	Section 7.3.6	m	
FACES	Array of Integer Triplets	Section 7.3		
ID	Character	Section 7.3		'geom'_{#}
IJK	Integer Triplet	Section 7.3.6		0,0,0
IS_TERRAIN	Logical	Section 7.3.6		F
MOVE_ID	Character	Section 7.3.8		'null'
N_LAT	Integer	Section 7.3.6		0
N_LEVELS	Integer	Section 7.3.6		0
N_LONG	Integer	Section 7.3.6		0
POLY	Integer Array	Section 7.3.6		

Table 23.10: Continued

GEOM (Unstructured Geometry Parameters)				
Keyword	Type	Description	Units	Default
RGB (3)	Integer Triplet	Section 7.5		
SPHERE_ORIGIN	Real Triplet	Section 7.3.6	m	
SPHERE_RADIUS	Real	Section 7.3.6	m	
SPHERE_TYPE	Integer	Section 7.3.6		
SURF_ID	Character	Section 7.3		'INERT'
SURF_ID6 (6)	Character Sextuplet	Section 7.2.1		
SURF_IDS (3)	Character Triplet	Section 7.2.1		
TEXTURE_MAPPING	Character	Section 7.5.2		'RECTANGULAR'
TEXTURE_ORIGIN	Real Triplet	Section 7.5.2	m	0.0,0.0,0.0
TEXTURE_SCALE	Real	Section 7.5.2		1.0
TRANSPARENCY	Real	Section 7.5		1.0
VERTS	Array of Real Triplets	Section 7.3	m	
XB (6)	Real Array	Section 7.3.6	m	
ZMIN	Real	Section 7.3.6	m	
ZVALS	Real Array	Section 7.3.6	m	
ZVAL_HORIZON	Real	Section 7.3.6	m	

23.11 HEAD (Header Parameters)

Table 23.11: For more information see Section 6.1.

HEAD (Header Parameters)				
Keyword	Type	Description	Units	Default
CHID	Character	Section 6.1		'output'
TITLE	Character	Section 22.7		

23.12 HOLE (Obstruction Cutout Parameters)

Table 23.12: For more information see Section 7.2.8.

HOLE (Obstruction Cutout Parameters)				
Keyword	Type	Description	Units	Default
COLOR	Character	Section 7.5		
CTRL_ID	Character	Section 7.2.8		
DEVC_ID	Character	Section 7.2.8		
ID	Character	Identifier for input line		

Table 23.12: Continued

HOLE (Obstruction Cutout Parameters)				
Keyword	Type	Description	Units	Default
MULT_ID	Character	Section 7.6		
RGB (3)	Integer Triplet	Section 7.5		
TRANSPARENCY	Real	Section 7.2.8		
XB (6)	Real Sextuplet	Section 7.6	m	

23.13 HVAC (HVAC System Definition)

Table 23.13: For more information see Section 10.2.

HVAC (HVAC System Definition)				
Keyword	Type	Description	Units	Default
AIRCOIL_ID	Character	Section 10.2.1		
AMBIENT	Logical	Section 10.2.3		F
AREA	Real	Section 10.2.1	m ²	
CLEAN_LOSS	Real	Section 10.2.5		
COOLANT_MASS_FLOW	Real	Section 10.2.6	kg/s	
COOLANT_SPECIFIC_HEAT	Real	Section 10.2.6	kJ/(kg K)	
COOLANT_TEMPERATURE	Real	Section 10.2.6	°C	
CTRL_ID	Character	Sections 10.2.1, 10.2.4, 10.2.5		
DAMPER	Logical	Sections 10.2.1, 10.2.2		F
DEVC_ID	Character	Sections 10.2.1, 10.2.4, 10.2.5		
DIAMETER	Real	Section 10.2.1	m	
DISCHARGE_COEFFICIENT	Real	Section 10.3.2		1.
DUCT_ID	Char. Array	Section 10.2.3		
EFFICIENCY	Real Array	Sections 10.2.5, 10.2.6		1.0
FAN_ID	Character	Section 10.2.1		
FILTER_ID	Character	Section 10.2.3		
FIXED_Q	Real	Section 10.2.6	kW	
ID	Character	Section 10.2		
LEAK_ENTHALPY	Logical	Section 10.3.2		F
LEAK_PRESSURE_EXPONENT	Real	Section 10.3.2		0.5
LEAK_REFERENCE_PRESSURE	Real	Section 10.3.2	Pa	4
LENGTH	Real	Section 10.2.1	m	
LOADING	Real Array	Section 10.2.5	kg	0.0
LOADING_MULTIPLIER	Real Array	Section 10.2.5	1/kg	1.0
LOSS	Real Array	Sections 10.2.1 – 10.2.5		0.0
MASS_FLOW	Real	Section 10.2.1	kg/s	
MAX_FLOW	Real	Section 10.2.4	m ³ /s	
MAX_PRESSURE	Real	Section 10.2.4	Pa	

Table 23.13: Continued

HVAC (HVAC System Definition)				
Keyword	Type	Description	Units	Default
N_CELLS	Integer	Section 10.2.8		10×LENGTH
NETWORK_ID	Character	Section 10.2		
NODE_ID	Char. Doublet	Section 10.2.1		
PERIMETER	Real	Section 10.2.1	m	
QUANTITY	Char. Array	Sections 22.16		
QUANTITY_SPEC_ID	Char. Array	Sections 22.16		
RAMP_ID	Character	Sections 10.2.1, 10.2.5, 10.2.4		
RAMP_LOSS	Character	Sections 10.2.1, 10.2.2		
REVERSE	Logical	Section 10.2.1		F
ROUND	Logical	Section 10.2.1		T
ROUGHNESS	Real	Section 10.2.1	m	0.0
SPEC_ID	Char. Array	Section 10.2.5		
SQUARE	Logical	Section 10.2.1		T
TAU_AC	Real	Section 10.2.6	s	1.0
TAU_FAN	Real	Section 10.2.4	s	1.0
TAU_VF	Real	Section 10.2.1	s	1.0
TRANSPORT_PARTICLES	Logical	Section 10.3.2	s	F
TYPE_ID	Character	Section 10.2		
VENT_ID	Character	Section 10.2.3		
VENT2_ID	Character	Section 10.3.2		
VOLUME_FLOW	Real	Section 10.2.1, 10.2.4	m ³ /s	
WAYPOINTS	Real Array	Section 10.2.1	m	
XYZ	Real Triplet	Section 10.2.3	m	0.0

23.14 INIT (Initial Conditions)

Table 23.14: For more information see Section 20.

INIT (Initial Conditions)				
Keyword	Type	Description	Units	Default
BULK_DENSITY_FACTOR	Real	Section 17.2.2		1.
BULK_DENSITY_FILE	Character	Section 17.2.2		
CELL_CENTERED	Logical	Section 15.5.3		F
CROWN_BASE_HEIGHT	Real	Section 17.2.1	m	
CROWN_BASE_WIDTH	Real	Section 17.2.1	m	
CTRL_ID	Character	Section 15.5.3		
DB	Character	Section 20.1		
DEVC_ID	Character	Section 15.5.3		
DIAMETER	Real	Section 15.5.3	μm	

Table 23.14: Continued

INIT (Initial Conditions)				
Keyword	Type	Description	Units	Default
DRY	Logical	Section 17.2		F
DT_INSERT	Real	Section 15.5.3	s	
DX	Real	Section 15.5.3	m	0.
DY	Real	Section 15.5.3	m	0.
DZ	Real	Section 15.5.3	m	0.
HEIGHT	Real	Section 15.5.3	m	
HRRPUV	Real	Section 20.3	kW/m ³	
ID	Character	Section 15.4		
INNER_RADIUS	Real	Section 15.5.3	m	0.
MASS_FRACTION(:)	Real Array	Section 20.1	kg/kg	Ambient
MASS_PER_TIME	Real	Section 15.5.3	kg/s	
MASS_PER_VOLUME	Real	Section 15.5.3	kg/m ³	1
MULT_ID	Character	Section 7.6		
NODE_ID	Character	Section 10.2.8		
N_PARTICLES	Integer	Section 15.5.3		0
N_PARTICLES_PER_CELL	Integer	Section 15.5.3		0
ORIENTATION_RAMP(3)	Character	Section 15.5.4		
PART_ID	Character	Section 15.5.3		
PARTICLE_WEIGHT_FACTOR	Real	Section 15.5.3		1.
PATH_RAMP(3)	Character	Section 15.5.3		
RADIATIVE_FRACTION	Real	Section 20.3		0.
RADIUS	Real	Section 15.5.3	m	
RAMP_MF_Z	Character	Section 20.1		
RAMP_PART	Character	Section 15.5.3		
RAMP_Q	Character	Section 20.3		
RAMP_TMP_Z	Character	Section 20.2		
RAMP_VF_Z	Character	Section 20.1		
SHAPE	Character	Section 15.5.3		'BLOCK'
SPEC_ID(N)	Character Array	Section 20.1		
TEMPERATURE	Real	Section 20.2	°C	TMFA
TREE_HEIGHT	Real	Section 17.2.1	m	
UNIFORM	Logical	Section 15.5.3		F
UVW(3)	Real Array	Section 15.5.3	m/s	0.
VOLUME_FRACTION(:)	Real Array	Section 20.1	mol/mol	Ambient
XB(6)	Real Array	Section 20.1	m	
XYZ(3)	Real Array	Section 15.5.3	m	

23.15 ISO (Isosurface Parameters)

Table 23.15: For more information see Section 22.6.

ISO (Isosurface Parameters)				
Keyword	Type	Description	Units	Default
DELTA	Real	Section 22.6		
QUANTITY	Character	Section 22.6		
QUANTITY2	Character	Section 22.6		
SKIP	Character	Section 22.6		
SPEC_ID	Character	Section 22.6		
SPEC_ID2	Character	Section 22.6		
VALUE (:)	Real Array	Section 22.6		
VELO_INDEX	Integer	Section 22.10.24		0
VELO_INDEX2	Integer	Section 22.10.24		0

23.16 MATL (Material Properties)

Table 23.16: For more information see Section 8.3.

MATL (Material Properties)				
Keyword	Type	Description	Units	Default
A (:)	Real Array	Section 9.2	1/s	
ABSORPTION_COEFFICIENT	Real	Section 8.3.2	1/m	50000.
ADJUST_H	Logical	Section 9.4		T
ALLOW_SHRINKING	Logical	Section 9.2.4		T
ALLOW_SWELLING	Logical	Section 9.2.4		T
BOILING_TEMPERATURE	Real	Section 9.2.7	°C	5000.
CONDUCTIVITY	Real	Section 8.3.2	W/(m K)	0.
CONDUCTIVITY_RAMP	Character	Section 8.3.2		
DENSITY	Real	Section 8.3.2	kg/m ³	0.
E (:)	Real Array	Section 9.2	J/mol	
EMISSION	Real	Section 8.3.2		0.9
GAS_DIFFUSION_DEPTH (:)	Real Array	Section 9.2	m	0.001
HEATING_RATE (:)	Real Array	Section 9.2	°C/min	5.
HEAT_OF_COMBUSTION (: , :)	Real Array	Section 9.2	kJ/kg	
HEAT_OF_REACTION (:)	Real Array	Section 9.2	kJ/kg	0.
ID	Character	Section 8.1		
MATL_ID (: , :)	Character	Section 9.2		
MAX_REACTION_RATE (:)	Real Array	Section 9.2.3	kg/(m ³ s)	∞
MW	Real	Section 9.2.7	g/mol	
N_O2 (:)	Real Array	Section 9.2		0.
N_REACTIONS	Integer	Section 9.2		0

Table 23.16: Continued

MATL (Material Properties)				
Keyword	Type	Description	Units	Default
N_S (:)	Real Array	Section 9.2		1.
N_T (:)	Real Array	Section 9.2		0.
NU_MATL (: , :)	Real Array	Section 9.2	kg/kg	0.
NU_PART (: , :)	Real Array	Section 9.2	kg/kg	0.
NU_SPEC (: , :)	Real Array	Section 9.2	kg/kg	0.
PART_ID (: , :)	Char. Array	Section 9.2		
PYROLYSIS_RANGE (:)	Real Array	Section 9.2.3	°C	80.
REAC_RATE_DELTA	Real	Section 8.3.8		0.05
REFERENCE_ENTHALPY	Real	Section 9.4	kJ/kg	
REFERENCE_ENTHALPY_TEMPERATURE	Real	Section 9.4	K	
REFERENCE_RATE (:)	Real Array	Section 9.2	1/s	
REFERENCE_TEMPERATURE (:)	Real Array	Section 9.2	°C	
SPECIFIC_HEAT	Real	Section 8.3.2	kJ/(kg K)	0.
SPECIFIC_HEAT_RAMP	Character	Section 8.3.2		
SPEC_ID (: , :)	Char. Array	Section 9.2		
SURFACE_OXIDATION_MODEL	Logical	Section 17.1		F
X_O2_PYRO	Logical	Section 9.4		

23.17 MESH (Mesh Parameters)

Table 23.17: For more information see Section 6.3.

MESH (Mesh Parameters)				
Keyword	Type	Description	Units	Default
BNDF_MESH	Logical	Section 22.5		T
CHECK_MESH_ALIGNMENT	Logical	Section 6.3.4		F
COLOR	Character	Section 6.3.3		'BLACK'
CYLINDRICAL	Logical	Section 6.3.2		F
ID	Character	Section 6.3.1		
IJK	Integer Triplet	Section 6.3.1		10,10,10
MPI_PROCESS	Integer	Section 6.3.3		
MULT_ID	Character	Section 7.6		
RGB	Integer Triplet	Section 6.3.3		0,0,0
TRNX_ID	Character	Section 6.3.5		
TRNY_ID	Character	Section 6.3.5		
TRNZ_ID	Character	Section 6.3.5		
XB (6)	Real Sextuplet	Section 6.3.1	m	0,1,0,1,0,1

23.18 MISC (Miscellaneous Parameters)

Table 23.18: For more information see Section 19.

MISC (Miscellaneous Parameters)				
Keyword	Type	Description	Units	Default
AEROSOL_AL2O3	Logical	Section 14.3		F
AEROSOL_SCRUBBING	Logical	Section 13.6		F
AGGLOMERATION	Logical	Section 13.5		T
ALIGNMENT_TOLERANCE	Real	Section 6.3.4		0.001
BNDF_DEFAULT	Logical	Section 22.5		T
C_DEARDORFF	Real	Section 19.2		0.1
CFL_MAX	Real	Section 19.3.1		1.0
CFL_MIN	Real	Section 19.3.1		0.8
CFL_VELOCITY_NORM	Integer	Section 19.3.1		
CHECK_FO	Logical	Section 8.3.8		F
CHECK_HT	Logical	Section 19.3.4		F
CHECK_VN	Logical	Section 19.3.2		T
CNF_CUTOFF	Real	Section 15.3.3		0.005
CONSTANT_SPECIFIC_HEAT_RATIO	Logical	Section 12.1.3		F
C_SMAGORINSKY	Real	Section 19.2		0.20
C_VREMAN	Real	Section 19.2		0.07
C_WALE	Real	Section 19.2		0.60
DEPOSITION	Logical	Section 13.4		T
EXTERNAL_FILENAME	Character	Section 18.8		
FIXED_LES_FILTER_WIDTH	Real	Section 19.2	m	
FLUX_LIMITER	Integer	Section 19.4		2
FREEZE_VELOCITY	Logical	Section 20.4.2		F
GAMMA	Real	Section 12.1.2		1.4
GRAVITATIONAL_DEPOSITION	Logical	Section 13.4		T
GRAVITATIONAL_SETTLING	Logical	Section 13.4		T
GVEC (3)	Real Array	Section 20.7	m/s ²	0,0,-9.81
H_F_REFERENCE_TEMPERATURE	Real	Section 22.10.25	°C	25.
HUMIDITY	Real	Section 12.1.1	%	40.
HVAC_LOCAL_PRESSURE	Logical	Section 10.2		T
HVAC_MASS_TRANSPORT_CELL_L	Real	Section 10.2.8	m	
HVAC_PRES_RELAX	Real	Section 10.2		1.0
HVAC_QFAN	Logical	Section 10.2.4		F
IBLANK_SMV	Logical	Section 22.4		T
I_MAX_TEMP	Integer	Section 19.5.1	K	5000
LES_FILTER_TYPE	Character	Section 19.2		'MEAN'
LEVEL_SET_ELLIPSE	Logical	Section 17.5		T
LEVEL_SET_MODE	Integer	Section 17.5		0
MAXIMUM_VISIBILITY	Real	Section 22.10.5	m	30
MAX_LEAK_PATHS	Integer	Section 10.3.2		200
MAX_RAMPS	Integer	Section 11		100

Table 23.18: Continued

MISC (Miscellaneous Parameters)				
Keyword	Type	Description	Units	Default
MINIMUM_ZONE_VOLUME	Real	Section 10.3.4	m ³	0
MPI_TIMEOUT	Real	Section 4.2	s	600.
NEAR_WALL_PARTICLE_INTERPOLATION	Logical	Section 15.4.2		F
NEIGHBOR_SEPARATION_DISTANCE	Real	Section 8.4	m	0.
NOISE	Logical	Section 20.4.1		T
NOISE_VELOCITY	Real	Section 20.4.1	m/s	0.005
NO_PRESSURE_ZONES	Logical	Section 10.3.1		F
NORTH_BEARING	Real	Section 17.5.5	deg.	
NUCLEATION_SITES	Real	Section 13.7	#/m ³	1×10^7
ORIGIN_LAT	Real	Section 17.5.5	deg.	
ORIGIN_LON	Real	Section 17.5.5	deg.	
OVERWRITE	Logical	Section 5.5		T
PARTICLE_CFL	Logical	Section 19.3.3		F
PARTICLE_CFL_MAX	Real	Section 19.3.3		1.0
PARTICLE_CFL_MIN	Real	Section 19.3.3		0.8
POROUS_FLOOR	Logical	Section 15.6		T
PR	Real	Section 19.2		0.5
P_INF	Real	Section 20.2	Pa	101325
RAMP_GX	Character	Section 20.7		
RAMP_GY	Character	Section 20.7		
RAMP_GZ	Character	Section 20.7		
RAMP_UX	Character	Section 20.4.3		
RAMP_UY	Character	Section 20.4.3		
RAMP_UZ	Character	Section 20.4.3		
RAMP_VX	Character	Section 20.4.3		
RAMP_VY	Character	Section 20.4.3		
RAMP_VZ	Character	Section 20.4.3		
RAMP_WX	Character	Section 20.4.3		
RAMP_WY	Character	Section 20.4.3		
RAMP_WZ	Character	Section 20.4.3		
RESTART	Logical	Section 5.6		F
RESTART_CHID	Character	Section 5.6		CHID
RND_SEED	Integer	Section 5.6		
SC	Real	Section 20.4.1		0.5
SIMULATION_MODE	Character	Section 19.1		'VLES'
SMOKE_ALBEDO	Real	Reference [2]		0.3
SOLID_PHASE_ONLY	Logical	Section 9.3		F
SOOT_DENSITY	Real	Section 13.4		1800
SOOT_OXIDATION	Logical	Section 13.4		F
TAU_DEFAULT	Real	Section 11.1	s	1.
TERRAIN_IMAGE	Character	Section 22.10.20		
TEXTURE_ORIGIN(3)	Real Triplet	Section 7.5.2	m	(0.,0.,0.)

Table 23.18: Continued

MISC (Miscellaneous Parameters)				
Keyword	Type	Description	Units	Default
THERMOPHORETIC_DEPOSITION	Logical	Section 13.4		T
THERMOPHORETIC_SETTLING	Logical	Section 13.4		T
THICKEN_OBSTRUCTIONS	Logical	Section 7.2.1		F
TPMA	Real	Section 20.2	°C	20.
TURBULENCE_MODEL	Character	Section 19.2		'DEARDORFF'
TURBULENT_DEPOSITION	Logical	Section 13.4		T
UNFREEZE_TIME	Real	Section 20.6		
VERBOSE	Logical	Section 6.3.3		
VISIBILITY_FACTOR	Real	Section 22.10.5		3
VN_MAX	Real	Section 19.3.2		1.0
VN_MIN	Real	Section 19.3.2		0.8
Y_CO2_INFTY	Real	Section 13.1.1	kg/kg	
Y_O2_INFTY	Real	Section 13.1.1	kg/kg	

23.19 MOVE (Coordinate Transformation Parameters)

Table 23.19: For more information see Section 11.4.

MOVE (Coordinate Transformation Parameters)				
Keyword	Type	Description	Units	Default
AXIS (3)	Real Array	Axis of rotation		(0.,0.,1.)
SCALE	Real	Scaling in all directions		1.
SCALEX	Real	Scaling in the unrotated x direction		1.
SCALEY	Real	Scaling in the unrotated y direction		1.
SCALEZ	Real	Scaling in the unrotated z direction		1.
DX	Real	Translation in the x direction	m	0.
DY	Real	Translation in the y direction	m	0.
DZ	Real	Translation in the z direction	m	0.
ID	Character	Identification tag		
ROTATION_ANGLE	Real	Angle of rotation about AXIS	deg.	0.
T34	Real Array	3×4 Transformation Matrix		0.
X0	Real	x origin	m	0.
Y0	Real	y origin	m	0.
Z0	Real	z origin	m	0.

23.20 MULT (Multiplier Function Parameters)

Table 23.20: For more information see Section 7.6.

MULT (Multiplier Function Parameters)				
Keyword	Type	Description	Units	Default
DX	Real	Spacing in the x direction	m	0.
DXB (6)	Real Array	Spacing for all 6 coordinates	m	0.
DX0	Real	Translation in the x direction	m	0.
DY	Real	Spacing in the y direction	m	0.
DY0	Real	Translation in the y direction	m	0.
DZ	Real	Spacing in the z direction	m	0.
DZ0	Real	Translation in the z direction	m	0.
ID	Character	Identification tag		
I_LOWER	Integer	Lower array bound, x direction		0
I_LOWER_SKIP	Integer	Lower array bound begin skip, x direction		
I_UPPER	Integer	Upper array bound, x direction		0
I_UPPER_SKIP	Integer	Upper array bound end skip, x direction		
J_LOWER	Integer	Lower array bound, y direction		0
J_LOWER_SKIP	Integer	Lower array bound begin skip, y direction		
J_UPPER	Integer	Upper array bound, y direction		0
J_UPPER_SKIP	Integer	Upper array bound end skip, y direction		
K_LOWER	Integer	Lower array bound, z direction		0
K_LOWER_SKIP	Integer	Lower array bound begin skip, z direction		
K_UPPER	Integer	Upper array bound, z direction		0
K_UPPER_SKIP	Integer	Upper array bound end skip, z direction		
N_LOWER	Integer	Lower sequence bound		0
N_LOWER_SKIP	Integer	Lower sequence bound begin skip		
N_UPPER	Integer	Upper sequence bound		0
N_UPPER_SKIP	Integer	Upper sequence bound end skip		

23.21 OBST (Obstruction Parameters)

Table 23.21: For more information see Section 7.2.

OBST (Obstruction Parameters)				
Keyword	Type	Description	Units	Default
ALLOW_VENT	Logical	Section 7.2.1		T
BNDF_FACE (-3:3)	Logical Array	Section 22.5		T
BNDF_OBST	Logical	Section 22.5		T
BULK_DENSITY	Real	Section 9.2.9	kg/m ³	
CELL_SIZE	Real	Section 8.4.1	m	
CELL_SIZE_FACTOR	Real	Section 8.4.1		

Table 23.21: Continued

OBST (Obstruction Parameters)				
Keyword	Type	Description	Units	Default
COLOR	Character	Section 7.2.1		
CTRL_ID	Character	Section 18.4.2		
DEVC_ID	Character	Section 18.4.2		
HEIGHT	Real	Section 7.6.2	m	
ID	Character	Section 7.2.1		
INTERNAL_HEAT_SOURCE	Real	Section 8.4.1	kW/m ³	0.
LENGTH	Real	Section 7.6.2	m	
MATL_ID (:)	Char. Array	Section 8.4.5		
MATL_MASS_FRACTION (:)	Real Array	Section 8.4.5	kg/kg	
MULT_ID	Character	Section 7.6		
N_LAYER_CELLS_MAX	Integer	Section 8.4.1		
ORIENTATION (3)	Real Array	Section 7.6.2	m	(0.,0.,1.)
OUTLINE	Logical	Section 7.2.1		F
OVERLAY	Logical	Section 7.2.4		T
PERMIT_HOLE	Logical	Section 7.2.8		T
RADIUS	Real	Section 7.6.2	m	
RAMP_IHS	Character	Section 8.4.1		
REMOVABLE	Logical	Section 7.2.8		T
RGB (3)	Integer Array	Section 7.2.1		
SHAPE	Character	Section 7.6.2		
STRETCH_FACTOR	Real	Section 8.4.1		
SURF_ID	Character	Section 7.2.1		
SURF_ID_INTERIOR	Character	Section 9.2.9		
SURF_ID6 (6)	Char. Array	Section 7.2.1		
SURF_IDS (3)	Char. Array	Section 7.2.1		
TEXTURE_ORIGIN (3)	Real Array	Section 7.5.2	m	(0.,0.,0.)
THETA	Real	Section 7.6.2	deg.	
THICKEN	Logical	Section 7.2.1		F
TRANSPARENCY	Real	Section 7.2.1		1
WIDTH	Real	Section 7.6.2	m	
XB (6)	Real Array	Section 7.2.1	m	
XYZ (3)	Real Array	Section 7.6.2	m	(0.,0.,0.)

23.22 PART (Lagrangian Particles/Droplets)

Table 23.22: For more information see Chapter 15.

PART (Lagrangian Particles/Droplets)				
Keyword	Type	Description	Units	Default
ADHERE_TO_SOLID	Integer	Section 15.4.5		0
AGE	Real	Section 15.5	s	1×10^5
BREAKUP	Logical	Section 15.3.6		F
BREAKUP_CNF_RAMP_ID	Character	Section 15.3.6		
BREAKUP_DISTRIBUTION	Character	Section 15.3.6		'ROSIN...'
BREAKUP_GAMMA_D	Real	Section 15.3.6		2.4
BREAKUP_RATIO	Real	Section 15.3.6		3/7
BREAKUP_SIGMA_D	Real	Section 15.3.6		
CHECK_DISTRIBUTION	Logical	Section 15.3.3		F
CNF_RAMP_ID	Character	Section 15.3.3		
COLOR	Character	Section 22.9		'BLACK'
COMPLEX_REFRACTIVE_INDEX	Real	Section 15.3.2		0.01
CTRL_ID	Character	Section 15.5.1		
DENSE_VOLUME_FRACTION	Real	Section 15.3.4		1×10^{-5}
DEVC_ID	Character	Section 15.5.1		
DIAMETER	Real	Section 15.3.3	μm	
DISTRIBUTION	Character	Section 15.3.3		'ROSIN...'
DRAG_COEFFICIENT(3)	Real Array	Section 15.4.2		
DRAG_LAW	Character	Section 15.4.2		'SPHERE'
EVAP_MODEL	Character	Section 15.3		B-number
EMBER_DENSITY_THRESHOLD	Real	Section 17.2.4		
EMBER_PARTICLE	Logical	Section 17.2.4		F
EMBER_VELOCITY_THRESHOLD	Real	Section 17.2.4		
FREE_AREA_FRACTION	Real	Section 15.4.9		
GAMMA_D	Real	Section 15.3.3		2.4
HEAT_OF_COMBUSTION	Real	Section 15.3.1	kJ/kg	
HEAT_TRANSFER_COEFFICIENT_GAS	Real	Section 15.3.1	W/m ² /K	
HEAT_TRANSFER_COEFFICIENT_SOLID	Real	Section 15.7.1	W/m ² /K	
HORIZONTAL_VELOCITY	Real	Section 15.7.1	m/s	0.2
ID	Character	Section 15.1		
INITIAL_TEMPERATURE	Real	Section 15.3.1	°C	TMPA
KILL_DIAMETER	Real	Section 15.3.3	μm	
MASSLESS	Logical	Section 15.2		F
MASS_TRANSFER_COEFFICIENT	Real	Section 15.3.1	m/s	
MAXIMUM_DIAMETER	Real	Section 15.3.3	μm	1000000
MINIMUM_DIAMETER	Real	Section 15.3.3	μm	
MONODISPERSE	Logical	Section 15.3.3		F
N_STRATA	Integer	Section 15.3.3		6
NEW_PARTICLE_INCREMENT	Integer	Section 15.5		1000
ORIENTATION(1:3,:)	Real Array	Section 15.4		

Table 23.22: Continued

PART (Lagrangian Particles/Droplets)				
Keyword	Type	Description	Units	Default
PERMEABILITY (3)	Real Array	Section 15.4.8		
POROUS_VOLUME_FRACTION	Real	Section 15.4.8		
PRIMARY_BREAKUP_LENGTH	Real	Section 15.3.5	m	
PRIMARY_BREAKUP_DRAG_REDUCTION_FACTOR	Real	Section 15.3.5		1.0
PROP_ID	Character	Section 15.1		
QUANTITIES (10)	Character	Section 22.9		
QUANTITIES_SPEC_ID (10)	Character	Section 22.9		
RADIATIVE_PROPERTY_TABLE	Real	Section 15.3.2		
REAL_REFRACTIVE_INDEX	Real	Section 15.3.2		1.33
RGB (3)	Integers	Section 22.9		
RUNNING_AVERAGE_FACTOR	Real	Section 15.3.2		0.5
RUNNING_AVERAGE_FACTOR_WALL	Real	Section 15.3.2		0.5
SAMPLING_FACTOR	Integer	Section 22.9		1
SHAPE_FACTOR	Real	Section 17.2		0.25
SIGMA_D	Real	Section 15.3.3		
SPEC_ID	Character	Section 15.3.1		
STATIC	Logical	Section 15.4		F
SURFACE_DIAMETER	Real	Section 15.7.1	μm	
SURFACE_TENSION	Real	Section 15.3.6	N/m	7.28×10^{-2}
SURF_ID	Character	Section 15.4		
TARGET_ONLY	Logical	Section 15.4.11		F
TRACK_EMBERS	Logical	Section 17.2.4		T
TURBULENT_DISPERSION	Logical	Section 15.2		F
VERTICAL_VELOCITY	Real	Section 15.7.1	m/s	0.5

23.23 PRES (Pressure Solver Parameters)

Table 23.23: For more information see Section 21.

PRES (Pressure Solver Parameters)				
Keyword	Type	Description	Units	Default
BAROCLINIC	Logical	Section 21.2		T
CHECK_POISSON	Logical	Section 21.1		F
FISHPAK_BC (3)	Integer Array	Section 7.4.2		
ITERATION_SUSPEND_FACTOR	Real	Section 21.1	s	0.95
MAX_PRESSURE_ITERATIONS	Integer	Section 21.1		10
PRESSURE_RELAX_TIME	Real	Section 10.3.3	s	1.
PRESSURE_TOLERANCE	Real	Section 21.1	s^{-2}	
RELAXATION_FACTOR	Real	Section 10.3.3		1.

Table 23.23: Continued

PRES (Pressure Solver Parameters)				
Keyword	Type	Description	Units	Default
SOLVER	Character	Section 21.1.1		'FFT'
SUSPEND_PRESSURE_ITERATIONS	Logical	Section 21.1		T
TUNNEL_PRECONDITIONER	Logical	Section 21.3		F
VELOCITY_TOLERANCE	Real	Section 21.1	m/s	

23.24 PROF (Wall Profile Parameters)

Table 23.24: For more information see Section 22.3.

PROF (Wall Profile Parameters)				
Keyword	Type	Description	Units	Default
CELL_CENTERED	Logical	Section 22.3		F
FORMAT_INDEX	Integer	Section 22.3		1
ID	Character	Section 22.3		
INIT_ID	Character	Section 22.3		
IOR	Real	Section 22.3		
LP_TAG	Integer	Section 22.3		1
MATL_ID	Character	Section 22.3		
PART_ID	Character	Section 22.3		
QUANTITY	Character	Section 22.3		
XYZ	Real Triplet	Section 22.3	m	

23.25 PROP (Device Properties)

Table 23.25: For more information see Section 18.3.

PROP (Device Properties)				
Keyword	Type	Description	Units	Default
ACTIVATION_OBSCURATION	Real	Section 18.3.5	%/m	3.24
ACTIVATION_TEMPERATURE	Real	Section 18.3.1	°C	74.
ALPHA_C	Real	Section 18.3.5		1.8
ALPHA_E	Real	Section 18.3.5		0.
BETA_C	Real	Section 18.3.5		1.
BETA_E	Real	Section 18.3.5		1.
CHARACTERISTIC_VELOCITY	Real	Section 22.10.21	m/s	1.
C_FACTOR	Real	Section 18.3.1	(m/s) ^{1/2}	0.

Table 23.25: Continued

PROP (Device Properties)				
Keyword	Type	Description	Units	Default
DENSITY	Real	Section 22.10.8	kg/m ³	8908.
DIAMETER	Real	Section 22.10.8	m	0.001
EMISSIVITY	Real	Section 22.10.8		0.85
FED_ACTIVITY	Integer	Section 22.10.18		2
FLOW_RAMP	Character	Section 18.3.1		
FLOW_RATE	Real	Section 18.3.1	L/min	
FLOW_TAU	Real	Section 18.3.1		0.
GAUGE_EMISSIVITY	Real	Section 22.10.12		1.
GAUGE_TEMPERATURE	Real	Section 22.10.12	°C	TMPA
HEAT_TRANSFER_COEFFICIENT	Real	Section 22.10.8	W/(m ² K)	
HISTOGRAM	Logical	Section 22.10.19		F
HISTOGRAM_CUMULATIVE	Logical	Section 22.10.19		F
HISTOGRAM_LIMITS(2)	Real Array	Section 22.10.19		
HISTOGRAM_NBINS	Integer	Section 22.10.19		10
HISTOGRAM_NORMALIZE	Logical	Section 22.10.19		T
ID	Character	Section 18.3		
INITIAL_TEMPERATURE	Real	Section 18.3.1	°C	TMPA
K_FACTOR	Real	Section 18.3.1	L/(min bar ^{1/2})	1.
LENGTH	Real	Section 18.3.5	m	1.8
MASS_FLOW_RATE	Real	Section 18.3.1	kg/s	
OFFSET	Real	Section 18.3.1	m	0.05
OPERATING_PRESSURE	Real	Section 18.3.1	bar	1.
ORIFICE_DIAMETER	Real	Section 18.3.1	m	0.
PARTICLES_PER_SECOND	Integer	Section 18.3.1		5000
PARTICLE_VELOCITY	Real	Section 18.3.1	m/s	0.
PART_ID	Character	Section 18.3.1		
PDPA_END	Real	Section 22.10.16	s	T_END
PDPA_INTEGRATE	Logical	Section 22.10.16		T
PDPA_M	Integer	Section 22.10.16		0
PDPA_N	Integer	Section 22.10.16		0
PDPA_NORMALIZE	Logical	Section 22.10.16		T
PDPA_RADIUS	Real	Section 22.10.16	m	0.
PDPA_START	Real	Section 22.10.16	s	0.
PRESSURE_RAMP	Character	Section 18.3.1		
P0	Real	Section 18.3.3	m/s	0.
PX(3)	Real	Section 18.3.3	m/s	0.
PXX(3,3)	Real	Section 18.3.3	m/s	0.
QUANTITY	Character	Section 18.3.1		
RTI	Real	Section 18.3.1	√ms	100.
SMOKEVIEW_ID(:)	Char. Array	Section 18.7.1		
SMOKEVIEW_PARAMETERS(:)	Char. Array	Section 18.7.2		
SPARK	Logical	Section 13.1.7		F

Table 23.25: Continued

PROP (Device Properties)				
Keyword	Type	Description	Units	Default
SPEC_ID	Character	Section 18.3.5		
SPECIFIC_HEAT	Real	Section 22.10.8	kJ/(kg K)	0.44
SPRAY_ANGLE (2, 2)	Real	Section 18.3.1	degrees	60., 75.
SPRAY_PATTERN_BETA	Real	Section 18.3.1	degrees	5.
SPRAY_PATTERN_MU	Real	Section 18.3.1	degrees	0.
SPRAY_PATTERN_SHAPE	Character	Section 18.3.1		'GAUSSIAN'
SPRAY_PATTERN_TABLE	Character	Section 18.3.1		
TIME_CONSTANT	Real	Section 22.10.8	s	
VELOCITY_COMPONENT	Integer	Section 18.3.3		
VIEW_ANGLE	Real	Section 22.10.12	degrees	180.

23.26 RADF (Radiation Output File Parameters)

Table 23.26: For more information see Section 22.10.14.

RADF (Radiation Output File Parameters)				
Keyword	Type	Description	Units	Default
I_STEP	Integer	Section 22.10.14		1
J_STEP	Integer	Section 22.10.14		1
K_STEP	Integer	Section 22.10.14		1
XB	Real Sextuplet	Section 22.10.14	m	

23.27 RAD I (Radiation Parameters)

Table 23.27: For more information see Section 14.1.

RAD I (Radiation Parameters)				
Keyword	Type	Description	Units	Default
ANGLE_INCREMENT	Integer	Section 14.3		5
BAND_LIMITS	Real Array	Section 14.3.2	μm	
C_MAX	Real	Section 14.1		100
C_MIN	Real	Section 14.1		0.1
INITIAL_RADIATION_ITERATIONS	Integer	Section 14.2		3
KAPPA0	Real	Section 14.3	1/m	0
MIE_MINIMUM_DIAMETER	Real	Section 14.4	μm	0.5
MIE_MAXIMUM_DIAMETER	Real	Section 14.4	μm	$1.5 \times D$

Table 23.27: Continued

RADI (Radiation Parameters)				
Keyword	Type	Description	Units	Default
MIE_NDG	Integer	Section 14.4		50
NMIEANG	Integer	Section 14.4		15
NUMBER_RADIATION_ANGLES	Integer	Section 14.2		100
OPTICALLY_THIN	Logical	Section 14.1		F
PATH_LENGTH	Real	Section 14.3.1	m	0.1
QR_CLIP	Real	Section 14.1	kW/m ³	1
RADIATION	Logical	Section 14.1.1		T
RADIATION_ITERATIONS	Integer	Section 14.2		1
RADTMP	Real	Section 14.4	°C	900
TIME_STEP_INCREMENT	Integer	Section 14.2		3
WIDE_BAND_MODEL	Logical	Section 14.3.2		F
WSGG_MODEL	Logical	Section 14.3.3		F

23.28 RAMP (Ramp Function Parameters)

Table 23.28: For more information see Chapter 11.

RAMP (Ramp Function Parameters)				
Keyword	Type	Description	Units	Default
CTRL_ID	Character	Section 18.6.1		
CTRL_ID_DEP	Character	Section 18.6.2		
DEVC_ID	Character	Section 18.6.1		
DEVC_ID_DEP	Character	Section 18.6.2		
EXTERNAL_FILE	Logical	Section 18.8		F
F	Real	Chapter 11		
ID	Character	Chapter 11		
INITIAL_VALUE	Real	Section 18.8		
NUMBER_INTERPOLATION_POINTS	Integer	Chapter 11		5000
T	Real	Chapter 11	s (or °C)	
X	Real	Section 20.7	m	
Z	Real	Section 16.1	m	

23.29 REAC (Reaction Parameters)

Table 23.29: For more information see Chapter 13.

REAC (Reaction Parameters)				
Keyword	Type	Description	Units	Default
A	Real	Section 13.3		
A_LOW_PR	Real	Section 13.3.5		
A_TROE	Real	Section 13.3.5		
AIT_EXCLUSION_ZONE(6, :)	Real Array	Section 13.1.7	m	
AIT_EXCLUSION_ZONE_CTRL_ID(:)	Char. Array	Section 13.1.7		'null'
AIT_EXCLUSION_ZONE_DEVC_ID(:)	Char. Array	Section 13.1.7		'null'
AIT_EXCLUSION_ZONE_TEMPERATURE(:)	Real Array	Section 13.1.8	°C	
AUTO_IGNITION_TEMPERATURE	Real	Section 13.1.7	°C	-273 °C
CHECK_ATOM_BALANCE	Logical	Section 13.2		T
CO_YIELD	Real	Section 13.1.1	kg/kg	0
CRITICAL_FLAME_TEMPERATURE	Real	Section 13.1.6	°C	1427
E	Real	Section 13.3	J/mol	
E_LOW_PR	Real	Section 13.3.5	J/mol	
EPUMO2	Real	Section 13.1.2	kJ/kg	13100
EQUATION	Character	Section 13.2.5		
FUEL	Character	Section 13.1.1		
FUEL_C_TO_CO_FRACTION	Real	Section 13.1.3		2/3
FUEL_H_TO_H2_FRACTION	Real	Section 13.1.3		0
FUEL_N_TO_HCN_FRACTION	Real	Section 13.1.3		1/5
FUEL_RADCAL_ID	Character	Section 13.1.1		
HCN_YIELD	Real	Section 13.1.1	kg/kg	0
HEAT_OF_COMBUSTION	Real	Section 13.1.2	kJ/kg	
HOC_COMPLETE	Real	Section 13.1.4	kJ/kg	
ID	Character	Section 13.1.1		
IDEAL	Logical	Section 13.1.1		F
LOWER_OXYGEN_LIMIT	Real	Section 13.1.6	mol/mol	
N_S(:)	Real Array	Section 13.3		
N_SIMPLE_CHEMISTRY_REACTIONS	Integer	Section 13.1.3		1
N_T	Real	Section 13.3		
NU(:)	Real Array	Section 13.3		
PRIORITY	Integer	Section 13.2.3		1
RADIATIVE_FRACTION	Real	Section 14.1		
RAMP_CHI_R	Character	Section 14.1		
REAC_ATOM_ERROR	Real	Section 13.2	atoms	1.E-4
REAC_MASS_ERROR	Real	Section 13.2	kg/kg	1.E-4
REACTYPE	Character	Section 13.3.5		'ARRHENIUS-TYPE'
REVERSE	Logical	Section 13.3.3		F
SOOT_YIELD	Real	Section 13.1.1	kg/kg	0.0
SPEC_ID_N_S(:)	Char. Array	Section 13.3		
SPEC_ID_NU(:)	Char. Array	Section 13.3		

Table 23.29: Continued

REAC (Reaction Parameters)				
Keyword	Type	Description	Units	Default
T1_TROE	Real	Section 13.3.5	K	
T2_TROE	Real	Section 13.3.5	K	
T3_TROE	Real	Section 13.3.5	K	
THIRD_BODY	Logical	Section 13.3		F
THIRD_EFF (:)	Real Array	Section 13.3.4		
THIRD_EFF_ID (:)	Char. Array	Section 13.3.4		

23.30 SLCF (Slice File Parameters)

Table 23.30: For more information see Section 22.4.

SLCF (Slice File Parameters)				
Keyword	Type	Description	Units	Default
AGL_SLICE	Real	Section 22.10.20	m	
CELL_CENTERED	Logical	Section 22.4		F
DB	Character	Section 22.4		
ID	Character	Section 22.4		
MAXIMUM_VALUE	Real	Reference [2]		
MESH_NUMBER	Integer	Section 22.4		
MINIMUM_VALUE	Real	Reference [2]		
PART_ID	Character	Section 22.12		
PBX, PBZ, PBZ	Real	Section 22.4	m	
PROP_ID	Character	Section 22.10.18		
QUANTITY	Character	Section 22.12		
QUANTITY2	Character	Section 22.12		
REAC_ID	Character	See QUANTITY=`HRRPUV REAC`		
SPEC_ID	Character	Section 22.12		
VECTOR	Logical	Section 22.4		F
VELO_INDEX	Integer	Section 22.10.24		0
XB (6)	Real Array	Section 22.4	m	

23.31 SM3D (Smoke3D Parameters)

Table 23.31: For more information see Section 22.8.

SM3D (Smoke3D Parameters)				
Keyword	Type	Description	Units	Default
QUANTITY	Character	Section 22.8		
SPEC_ID	Character	Section 22.8		

23.32 SPEC (Species Parameters)

Table 23.32: For more information see Section 12.

SPEC (Species Parameters)				
Keyword	Type	Description	Units	Default
AEROSOL	Logical	Section 13.4		F
ALT_ID	Character	Section 12.1.1		
BACKGROUND	Logical	Section 12		F
BETA_LIQUID	Real	Section 15.3.1	1/K	
C	Real	Section 12.1.2		
COMPLEX_REFRACTIVE_INDEX	Real	Section 13.7		0.01
CONDUCTIVITY	Real	Section 12.1.2	W/(m K)	
CONDUCTIVITY_LIQUID	Real	Section 15.3.1	W/(m K)	
CONDUCTIVITY_SOLID	Real	Section 13.4	W/(m K)	0.26
COPY_LUMPED	Logical	Section 12.2		F
DENSITY_LIQUID	Real	Section 15.3.1	kg/m ³	
DENSITY_SOLID	Real	Section 13.4	kg/m ³	1800.
DIFFUSIVITY	Real	Section 12.1.3	m ² /s	
ENTHALPY_OF_FORMATION	Real	Section 15.3.1	kJ/mol	
EPSILONKLJ	Real	Section 12.1.2		0
FIC_CONCENTRATION	Real	Section 22.10.18	ppm	0.
FLD_LETHAL_DOSE	Real	Section 22.10.18	ppm × min	0.
FORMULA	Character	Section 12.1.2		
H	Real	Section 12.1.2		
HEAT_OF_VAPORIZATION	Real	Section 15.3.1	kJ/kg	
H_V_REFERENCE_TEMPERATURE	Real	Section 15.3.1	°C	
ID	Character	Section 12.1.1		
LUMPED_COMPONENT_ONLY	Logical	Section 12.2		F
MASS_EXTINCTION_COEFFICIENT	Real	Section 18.3.5		0
MASS_FRACTION(:)	Real Array	Section 12.2		0
MASS_FRACTION_0	Real	Section 12.1.1		0
MASS_FRACTION_COND_0	Real	Section 13.7		0
MAX_DIAMETER	Real	Section 13.5	m	

Table 23.32: Continued

SPEC (Species Parameters)				
Keyword	Type	Description	Units	Default
MEAN_DIAMETER	Real	Section 13.4	m	
MELTING_TEMPERATURE	Real	Section 15.3.1	°C	
MIN_DIAMETER	Real	Section 13.5	m	
MW	Real	Section 12.1.2	g/mol	29.
N	Real	Section 12.1.2		
N_BINS	Integer	Section 13.5		
O	Real	Section 12.1.2		
ODE_ABS_ERROR	Real	Section 13.3.7		
ODE_REL_ERROR	Real	Section 13.3.7		
POLYNOMIAL	Character	Section 12.1.3		
POLYNOMIAL_COEFF (:)	Real Array	Section 12.1.3		
POLYNOMIAL_TEMP (:)	Real Array	Section 12.1.3	K	
PR_GAS	Real	Section 12.1.2		PR
PRIMITIVE	Logical	Section 12.1.2		
RADICAL_ID	Character	Section 12.1.5		
RAMP_CP	Character	Section 12.1.2		
RAMP_CP_L	Character	Section 15.3.1		
RAMP_D	Character	Section 12.1.2		
RAMP_G_F	Character	Section 12.1.2		
RAMP_K	Character	Section 12.1.2		
RAMP_MU	Character	Section 12.1.2		
REAL_REFRACTIVE_INDEX	Real	Section 13.7		1.33
REFERENCE_ENTHALPY	Real	Section 12.1.2	kJ/kg	
REFERENCE_TEMPERATURE	Real	Section 12.1.2	°C	25.
SIGMALJ	Real	Section 12.1.2		0
SPEC_ID (:)	Char. Array	Section 12.2		
SPECIFIC_HEAT	Real	Section 12.1.2	kJ/(kg K)	
SPECIFIC_HEAT_LIQUID	Real	Section 15.3.1	kJ/(kg K)	
THERMOPHORETIC_DIAMETER	Real	Section 13.4	m	0.03E-6
TURBULENT_SCHMIDT_NUMBER	Real	Section 12.1.2		0.5
VAPORIZATION_TEMPERATURE	Real	Section 15.3.1	°C	
VISCOSITY	Real	Section 12.1.2	kg/(m s)	
VISCOSITY_LIQUID	Real	Section 15.3.1	kg/(m s)	
VOLUME_FRACTION (:)	Real Array	Section 12.2		

23.33 SURF (Surface Properties)

Table 23.33: For more information see Section 7.1.

SURF (Surface Properties)				
Keyword	Type	Description	Units	Default
ADIABATIC	Logical	Section 8.2.3		F
ALLOW_SURFACE_PARTICLES	Logical	Section 15.7.1		T
ALLOW_UNDERSIDE_PARTICLES	Logical	Section 15.7.1		F
AREA_MULTIPLIER	Real	Section 9.1.2		1.0
BACKING	Character	Section 8.3.3		'EXPOSED'
BLOWING	Logical	Section 8.2.2		
BURN_AWAY	Logical	Section 9.2.9		F
BURN_DURATION	Real	Section 9.2.8	s	1000000
CELL_SIZE(:)	Real Array	Section 8.3.8	m	
CELL_SIZE_FACTOR(:)	Real Array	Section 8.3.8		1.0
COLOR	Character	Section 7.5		
CONVECTION_LENGTH_SCALE	Real	Section 8.2.2	m	1.
CONVECTIVE_HEAT_FLUX	Real	Section 8.2.2	kW/m ²	
CONVERT_VOLUME_TO_MASS	Logical	Section 10.1.6		F
DEFAULT	Logical	Section 7.1		F
DELTA_TMP_MAX	Real	Section 8.3.8	°C	10
DRAG_COEFFICIENT	Real	Section 17.3		2.8
DSC_CONVERSION_FACTOR	Real	Section 9.3.2		1
DT_INSERT	Real	Section 15.5.1	s	0.01
E_COEFFICIENT	Real	Section 15.7	m ² /(kg s)	
EMBER_GENERATION_HEIGHT(2)	Real	Section 17.5.3	m	
EMBER_IGNITION_POWER_MEAN	Real	Section 17.5.4	kW	
EMBER_IGNITION_POWER_SIGMA	Real	Section 17.5.4	kW	0.001
EMBER_TRACKING_RATIO	Real	Section 17.5.3		100
EMBER_YIELD	Real	Section 17.5.3	kg/kg	
EMISSIVITY	Real	Section 8.2.2		0.9
EMISSIVITY_BACK	Real	Section 8.3.3		
EXTERNAL_FLUX	Real	Section 9.3	kW/m ²	
EXTINCTION_TEMPERATURE	Real	Section 9.1.3	°C	-273.
FREE_SLIP	Logical	Section 10.1.7		F
GEOMETRY	Character	Section 8.3.7		'CARTESIAN'
HEAT_OF_VAPORIZATION	Real	Section 9.1.3	kJ/kg	
HEAT_TRANSFER_COEFFICIENT	Real	Section 8.2.2	W/(m ² K)	
HEAT_TRANSFER_COEFFICIENT_BACK	Real	Section 8.2.2	W/(m ² K)	
HEAT_TRANSFER_COEFFICIENT_SIGMA	Real	Section 8.2.2	m	
HEAT_TRANSFER_MODEL	Character	Section 8.2.2		
HORIZONTAL	Logical	Section 8.3.7		F
HRRPUA	Real	Section 9.1	kW/m ²	
HT3D	Logical	Section 8.4		F

Table 23.33: Continued

SURF (Surface Properties)				
Keyword	Type	Description	Units	Default
ID	Character	Section 7.1		
IGNITION_TEMPERATURE	Real	Section 9.1.3	°C	5000.
INERT_Q_REF	Logical	Section 9.1.4		F
INIT_IDS	Char. Array	Section 17.2.1		
INIT_PER_AREA	Real	Section 17.2.1	m ⁻²	
INNER_RADIUS	Real	Section 15.4.1	m	
INTERNAL_HEAT_SOURCE	Real Array	Section 8.3.6	kW/m ³	
LAYER_DIVIDE	Real	Section 9.2.2		N_LAYER/2
LEAK_PATH	Int. Pair	Section 10.3.2		
LEAK_PATH_ID	Character Pair	Section 10.3.2		
LENGTH	Real	Section 15.4.1	m	
MASS_FLUX(:)	Real Array	Section 10.1.6	kg/(m ² s)	
MASS_FLUX_TOTAL	Real	Section 10.1.2	kg/(m ² s)	
MASS_FLUX_VAR	Real	Section 10.1.9		
MASS_FRACTION(:)	Real Array	Section 10.1.6		
MASS_PER_VOLUME(:)	Real Array	Section 17.2	kg/m ³	
MASS_TRANSFER_COEFFICIENT	Real	Section 9.2.7	m/s	
MATL_ID(:, :)	Char. Array	Section 9.2		
MATL_MASS_FRACTION(:, :)	Real Array	Section 9.2		
MAXIMUM_SCALING_HEAT_FLUX	Real	Section 9.1.4	kW/m ²	
MCC_CONVERSION_FACTOR	Real	Section 9.3.2		1
MINIMUM_BURNOUT_TIME	Real	Section 17.3.1	s	1000000
MINIMUM_LAYER_THICKNESS	Real	Section 8.3.8	m	1.E-4
MINIMUM_SCALING_HEAT_FLUX	Real	Section 9.1.4	kW/m ²	0
MLRPUA	Real	Section 9.1	kg/(m ² s)	
MOISTURE_FRACTION(:)	Real Array	Section 17.2		0.
N_LAYER_CELLS_MAX(:)	Integer Array	Section 8.3.8		1000
NEAR_WALL_EDDY_VISCOSITY	Real	Section 19.2	m ² /s	
NEAR_WALL_TURBULENCE_MODEL	Character	Section 19.2		
NET_HEAT_FLUX	Real	Section 8.2.2	kW/m ²	
NO_SLIP	Logical	Section 10.1.7		F
NPPC	Integer	Section 15.5.1		1
NUSSELT_C0	Real	Section 8.2.2		
NUSSELT_C1	Real	Section 8.2.2		
NUSSELT_C2	Real	Section 8.2.2		
NUSSELT_M	Real	Section 8.2.2		
PARTICLE_EXTRACTION_VELOCITY	Real	Section 15.6	m/s	
PARTICLE_MASS_FLUX	Real	Section 15.5.1	kg/(m ² s)	
PARTICLE_SURFACE_DENSITY	Real	Section 15.5.1	kg/m ²	
PART_ID	Character	Section 15.5.1		
PLE	Real	Section 16.5		0.3
PROFILE	Character	Section 10.5		

Table 23.33: Continued

SURF (Surface Properties)				
Keyword	Type	Description	Units	Default
RADIUS	Real	Section 15.4.1	m	
RAMP_EF	Character	Section 11.1		
RAMP_HEAT_TRANSFER_COEFFICIENT	Real	Section 8.2.2		
RAMP_HEAT_TRANSFER_COEFFICIENT_BACK	Real	Section 8.2.2		
RAMP_IHS	Character	Section 8.3.6		
RAMP_MF (:)	Character	Section 11.1		
RAMP_PART	Character	Section 11.1		
RAMP_Q	Character	Section 11.1		
RAMP_T	Character	Section 11.1		
RAMP_T_I	Character	Section 8.3.4		
RAMP_TMP_BACK	Character	Section 8.3.9		
RAMP_TMP_GAS_BACK	Character	Section 8.3.3		
RAMP_TMP_GAS_FRONT	Character	Section 8.3.3		
RAMP_V	Character	Section 11.1		
RAMP_V_X	Character	Section 11.3		
RAMP_V_Y	Character	Section 11.3		
RAMP_V_Z	Character	Section 11.3		
REFERENCE_HEAT_FLUX	Real Array	Section 9.1.4	kW/m ²	
REFERENCE_HEAT_FLUX_TIME_INTERVAL	Real	Section 9.1.4	s	1.0
REFERENCE_THICKNESS	Real Array	Section 9.1.4	m	
REMESH_RATIO	Real	Section 8.3.8		0.15
RGB (3)	Integer Array	Section 7.5		255,204,102
ROUGHNESS	Real	Section 10.1.7	m	0.
SHAPE_FACTOR	Real	Section 17.3		0.25
SPEC_ID	Character	Section 10.1.6		
SPREAD_RATE	Real	Section 9.1.1	m/s	
STRETCH_FACTOR (:)	Real Array	Section 8.3.8		2.
SURFACE_VOLUME_RATIO (:)	Real	Section 17.2	l/m	
TAU_EF	Real	Section 11.1	s	1.
TAU_MF (:)	Real Array	Section 11.1	s	1.
TAU_PART	Real	Section 11.1	s	1.
TAU_Q	Real	Section 11.1	s	1.
TAU_T	Real	Section 11.1	s	1.
TAU_V	Real	Section 11.1	s	1.
TEXTURE_HEIGHT	Real	Section 7.5.2	m	1.
TEXTURE_MAP	Character	Section 7.5.2		
TEXTURE_WIDTH	Real	Section 7.5.2	m	1.
TGA_ANALYSIS	Logical	Section 9.3.2		F
TGA_CONVERSION_FACTOR	Real	Section 9.3.2		1
TGA_DT	Real	Section 9.3.2	s	0.01
TGA_DUMP	Real	Section 9.3.2	K	1
TGA_FINAL_TEMPERATURE	Real	Section 9.3.2	°C	800.

Table 23.33: Continued

SURF (Surface Properties)				
Keyword	Type	Description	Units	Default
TGA_HEATING_RATE	Real	Section 9.3.2	°C/min	5.
THICKNESS (:)	Real Array	Section 8.1	m	
TIME_STEP_FACTOR	Real	Section 8.3.8		10.
TMP_BACK	Real	Section 8.3.9	°C	
TMP_FRONT	Real	Section 8.2.1	°C	20.
TMP_FRONT_INITIAL	Real	Section 8.2.3	°C	
TMP_GAS_FRONT	Real	Section 8.3.3	°C	
TMP_GAS_BACK	Real	Section 8.3.3	°C	
TMP_INNER	Real	Section 8.3.4	°C	20.
TRANSPARENCY	Real	Section 7.5		1.
VARIABLE_THICKNESS	Logical	Section 8.4.5		F
VEG_LSET_BETA	Real	Section 17.5		0.01
VEG_LSET_CHAR_FRACTION	Real	Section 17.5		0.2
VEG_LSET_FIREBASE_TIME	Real	Section 17.5	s	
VEG_LSET_FUEL_INDEX	Integer	Section 17.5		
VEG_LSET_HT	Real	Section 17.5	m	0.2
VEG_LSET_IGNITE_TIME	Real	Section 17.5	s	
VEG_LSET_M1	Real	Section 17.5		0.03
VEG_LSET_M10	Real	Section 17.5		0.04
VEG_LSET_M100	Real	Section 17.5		0.05
VEG_LSET_MLW	Real	Section 17.5		0.70
VEG_LSET_MLH	Real	Section 17.5		0.70
VEG_LSET_QCON	Real	Section 17.5	kW/m ²	0.
VEG_LSET_ROS_00	Real	Section 17.5	m/s	0.
VEG_LSET_ROS_BACK	Real	Section 17.5	m/s	0.
VEG_LSET_ROS_FLANK	Real	Section 17.5	m/s	0.
VEG_LSET_ROS_HEAD	Real	Section 17.5	m/s	0.
VEG_LSET_ROS_FIXED	Logical	Section 17.5		F
VEG_LSET_SIGMA	Real	Section 17.5	1/m	5000.
VEG_LSET_SURF_LOAD	Real	Section 17.5	kg/m ²	1.0
VEG_LSET_TAN2	Real	Section 17.5		
VEG_LSET_WIND_EXP	Real	Section 17.5		1.
VEL	Real	Section 10.1	m/s	
VEL_BULK	Real	Section 10.5	m/s	
VEL_GRAD	Real	Section 10.1.5	1/s	
VEL_PART	Real	Section 15.5.1	m/s	
VEL_T (2)	Real Array	Section 10.1.4	m/s	
VOLUME_FLOW	Real	Section 10.1	m ³ /s	
WIDTH	Real	Section 15.4.1	m	
XYZ (3)	Real Array	Section 9.1.1	m	
Z0	Real	Section 16.5	m	10.
Z_0	Real	Section 16.2.2	m	0.

23.34 TABL (Table Parameters)

Table 23.34: For more information see Section 18.3.1.

TABL (Table Parameters)				
Keyword	Type	Description	Units	Default
ID	Character	Section 18.3.1		
TABLE_DATA (9)	Real Array	Section 18.3.1		

23.35 TIME (Time Parameters)

Table 23.35: For more information see Section 6.2.

TIME (Time Parameters)				
Keyword	Type	Description	Units	Default
DT	Real	Section 6.2.2	s	
DT_END_FILL	Real	Section 6.2.2	s	1.0×10^{-6}
DT_END_MINIMUM	Real	Section 6.2.2	s	2.*EPSILON
DT_EXTERNAL	Real	Section 18.8	s	0
DT_EXTERNAL_HEARTBEAT	Real	Section 18.8	s	0
EXTERNAL_HEARTBEAT_FILENAME	Character	Section 18.8		
HEARTBEAT_FAIL	Logical	Section 18.8		T
LIMITING_DT_RATIO	Real	Section 4.2		0.0001
LOCK_TIME_STEP	Logical	Section 6.2.2		F
RAMP_TIME	Character	Section 6.2.2		
RESTRICT_TIME_STEP	Logical	Section 6.2.2		T
T_BEGIN	Real	Section 6.2.1	s	0.
T_END	Real	Section 6.2.1	s	1.
TIME_SHRINK_FACTOR	Real	Section 6.2.3		1.
WALL_INCREMENT	Integer	Section 8.3.8		2

23.36 TRNX, TRNY, TRNZ (MESH Transformations)

Table 23.36: For more information see Section [6.3.5](#).

TRNX, TRNY, TRNZ (MESH Transformations)				
Keyword	Type	Description	Units	Default
CC	Real	Section 6.3.5	m	
ID	Character	Section 6.3.5		
IDERIV	Integer	Section 6.3.5		
MESH_NUMBER	Integer	Section 6.3.5		
PC	Real	Section 6.3.5		

23.37 VENT (Vent Parameters)

Table 23.37: For more information see Section 7.4.

VENT (Vent Parameters)				
Keyword	Type	Description	Units	Default
AREA_ADJUST	Logical	Section 18.6.4		F
COLOR	Character	Section 7.5		
CTRL_ID	Character	Section 18.4.2		
DB	Character	Section 7.4.1		
DEVC_ID	Character	Section 18.4.2		
DYNAMIC_PRESSURE	Real	Section 10.4	Pa	0.
GEOM	Logical	Section 17.5		F
ID	Character	Section 7.4.1		
IOR	Integer	Section 7.4.4		
L_EDDY	Real	Section 10.1.8	m	0.
L_EDDY_IJ (3, 3)	Real Array	Section 10.1.8	m	0.
MB	Character	Section 7.4.1		
MULT_ID	Character	Section 7.6		
N_EDDY	Integer	Section 10.1.8		0
OBST_ID	Character	Section 18.4.2		
OUTLINE	Logical	Section 7.4.1		F
PBX, PBZ, PBZ	Real	Section 7.4.1		
PRESSURE_RAMP	Character	Section 10.4		
RADIUS	Real	Section 7.4.2	m	
REYNOLDS_STRESS (3, 3)	Real Array	Section 10.1.8	m ² /s ²	0.
RGB (3)	Integer Array	Section 7.5		
SPREAD_RATE	Real	Section 9.1.1	m/s	0.05
SURF_ID	Character	Section 7.4.1		'INERT'
TEXTURE_ORIGIN (3)	Real Array	Section 7.5.2	m	(0.,0.,0.)
TMP_EXTERIOR	Real	Section 7.4.2	°C	
TMP_EXTERIOR_RAMP	Character	Section 7.4.2		
TRANSPARENCY	Real	Section 7.5		1.0
UVW (3)	Real Array	Section 10.2.7		
VEL_RMS	Real	Section 10.1.8	m/s	0.
XB (6)	Real Array	Section 7.4.1	m	
XYZ (3)	Real Array	Section 9.1.1	m	

23.38 WIND (Wind and Atmospheric Parameters)

Table 23.38: For more information see Section 16.2.

WIND (Wind and atmospheric parameters)				
Keyword	Type	Description	Units	Default
CORIOLIS_VECTOR(3)	Real	Section 16.3.2		0.
DIRECTION	Real	Section 16.2	degrees	270
FORCE_VECTOR(3)	Real	Section 16.3.1	Pa/m	0.
GEOSTROPHIC_WIND(2)	Real	Section 16.3.3	m/s	
GROUND_LEVEL	Real	Section 16.5	m	0.
L	Real	Section 16.2	m	0
LAPSE_RATE	Real	Section 16.5	°C/m	0
LATITUDE	Real	Section 16.3.2	degrees	
PRESSURE_GRADIENT_FORCE	Real	Section 16.3.1	Pa/m	
RAMP_DIRECTION_T	Character	Section 16.2		
RAMP_DIRECTION_Z	Character	Section 16.2		
RAMP_FVX_T	Character	Section 16.3.1		
RAMP_FVY_T	Character	Section 16.3.1		
RAMP_FVZ_T	Character	Section 16.3.1		
RAMP_PGF_T	Character	Section 16.3.1		
RAMP_SPEED_T	Character	Section 16.1		
RAMP_SPEED_Z	Character	Section 16.1		
RAMP_TMP0_Z	Character	Section 16.5		
SIGMA_THETA	Real	Section 16.1	degrees	
SPEED	Real	Section 16.2	m/s	0.
STRATIFICATION	Logical	Section 16.5		T
TAU_THETA	Real	Section 16.1	s	300
THETA_STAR	Real	Section 16.2	K	
TMP_REF	Real	Section 16.2	°C	TMPA
U_STAR	Real	Section 16.2	m/s	
U0,V0,W0	Reals	Section 16.2	m/s	0.
USE_ATMOSPHERIC_INTERPOLATION	Logical	FDS Tech Guide [3]		T
Z_0	Real	Section 16.2	m	0.03
Z_REF	Real	Section 16.2	m	10.

23.39 ZONE (Pressure Zone Parameters)

Table 23.39: For more information see Section 10.3.

ZONE (Pressure Zone Parameters)				
Keyword	Type	Description	Units	Default
DISCHARGE_COEFFICIENT (N)	Real	Section 10.3.2		1.
ID	Character	Section 10.3.1		
LEAK_AREA (N)	Real	Section 10.3.2	m ²	0
LEAK_PRESSURE_EXPONENT (N)	Real	Section 10.3.2		0.5
LEAK_REFERENCE_PRESSURE (N)	Real	Section 10.3.2	Pa	4
XYZ (3, :)	Real Array	Section 10.3.1	m	

Chapter 24

Error Codes

101	Problem with ... line.	Chapter 23
102	Input file ... not found in the current directory.	Section 3.2
103	CATF file ... not found.	Section 5.4
104	OVERWRITE=F and concatenated file ... exists.	Section 5.5
105	Input file ... has line with ... characters.	Section 5.4
106	CATF file ... has line with ... characters.	Section 5.4
107	Hidden carriage return character in line starting with ...	Section 5.1
108	No periods allowed in CHID.	Section 5.1
109	Remove the file ... from the current directory.	Section 5.6
110	All meshes must be CYLINDRICAL ...	Section 6.3.2
111	MULT_ID ... on MESH line ... not found.	Section 6.3.3
112	The number of MPI processes ... exceeds the number of meshes	Section 3.1.2
113	No MESH line(s) defined.	Section 6.3.1
114	MPI_PROCESS for MESH ... greater than total ...	Section 6.3.3
115	Number of meshes exceeds number of MPI processes ...	Section 3.1.2
116	IJK(2) must be 1 for all grids in 2D Calculation.	Section 6.3.2
117	MPI_PROCESS must be continuous and monotonically increasing.	Section 6.3.3
118	MESH 1 must be assigned to MPI_PROCESS 0.	Section 6.3.3
119	XB(:)=XB(:) on MESH ...	Section 6.3.1
120	XB(1)<0 with CYLINDRICAL on MESH ...	Section 6.3.2
121	J>1 with CYLINDRICAL on MESH ...	Section 6.3.2
122	Too many MPI processes have been assigned to this job.	Section 3.1.2
123	... timed out for MPI process ...	Section 4.2
124	Problem with grid transformation.	Section 6.3.5
125	Do not specify endpoints in linear grid transformation.	Section 6.3.5
126	x (y,z) transformation not monotonic, MESH ...	Section 6.3.5
127	TURBULENCE_MODEL ... is not appropriate for DNS.	Section 19.2
128	SIMULATION_MODE ... is not an option.	Section 19.1
129	TURBULENCE_MODEL ... is not recognized.	Section 19.2
130	FLUX_LIMITER ... is not recognized.	Section 19.4
131	GEOSTROPHIC_WIND requires Coriolis force ...	Section 16.3.3

132	SPEC N_BINS must be >=2.	Section 13.5
133	SPEC AEROSOL must be .TRUE. to use N_BINS.	Section 13.5
134	SPEC Do not specify MEAN_DIAMETER and N_BINS.	Section 13.5
135	SPEC MAX(MIN)_DIAMETER not set.	Section 13.5
136	SPEC MAX_DIAMETER <= MIN_DIAMETER.	Section 13.5
137	SPEC LUMPED_COMPONENT_ONLY must be .TRUE. to use N_BINS.	Section 13.5
138	SPEC Cannot set N_BINS for a condensable species.	Section 13.7
139	SPEC Condensable species cannot be LUMPED_SPECIES_ONLY.	Section 13.7
140	SPEC Condensable species cannot be a lumped species.	Section 13.7
141	Species ... is primitive but more than one SPEC_ID given.	Section 12.2
142	SPEC ... cannot specify both COPY_LUMPED and N_BINS.	Section 12.2
143	SPEC ... cannot specify both COPY_LUMPED and BACKGROUND.	Section 12.2
144	Only one BACKGROUND SPECies can be defined.	Section 12.1
145	Cannot define a LUMPED_COMPONENT_ONLY ...	Section 12.2
146	Cannot define a BACKGROUND species or redefine AIR ...	Section 12.2
147	Species ... needs a name (ID=...).	Section 12.1.1
148	SPEC ... cannot redefine AIR unless ...	Section 12.1.4
149	SPEC ... cannot specify both SPECIFIC_HEAT and RAMP_CP.	Section 12.1.3
150	SPEC ... cannot specify both SPECIFIC_HEAT_LIQUID ...	Section 12.1.3
151	SPEC ... cannot specify both CONDUCTIVITY and RAMP_K.	Section 12.1.3
152	SPEC ... cannot specify both DIFFUSIVITY and RAMP_D.	Section 12.1.3
153	SPEC ... cannot specify both VISCOSITY and RAMP_MU.	Section 12.1.3
154	SPEC ... cannot define both REFERENCE_ENTHALPY and ...	Section 12.1.3
155	Species ... has the same ID as species ...	Section 12.1.1
156	Cannot define MASS_FRACTION_0 for a LUMPED ...	Section 12.1.1
157	SPEC If one of SPECIFIC_HEAT_LIQUID ...	Section 15.3.1
158	SPEC No MEAN_DIAMETER given.	Section 13.4
159	SPEC ... cannot specify both MASS and VOLUME_FRACTION.	Section 12.2
160	SPEC ... cannot specify both FORMULA and C,H,O, or N.	Section 12.1.3
161	Simple chemistry FUEL ... not defined on REAC or SPEC.	Section 13.1.1
162	SPEC ... has the same names as a predefined lumped species.	Section 13.1.1
163	REAC FORMULA limited to C,H,O,N for simple chemistry.	Section 13.1.1
164	REAC Specify C and/or H for simple chemistry.	Section 13.1.1
165	Cannot use simple chemistry with SOOT_OXIDATION.	Section 13.4.2
166	SOOT_OXIDATION set without SOOT as a species.	Section 13.4.2
167	SOOT_OXIDATION set without SOOT defined as an AEROSOL species.	Section 13.4.2
168	SPEC Subspecies ... not found.	Section 12.2
169	SPEC Mass or volume fraction for subspecies ...	Section 12.2
170	SPEC Cannot have duplicate species in SPEC_ID.	Section 12.2
171	REAC, Not enough carbon for the CO_YIELD ...	Section 13.1.1
172	REAC, Not enough hydrogen for the SOOT_YIELD ...	Section 13.1.1
173	REAC, Not enough nitrogen for the HCN_YIELD.	Section 13.1.1
174	REAC, Not enough nitrogen for the FUEL_N_TO_HCN_FRACTION.	Section 13.1.1
175	REAC, Not enough carbon for the FUEL_C_TO_CO_FRACTION ...	Section 13.1.1
176	REAC, Not enough hydrogen for the FUEL_H_TO_H2_FRACTION ...	Section 13.1.1
180	SPEC ... does not have a SPECIFIC_HEAT_LIQUID ...	Section 15.3.1
181	SPEC ... does not have a MELTING_TEMPERATURE.	Section 15.3.1

182	SPEC ... does not have a VAPORIZATION_TEMPERATURE.	Section 15.3.1
183	SPEC ... MELTING_TEMPERATURE must be less than ...	Section 15.3.1
184	SPEC ... does not have a HEAT_OF_VAPORIZATION.	Section 15.3.1
185	SPEC ... does not have a DENSITY_LIQUID.	Section 15.3.1
187	SPEC ... is used for droplets and does not have liquid props.	Section 15.3.1
188	EXTINCTION_MODEL ... is not recognized.	Section 13.1.6
189	REAC ... cannot use both finite rate and simple chemistry.	Section 13.1.1
190	REAC ... requires a FUEL.	Section 13.1.1
191	FORMULA for SIMPLE_CHEMISTRY can only contain C,H,O, and N.	Section 13.1.1
192	Specify fuel using C and/or H when using simple chemistry.	Section 13.1.1
193	FUEL_C_TO_CO_FRACTION must be between 0 and 1.	Section 13.1.3
194	FUEL_H_TO_H2O_FRACTION must be between 0 and 1.	Section 13.1.3
195	FUEL_N_TO_HCN_FRACTION must be between 0 and 1.	Section 13.1.3
196	REAC ... SPEC_ID_NU and NU arrays or EQUATION must be defined.	Section 13.3
197	REAC ... THIRD_EFF values must be >= 0.	Section 13.3.4
198	REAC ... invalid EQUATION specified.	Section 13.2.5
199	REAC ... uses simple chemistry and has a duplicate fuel ...	Section 13.1.1
200	Fuel for simple chemistry has NU_O2<= 0 and requires air ...	Section 13.1.1
201	REAC ... Cannot set NUs if an EQUATION is specified.	Section 13.2.5
202	REAC ... Tracked species ... not found.	Section 13.3
204	REAC ... Primitive species ... not found.	Section 13.3
205	REAC ... Products not specified.	Section 13.3
206	REAC ... Reactants not specified.	Section 13.3
207	REAC ... Mass of products does not equal mass of reactants.	Section 13.2
208	REAC ... THIRD_EFF primitive species ... not found.	Section 13.3.4
209	Name of ODE_SOLVER is not recognized.	Section 13.3.7
210	REAC ... Missing an ENTHALPY_OF_FORMATION.	Section 15.3.1
212	REAC ... Reversible reaction species ... missing G_F.	Section 12.1.3
213	PART ... cannot have both a specified DIAMETER and a SURF_ID.	Section 15.4
214	PART ... requires a specified DIAMETER.	Section 15.3.3
215	Cannot have MASSLESS=.TRUE. with evaporating PARTICLES.	Section 15.2
216	PART ... needs a SURF_ID.	Section 15.4
217	PART SPEC_ID ... not found.	Section 15.3.1
218	PART ... Particles cannot evaporate to a lumped species.	Section 15.3
219	PART ... No DRAG_COEFFICIENT when using a DRAG_LAW.	Section 15.4.2
220	PART ... Specify exactly one ORIENTATION for a SCREEN.	Section 15.4.9
221	PART ... Specify FREE_AREA_FRACTION for a SCREEN.	Section 15.4.9
222	PART ... Specify all components for DRAG_COEFFICIENT and ...	Section 15.4.8
223	PART ... Unrecognized drag law.	Section 15.4.2
224	PART ... Invalid EVAP_MODEL.	Section 15.3
225	HEAT_TRANSFER_MODEL not appropriate for PART.	Section 8.2.2
231	PROP ... values for SPRAY_ANGLE cannot be the same.	Section 18.3.1
232	PROP ... HISTOGRAM needs HISTOGRAM_NBINS>2.	Section 22.10.19
233	PROP ... HISTOGRAM needs HISTOGRAM_LIMITS.	Section 22.10.19
234	PROP ... FED_ACTIVITY out of range: ...	Section 22.10.18
235	PROP ... VELOCITY_COMPONENT > 3: ...	Section 18.3.3

236	PROP ... VELOCITY_PATCH requires P0.	Section 18.3.3
237	PROP ... specify PARTICLE_VELOCITY with MASS_FLOW_RATE.	Section 18.3.1
238	PROP ... too few flow parameters.	Section 18.3.1
239	PROP SPEC_ID ... not found.	Section 18.3.5
240	PART_ID for PROP ... not found.	Section 18.3.1
241	PART_ID for PROP ... cannot refer to MASSLESS particles.	Section 18.3.1
242	Spray Pattern Table ... massflux <= 0 for line ...	Section 18.3.1
243	PROP ... VIEW_ANGLE must be between 0 and 180	Section 18.3.1
251	MATL ... REAC ... Set REFERENCE_TEMPERATURE or E, A.	Section 9.2.3
252	MATL ... HEAT_OF_REACTION should be greater than 0.	Section 9.2.6
253	MATL ... DENSITY=0.	Section 8.3.2
254	MATL ... CONDUCTIVITY=0.	Section 8.3.2
255	MATL ... SPECIFIC_HEAT=0.	Section 8.3.2
256	MATL ... SURFACE_OXIDATION_MODEL but no OXYGEN.	Section 17.1
257	Residue ... of ... is not defined.	Section 9.2
258	PARTicle ... corresponding to MATL ... cannot be MASSLESS.	Section 9.2
259	MATL ... PART_ID ... not defined.	Section 9.2
260	MATL ... PART_ID ... has a NU_PART<=0.	Section 9.2
261	Duplicate material name: ...	Section 8.3
262	MATL ... requires a SPEC_ID for yield ... of reaction ...	Section 9.2
263	MATL ... can only specify one SPEC_ID for a liquid.	Section 9.2.7
264	MATL ... SPEC_ID ... not a tracked.	Section 9.2
265	MATL ... Sum of NU inputs more than 1.	Section 9.2
266	REAC ... FUEL ... is not a predefined or tracked species.	Section 13.1.1
267	REAC ... has no consumed species.	Section 13.2
268	BACK tracked species ... not found.	Section 22.2.2
299	SURF ... has no solid or particle for TGA.	Section 9.3.2
300	N_LAYER_CELLS_MAX should be at least ... for ...	Section 8.3.8
301	SURF line must have an ID.	Section 7.1
302	SURF ID ... is used more than once.	Section 7.1
303	MOISTURE_FRACTION on SURF ... exceeds theoretical limit.	Section 17.2
304	SURF ... One layer only for TGA_ANALYSIS=T.	Section 9.3.2
305	SURF ... indicates a level set simulation ...	Section 17.5
306	SURF ... cannot have a specified flux and a MATL_ID.	Section 10.1.6
307	SURF ... must have a specified THICKNESS for Layer ...	Section 8.1
308	SURF ... needs a RADIUS or THICKNESS.	Section 8.3.7
309	SURF ... BACKING ... not recognized.	Section 8.3.3
310	SURF ... must specify TMP_FRONT for CONVERT_VOLUME_TO_MASS.	Section 10.1.6
311	SURF ... cannot use velocity RAMP with CONVERT_VOLUME_TO_MASS.	Section 10.1.6
312	SURF ... GEOMETRY not recognized.	Section 8.3.7
313	SURF ... HEAT_TRANSFER_MODEL not recognized.	Section 8.2.2
314	SURF ... must have a REAC line when using HRRPUA or MLRPUA.	Section 9.1
315	SURF ... should have only one LEAK_PATH and LEAK_PATH_ID.	Section 10.3.2
316	SURF ... N_LAYER_CELLS_MAX must be > 0.	Section 8.3.8
317	SURF ... Specify either ROUGHNESS or Z_0, not both.	Section 10.1.7
318	SURF ... MASS_FLUX_TOTAL is only for outflow.	Section 10.1.2

319	SURF ... must have a MATL_ID.	Section 8.4.2
320	SURF ... cannot use both MASS_FLUX and MASS_FRACTION.	Section 10.1.6
321	SURF ... MASS_FLUX cannot be less than zero.	Section 10.1.6
322	SURF ... cannot use both MASS_FLUX and VEL.	Section 10.1.6
323	SURF ... cannot use both MASS_FLUX and MASS_FLUX_TOTAL.	Section 10.1.2
324	SURF ... cannot use both MASS_FLUX_TOTAL and VEL.	Section 10.1.2
325	SURF ... cannot use a negative MASS_FRACTION.	Section 10.1.6
326	SURF ... cannot use RAMP_MF with MLRPUA or HRRPUA.	Section 11.1
327	SURF ... Must define SPEC_ID when using MASS_FLUX ...	Section 10.1.6
328	SURF ... SPEC ... not found.	Section 10.1.6
329	SURF ... sum of mass fractions greater than 1.	Section 10.1.6
330	SURF ... cannot use background species for MASS_FRACTION.	Section 10.1.6
331	SURF ... cannot use a RAYLEIGH model with GRAV=0.	Section 8.2.2
332	SURF ... REFERENCE_HEAT_FLUX requires HRRPUA, ...	Section 9.1.4
333	SURF ... MATL_ID ... not found.	Section 10.3.2
334	SURF ... has too many materials.	Section 9.2
335	SURF ... cannot have a reacting MATL and IGNITION_TEMPERATURE.	Section 9.1.3
336	SURF ... cannot use both ADIABATIC and NET and CONVECTIVE ...	Section 8.2.3
337	SURF ... cannot use both NET and CONVECTIVE_HEAT_FLUX.	Section 8.2.2
338	SURF ... cannot use TMP_FRONT and NET_HEAT_FLUX.	Section 8.2.2
339	SURF ... RAMP_T_I cannot be used with HT3D.	Section 8.3.4
340	SURF ... RAMP_T_I requires a thermally thick surface.	Section 8.3.4
341	SURF ... RAMP_T_I cannot be used with TMP_FRONT_INITIAL.	Section 8.3.4
342	SURF ... RAMP_T_I cannot be used with TMP_FRONT.	Section 8.3.4
243	SURF ... RAMP_T_I cannot be used with TMP_BACK.	Section 8.3.4
344	SURF ... RAMP_T_I cannot be used with TMP_INNER.	Section 8.3.4
345	SURF ... VEL_BULK must be less than or equal to VEL.	Section 10.5
346	SURF ... ZONE ID for LEAK_PATH_ID(1) not found.	Section 10.3.2
347	SURF ... ZONE ID for LEAK_PATH_ID(2) not found.	Section 10.3.2
348	SURF ... Cannot set the same ZONE for each leakage path.	Section 10.3.2
349	SURF ... LEAK_PATH greater than number of ZONES.	Section 10.3.2
350	SURF ... needs a THICKNESS.	Section 8.3.7
351	SURF ... needs a LENGTH.	Section 8.3.7
352	SURF ... needs a WIDTH.	Section 8.3.7
353	PART ... SURF_ID ... not found.	Section 15.4.1
354	SURF ... PART_ID ... not found.	Section 15.5.1
355	SURF ... not Cartesian and cannot have a MATL with ...	Section 8.3.2
356	SURF ... zero emissivity of MATL ... is inconsistent ...	Section 8.3.2
357	SURF ... has a specified HRRPUA/MLRPUA and a pyrolysis model.	Section 9.1
359	SURF ... cannot have a specified velocity or volume flux.	Section 9.1
360	SURF ... uses HRRPUA and species ... is the FUEL for ...	Section 9.1
361	SURF ... uses HRRPUA and MASS_FRACTION but no REAC ...	Section 13.1.1
362	SURF ... uses MLRPUA and species ... is the FUEL for ...	Section 9.1
363	SURF ... uses MLRPUA and MASS_FRACTION but no REAC ...	Section 13.1.1
364	SURF ... cannot specify mass fraction with mass flux ...	Section 10.1.6
365	SURF ... cannot specify mass fraction and outflow velocity.	Section 10.1.6
366	SURF ... cannot leak and specify flow or pyrolysis ...	Section 10.3.2
367	SURF ... cannot have both a mass flux and specified velocity.	Section 10.1.6

368	SURF ... has a problem with VARIABLE_THICKNESS or HT3D.	Section 8.4.2
369	SURF ... cannot have both HT3D and BURN_AWAY.	Section 8.4.2
370	SURF ... No wall or particle for TGA_ANALYSIS.	Section 9.3.2
371	Pressure solver ... not known.	Section 21.1.1
372	Cannot use FISHPAK_BC for multiple mesh simulations.	Section 7.4.2
373	Cannot have FISHPAK_BC>0.	Section 7.4.2
374	Numerical Instability - FDS stopped.	Section 4.2
375	OBST ... is VARIABLE_THICKNESS or HT3D and needs a MATL_ID.	Section 8.4.2
376	Meshes must have the same y/z bounds for TUNNEL_PRECONDITIONER.	Section 21.3
377	SURF ... IMPINGING JET model requires ...	Section 8.2.2
378	SURF ... cannot be applied to a 3-D conducting solid.	Section 8.3.3
381	Need more spectral band limits.	Section 14.3.2
382	Spectral band limits should be given in ascending order.	Section 14.3.2
390	RAMP ... is externally controlled, requires INITIAL_VALUE.	Section 18.8
391	RAMP ... not found.	Chapter 11
392	RAMP ... has only one point.	Chapter 11
393	RAMP ... cannot specify both CTRL_ID and DEVC_ID.	Chapter 18.6.1
394	RAMP ... variable T must be monotonically increasing.	Chapter 11
395	RAMP with EXTERNAL_FILE is present but no EXTERNAL_FILENAME.	Section 18.8
396	Row ... of ... has a bad 1st latitude.	Section 18.3.1
397	Row ... of ... has a bad 2nd latitude.	Section 18.3.1
398	Row ... of ... has a bad 1st longitude.	Section 18.3.1
399	Row ... of ... has a bad 2nd longitude.	Section 18.3.1
400	Row ... of ... has a bad velocity.	Section 18.3.1
401	Row ... of ... has a bad mass flow.	Section 18.3.1
402	Row ... of ... has a bad wavelength.	Section 15.3.2
403	Row ... of ... has a bad real index.	Section 15.3.2
404	Row ... of ... has a bad complex index.	Section 15.3.2
407	TABLE ... not found.	Section 18.3.1
421	SURF ... cannot be applied to a thin obstruction ...	Section 7.2.2
422	VENT ... cannot be applied to a thin obstruction ...	Section 7.2.2
423	HT3D solid must have at least one face exposed ...	Section 8.4.2
424	HT3D thin solid must have at least one face exposed ...	Section 8.4.2
425	MESH ... can stretch in at most 2 coordinate directions.	Section 6.3.5
426	MESH ... Poisson initialization error: ...	Section 4.2
427	DEVC ... requires repositioning.	Section 22.2.1
428	DEVC ... must be associated with a heat-conducting surface.	Section 22.13
429	PROF ... requires repositioning.	Section 22.3
430	PROF ... must be associated with a heat-conducting surface.	Section 22.13
431	MESH ... is not in alignment with MESH ...	Section 6.3.4
432	SURF ... must specify velocity boundary condition ...	Section 10.1.6
433	SURF ... cannot be applied below GROUND_LEVEL.	Section 16.4
434	RAMP_V_X assigned to SURF ...	Section 11.3
435	RAMP_V_Y assigned to SURF ...	Section 11.3

436	RAMP_V_Z assigned to SURF ...	Section 11.3
437	SURF ... cannot be applied to an exterior boundary.	Section 8.4.2
438	SURF ... layers are thicker than the underlying obstruction.	Section 8.4.3
439	MESH ... UVWFILE ... does not exist.	Section 20.5
440	MESH ... TMPFILE ... does not exist.	Section 20.5
441	MESH ... SPECFILE ... does not exist.	Section 20.5
442	SURF ... has more RAMP_Q than REFERENCE_HEAT_FLUX.	Section 9.1.4
443	SURF ... has more REFERENCE_HEAT_FLUX than RAMP_Q.	Section 9.1.4
444	SURF ... If one REFERENCE_THICKNESS is set, all must be set.	Section 9.1.4
445	SURF ... REFERENCE_HEAT_FLUX values must increase for each thickness.	Section 9.1.4
446	SURF REFERENCE_THICKNESS values must increase for each new group of REFERENCE_HEAT_FLUX.	Section 9.1.4
447	RAMP ... used with REFERENCE_HEAT_FLUX must start at T = 0.	Section 9.1.4
501	No ID provided ...	Section 10.2
502	Invalid TYPE_ID provided ...	Section 10.2
503	Must have both DUCTs and NODEs in the input file.	Section 10.2
504	Duct has no ID ...	Section 10.2
505	Duct has no AREA, DIAMETER, or PERIMETER ...	Section 10.2.1
506	Duct without AREA has no DIAMETER ...	Section 10.2.1
507	If both ROUND and SQUARE are FALSE, Duct with DIAMETER must also have PERIMETER ...	Section 10.2.1
508	Duct cannot input both PERIMETER and DIAMETER with AREA ...	Section 10.2.1
509	Can only specify one of CTRL_ID or DEVC_ID. ...	Section 10.2.1
510	Duct can only have one of damper, fan or aircoil ...	Section 10.2.1
511	Duct has fan specified but no fans have been defined ...	Section 10.2.1
512	Duct has both MASS_FLOW and VOLUME_FLOW defined ...	Section 10.2.1
513	Ductnode has no ID ...	Section 10.2
514	Ambient or internal ductnode requires an elevation, XYZ(3)...	Section 10.2.3
515	Non-AMBIENT or non VENT-connected ductnode needs >=2 ducts	Section 10.2.3
516	AMBIENT or VENT-connected ductnode must have 1 duct ...	Section 10.2.3
517	No ducts specified for ductnode ...	Section 10.2.3
518	Ductnode with a filter must have 2 ducts ...	Section 10.2.5
519	Fan has no ID ...	Section 10.2
520	FAN can only be one of constant volume, quadratic or ramp ...	Section 10.2.4
521	IF one of MAX_PRESSURE or MAX_FLOW given, both must be specified ...	Section 10.2.4
522	MAX_PRESSURE must be > 0 ...	Section 10.2.4
523	MAX_FLOW must be > 0 ...	Section 10.2.4
524	Filter has no ID ...	Section 10.2
525	Problem with filter ... SPEC ... not found	Section 10.2.5
526	Aircoil has no ID ...	Section 10.2
527	Localized leakage has no ID ...	Section 10.3.2
528	Leakage path must have VENT_ID defined ...	Section 10.3.2
529	Leakage to AMBIENT must have VENT2_ID for the AMBIENT node ...	Section 10.3.2
530	Leakage path must have VENT2_ID defined ...	Section 10.3.2
531	Leakage has no AREA ...	Section 10.3.2

532	Can only have one HVAC input with TYPE_ID of DUCT QUANTITY LIST ...	Section 22.16
533	Can only have one HVAC input with TYPE_ID of NODE QUANTITY LIST ...	Section 22.16
534	Both nodes have the same ID ...	Section 10.2.1
535	Two ducts with the same ID ...	Section 10.2
536	Duct ... Node ... does not contain the duct in its duct list.	Section 10.2.3
537	... ductnode not located for duct	Section 10.2.1
538	Fan not located ...	Section 10.2.1
539	Aircoil not located ...	Section 10.2.1
540	Two duct nodes with the same ID ...	Section 10.2
541	VENT for ductnode has a DEVC_ID or CTRL_ID ...	Section 10.2.3
542	Ductnode attached to VENT without SURF_ID HVAC ...	Section 10.2.3
543	Cannot leak and specify flow or pyrolysis at the same time ...	Section 10.3.2
544	Cannot specify custom leakage and zone leakage with the same surface ...	Section 10.3.2
545	Cannot find VENT_ID: ... for Ductnode: ...	Section 10.2.3
546	Ductnode cannot be AMBIENT and have an assigned VENT_ID	Section 10.2.3
547	Internal ductnode must have at least two attached ducts ...	Section 10.2.3
548	External ductnode can only have one attached duct. Ductnode: ...	Section 10.2.3
549	Duct ... not found. Ductnode: ... Ductnode ID: ...	Section 10.2.3
550	Duct: ... does not contain Ductnode: ...	Section 10.2.1
551	Problem with ductnode: ... FILTER ... not found	Section 10.2.5
552	Ductnode must lie with a single pressure zone. Node: ...	Section 10.2.3
553	Cannot specify fixed flows for all branches of internal ductnode ...	Section 10.2.9
554	Duct has no LENGTH and one node lacks an XYZ. Duct: ...	Section 10.2.1
555	Problem with HVAC network. Insufficient LOSS definitions for DUCTs and NODEs	Section 10.2.9
556	Duct ... has N_CELLS=0 and is in a duct run with ducts that have N_CELLS>0.	Section 10.2.8
557	SPEC_ID ... is not explicitly specified for QUANTITY ...	Section 22.16
558	Output QUANTITY ... requires a SPEC_ID	Section 22.16
559	Output QUANTITY ... SPEC_ID ... not found	Section 22.16
560	HVAC_SMV QUANTITY ... not appropriate for .hvac file.	Section 22.16
561	HVAC_SMV QUANTITY ... not found	Section 22.16
562	VENT ID ... used for localized leakage has SURF_ID HVAC.	Section 10.3.2
563	VENT ID ... used for localized leakage has a DEVC_ID or CTRL_ID.	Section 10.3.2
564	VENT: ... is used for HVAC and attached to a removable OBST.	Section 10.2.3
565	REAC: ... FALLOFF-LINDEMANN reactions must have A_LOW_PR and E_LOW_PR	Section 13.3.5
566	REAC: ... FALLOFF-TROE reactions must have A_LOW_PR, E_LOW_PR, A_TROE, T1_TROE, and T3_TROE.	Section 13.3.5
567	REAC: ... Tracked species ... used in a finite rate reaction without N_S defined is not a primitive species.	Section 13.3
601	OBST ... MULT_ID ... not found.	Section 7.6
602	OBST ... SHAPE requires RADIUS.	Section 7.6.2

603	OBST ... SHAPE requires HEIGHT.	Section 7.6.2
604	OBST ... BOX SHAPE requires LENGTH, WIDTH, HEIGHT.	Section 7.6.2
605	SURF_ID ... does not exist.	Section 7.2.1
606	SURF_ID_INTERIOR ... does not exist.	Section 9.2.9
607	OBST ... needs a BULK_DENSITY if it is to BURN_AWAY.	Section 9.2.9
608	MATL_ID ... not found.	Section 8.4.5
609	MULT_ID ... not found on HOLE line ...	Section 7.6
610	HOLE ... Cannot overlap HOLES with a DEVC or CTRL_ID.	Section 7.2.8
611	OBST ... has a BULK_DENSITY but zero volume.	Section 9.2.9
612	OBST ... must have a volume to be assigned HT3D.	Section 8.4.4
613	OBST ... and OBST ... cannot overlap in Mesh ...	Section 7.2.4
701	problem with GEOM, local SURF_ID index ... out of bounds.	Section 7.3.2
702	problem with GEOM, SURF_IDS not defined properly.	Section 7.3.1
703	missing SURF_ID in &GEOM line ...	Section 7.3.9
704	problem with GEOM, number of surfaces in SURF_ID field ...	Section 7.3.9
705	could not read binary connectivity for GEOM ...	Section 7.3.9
706	GEOM Expected ... Z values, found ...	Section 7.3.6
707	GEOM IJK(1)and IJK(2)on &GEOM line needs to be at least 2.	Section 7.3.6
708	GEOM At least 4 XB values ... required when using ZVALS.	Section 7.3.6
709	For terrain GEOM, unconnected boundary edge at node number ...	Section 7.3.6
710	For terrain GEOM, unconnected boundary edge at nodes ...	Section 7.3.6
711	For terrain GEOM, same boundary vertex ...	Section 7.3.6
712	For extruded Polygon GEOM, extrusion distance ...	Section 7.3.6
713	For extruded Polygon GEOM, Number of POLY indexes ...	Section 7.3.6
714	For extruded Polygon GEOM, Repeated vertex ...	Section 7.3.6
715	For extruded Polygon GEOM, Vertices X, Y have same position.	Section 7.3.6
716	problem with GEOM, the surface ID, is not defined.	Section 7.3.2
717	problem with GEOM, vertex index out of bounds.	Section 7.3.2
718	Out of Bounds.GEOM, FACE=, has vertex index I less than 1.	Section 7.3.2
719	Out of Bounds.GEOM, FACE=, has vertex index I higher than ...	Section 7.3.2
720	GEOM has currently unsupported BURN_AWAY feature ...	Section 7.3.11
721	For extruded Polygon GEOM, Node I not in the plane of ...	Section 7.3.6
722	GEOM Segments ... intersect in average POLY plane.	Section 7.3.6
723	For extruded Polygon GEOM, Not enough valid vertices ...	Section 7.3.6
724	For extruded Polygon GEOM, Could not triangulate polygon.	Section 7.3.6
725	&GEOM, MOVE_ID is not recognized.	Section 7.3.8
726	GEOM ID Unknown: Face normals are probably pointing in ...	Section 7.3.2
727	GEOM ID=, Edge length too small at ...	Section 7.3.3
728	GEOM ID=, Face area too small at ...	Section 7.3.3
729	GEOM ID=, Geometry volume too small.	Section 7.3.3
730	GEOM ID=, Open geometry at edge with nodes ...	Section 7.3.3
731	GEOM ID=, Non manifold geometry in adjacent faces at edge ...	Section 7.3.3
732	GEOM ID=, Opposite normals on triangles sharing edge with ...	Section 7.3.3
733	GEOM ID=, Open geometry at edge with nodes ...	Section 7.3.3
734	Mismatched mesh boundary location between meshes ...	Section 7.3.2
801	VENT ... cannot use MULT_ID because it involves HVAC.	Section 10.2

802	VENT ... needs an explicit ID because it involves HVAC.	Section 10.2
803	VENT ... cannot use MULT_ID because it uses PBX, PBZ or PBZ.	Section 7.6
804	VENT ... should use PBX, PBZ or XB if it is PERIODIC.	Section 7.4.2
805	VENT ... cannot use MULT_ID because it uses MB.	Section 7.6
806	VENT ... must set MB to XMIN, XMAX, YMIN, YMAX, ZMIN, or ZMAX.	Section 7.4.1
807	VENT ... cannot use MULT_ID because it uses DB.	Section 7.6
808	VENT ... must set DB to XMIN, XMAX, YMIN, YMAX, ZMIN, or ZMAX.	Section 7.4.1
809	VENT ... cannot be specified on a y boundary in a 2D calc.	Section 6.3.2
810	VENT ... must be a plane.	Section 7.4.1
811	VENT ... MULT_ID ... not found.	Section 7.6
812	VENT ... SURF_ID ... not found.	Section 7.4.1
813	VENT ... cannot be controlled by a device.	Section 7.4.2
814	VENT ... requires center point XYZ.	Section 7.4.2
815	VENT ... L_EDDY = 0 in Synthetic Eddy Method.	Section 10.1.8
816	VENT ... VEL_RMS = 0 in Synthetic Eddy Method.	Section 10.1.8
817	VENT ... Synthetic Eddy Method not permitted with HVAC.	Section 10.1.8
818	VENT ... requires on orientation index, IOR.	Section 7.4.4
819	VENT ... is OPEN, MIRROR OR PERIODIC ... exterior boundary.	Section 7.4.2
820	VENT ... has no solid backing or an IOR is needed.	Section 7.4.1
821	VENT ... cannot have normal component of UVW equal to 0.	Section 10.2.7
822	VENT ... has the same ID as another VENT.	Section 10.2
841	INIT ... has an unknown MULT_ID ...	Section 7.6
842	INIT ... cannot have both MASS and VOLUME_FRACTION.	Section 20.1
843	INIT ... requires a PART_ID.	Section 15.5.3
844	INIT ... requires a SPEC_ID.	Section 20.1
845	INIT ... cannot find SPEC_ID ...	Section 20.1
846	INIT ... cannot have sum of specified mass fractions > 1.	Section 20.1
847	INIT ... cannot use background species for MASS_FRACTION.	Section 20.1
848	INIT ... cannot use N_PARTICLES and N_PARTICLES_PER_CELL.	Section 15.5.3
849	INIT ... XB has no volume.	Section 20.1
850	INIT ... cannot find PART_ID ...	Section 15.5.3
851	INIT ... requires N_PARTICLES=1 for a PATH_RAMP.	Section 15.5.3
852	INIT ... PATH_RAMP(N)not found.	Section 15.5.3
853	INIT ... ORIENTATION_RAMP(N)not found.	Section 15.5.3
854	INIT ... ORIENTATION_RAMP components must all be defined.	Section 15.5.3
855	INIT ... DEVC_ID ... cannot use a SPATIAL_STATISTIC.	Section 15.5.3
856	The INIT_ID for DEVC ... cannot be found.	Section 15.4
857	INIT ... PARTicle class ... requires a density.	Section 15.5.3
858	INIT ... PARTicle class ... requires a ...	Section 15.5.3
871	LEAK_AREA specified twice for ZONE ... and ...	Section 10.3.2
872	ZONE ... overlaps ZONE ... in MESH ...	Section 10.3.1
873	ZONE ... meets ZONE ... at the boundary of MESH ...	Section 10.3.1
881	DEVC ... has QUANTITY_RANGE(2)<= QUANTITY_RANGE(1).	Section 22.2.3
882	DEVC ... MOVE_ID is not recognized.	Section 22.2.5
883	DEVC ... must have coordinates given in terms of XBP.	Section 22.2.5

884	DEVC ... components of ORIENTATION are all zero.	Section 18.1
885	DEVC ... must have coordinates, even if not a point quantity.	Section 18.1
886	DEVC ... must have either an output QUANTITY or PROP_ID.	Section 18.1
887	DEVC ... must have an ORIENTATION.	Section 22.10.12
888	DEVC ... TEMPORAL_STATISTIC is not recognized.	Section 22.2.3
889	DEVC ... SPATIAL_STATISTIC is not recognized.	Section 22.2.3
890	DEVC ... STATISTICS is not recognized.	Section 22.2.3
891	DEVC ... HVAC outputs specified with no HVAC inputs.	Section 10.2
892	DEVC ... or PROP ... must have a QUANTITY.	Section 10.2
893	DEVC ... QUANTITY ... requires XB coordinates.	Section 22.2.3
894	DEVC ... uses invalid solid phase SPATIAL_STATISTIC.	Section 22.2.3
895	DEVC ... uses invalid gas phase SPATIAL_STATISTIC.	Section 22.2.3
896	DEVC ... requires XB for SPATIAL_STATISTIC.	Section 22.2.3
897	DEVC ... NODE 1 = NODE 2.	Section 22.16
898	DEVC ... should not use XYZ or XB for an HVAC output QUANTITY.	Section 22.16
901	CTRL ... must have a FUNCTION_TYPE.	Section 18.4
902	CTRL ... PID controller must be given a TARGET_VALUE.	Section 18.5.7
903	CTRL ... FUNCTION_TYPE not recognized.	Section 18.4
904	CTRL ... FUNCTION_TYPE=EXTERNAL but no EXTERNAL_FILENAME given.	Section 18.8
905	CTRL ... must have at least one input.	Section 18.5
906	CTRL ... must have only one input.	Section 18.5
907	CTRL ... cannot use a CONSTANT INPUT_ID.	Section 18.5
908	CTRL ... must have at least two inputs.	Section 18.5
909	CTRL ... must have only two inputs.	Section 18.5
910	CTRL ... cannot use a control function as an input to itself.	Section 18.5
911	CTRL ... can only specify one input as a constant value.	Section 18.5.6
912	CTRL ... has no CONSTANT specified.	Section 18.5.6
913	CTRL ... uses PERCENTILE and must have a DEVC as input.	Section 18.5.8
914	CTRL ... is CUSTOM and must have a DEVC or math CTRL as input.	Section 18.5.5
915	CTRL ... input ... is the ID for both a DEVC and a CTRL.	Section 18.5
916	CTRL ... cannot locate INPUT_ID ...	Section 18.5
931	DEVC ... is a smoke detector and must have a PROP_ID.	Section 18.3.5
932	DEVC ... is a smoke detector and requires a smoke source.	Section 18.3.5
933	DEVC ... must have a PROP_ID.	Section 18.3.1
934	DEVC ... PROP ... needs an ACTIVATION_TEMPERATURE.	Section 18.3.1
935	DEVC ... is a THERMOCOUPLE and cannot use SPATIAL_STATISTIC.	Section 22.10.8
936	DEVC ... must have a MATL_ID.	Section 22.2.2
937	DEVC ... is partially outside of the domain.	Section 22.10.7
938	DEVC ... CTRL_ID ... does not exist.	Section 18.6.1
939	DEVC ... CTRL_ID ... is a logic only function ...	Section 18.6.1
940	DEVC ... must use QUANTITY='DENSITY' and a SPEC_ID.	Section 18.3.7
941	DEVC ... must be listed after all of its inputs.	Section 18.3.7
942	DEVC ... is a FED detector and requires an activity.	Section 22.10.18
943	DEVC ... is a PATCH and needs a DEVC_ID to control it.	Section 18.3.3
951	PROF ... requires an orientation index, IOR.	Section 22.3

952	PROF ... is not valid.	Section 22.3
953	PROF ... requires a PART_ID.	Section 22.3
961	CELL_CENTERED not allowed with AGL_SLICE.	Section 22.10.20
962	BNDF ... CPUA_Z, MPUA_Z, and AMPUA_Z require liquid droplets.	Section 22.9.1
1001	SPEC_ID ... for AERO. VOL. FRAC. must be a tracked species.	Section 22.10.23
1002	SPEC_ID ... for AERO. VOL. FRAC. cannot be a lumped species.	Section 22.10.23
1003	SPEC_ID ... for AERO. VOL. FRAC. is not an AEROSOL.	Section 22.10.23
1004	SPEC_ID ... is not explicitly specified for QUANTITY ...	Section 22.12
1005	Output QUANTITY ... requires a DUCT_ID.	Section 22.16
1006	CELL_L used for output QUANTITY ... is outside of DUCT_ID ...	Section 22.16
1007	SPEC_ID ... for FILTER LOADING is not a tracked species.	Section 22.16
1008	SPEC_ID ... for EQUIL. VAP. FRAC. is not a tracked species.	Section 22.16
1009	SPEC_ID ... for EQUIL. VAP. FRAC. is not an evap. species.	Section 22.16
1010	REAC ... MIXTURE FRACTION requires reaction A + B -> C.	Section 22.12
1011	HRRPUV REAC requires a REAC_ID.	Section 22.12
1012	REAC_ID ... not found for HRRPUV REAC.	Section 22.12
1013	Output QUANTITY2 ... requires a SPEC_ID.	Section 22.12
1014	Output QUANTITY2 ... is not appropriate for SLCF.	Section 22.12
1015	Output QUANTITY ... requires a QUANTITY2.	Section 22.12
1016	Output QUANTITY ... requires a SPEC_ID.	Section 22.12
1017	Output QUANTITY ... SPEC_ID ... not found.	Section 22.12
1018	Output QUANTITY ... requires a MATL_ID.	Section 22.12
1019	Output QUANTITY ... MATL_ID ... not found.	Section 22.12
1020	Output QUANTITY ... requires a PART_ID.	Section 22.12
1021	Output QUANTITY ... PART_ID ... not found.	Section 22.12
1022	Output QUANTITY ... requires a DUCT_ID.	Section 22.16
1023	Output QUANTITY ... DUCT_ID ... not found.	Section 22.16
1024	Output QUANTITY ... requires a positive CELL_L.	Section 22.16
1025	Output QUANTITY ... DUCT_ID ... requires pos. CELL_L.	Section 22.16
1026	Output QUANTITY ... DUCT_ID ... requires HVAC_MASS_TRAN.	Section 22.16
1027	Output QUANTITY ... requires a NODE_ID'	Section 22.16
1028	Output QUANTITY ... NODE_ID ... not found.	Section 22.16
1029	HUMIDITY requires SPEC_ID WATER VAPOR.	Section 12.1.1
1030	DIFFUSIVITY requires a tracked SPEC_ID when using DNS.	Section 12.1.3
1031	BACKGROUND species inappropriate for deposition.	Section 13.4
1032	SPEC_ID ... for SURF. DEP. is not an aerosol species.	Section 13.4
1033	SPEC_ID ... for SURF. DEP. is not an aerosol tracked species.	Section 13.4
1034	QUANTITY ... requires liquid droplets.	Section 22.13
1035	QUANTITY ... is not appropriate for SLCF.	Section 22.12
1036	QUANTITY ... is not appropriate for DEVC.	Section 22.12
1037	QUANTITY ... is not a particle output QUANTITY.	Section 22.15
1038	QUANTITY ... is not appropriate for BNDF.	Section 22.13
1039	QUANTITY ... is not appropriate for isosurface.	Section 22.12
1040	QUANTITY ... is not appropriate for gas phase.	Section 22.12
1041	QUANTITY ... is not appropriate for Plot3D.	Section 22.12
1042	QUANTITY ... not found.	Section 22.12

1043	PROP_ID ... not found.	Section 22.12
1044	Cannot set EXTERNAL_FILE=T for a RAMP used for an output clock ...	Section 18.8
1045	QUANTITY ... requires a PART_ID for particles with mass.	Section 22.7
1051	RESTART initial time equals T_END.	Section 5.6

Part III

FDS and Smokeview Development Tools

Chapter 25

The FDS and Smokeview Repositories

For those interested in obtaining the FDS and Smokeview source codes, either for development work or simply to compile on a particular platform, it is strongly suggested that you download onto your computer the FDS and Smokeview repositories. All project documents are maintained using the online utility [GitHub](#), a free service that supports software development for open source applications. GitHub uses Git version control software. Under this system, a centralized repository containing all project files resides on a GitHub server. Anyone can obtain a copy of the repository or retrieve a specific revision of the repository. However, only the FDS and Smokeview developers can commit changes directly to the repository. Others must submit a “pull request.” Detailed instructions for checking out the FDS repository can be found at <https://github.com/firemodels/fds>.

Both FDS and Smokeview live within a GitHub *organization* called “firemodels”. The current location of the organization is <https://github.com/firemodels>. The repositories that are used by the FDS and Smokeview projects are listed below along with a brief description:

fds	FDS source code, verification and validation tests, wikis, and documentation
smv	Smokeview source code, integration tests, and documentation
exp	Experimental data repository for FDS validation
fig	Figures used by various manuals
out	FDS output results for validation
bot	Firebot (continuous integration system) source
fds-smv	Web page html source

The wiki pages in the fds repository are particularly useful in describing the details of how you go about working with the repository assets.

Chapter 26

Compiling FDS

If a compiled version of FDS exists for the machine on which the calculation is to be run and no changes have been made to the original source code, there is no need to re-compile the code. For example, the file `fds.exe` is the compiled program for a Windows-based PC; thus PC users do not need a Fortran compiler and do not need to compile the source code. For computers for which an executable has not been compiled, you must compile the code. A Fortran 2018 compliant compiler is required.

26.1 FDS Source Code

Table 26.1 lists the files that make up the FDS source code. Files with the “.f90” suffix contain free-form Fortran 90 instructions conforming to the ANSI and ISO standards (2018 edition). A `makefile` is available in the [Build directory of the FDS GitHub repository](#) that contains platform specific options for compilation. Note the following:

- The source code consists entirely of Fortran statements organized into 32 files. Some compilers have a standard optimization level, plus various degrees of “aggressive” optimization. Be cautious in using the highest levels of optimization.
- To compile FDS, you need a Fortran compiler (2018 or greater) and MPI libraries.
- To compile routines for the ULMAT and UGLMAT pressure solvers, Intel Math Kernel Libraries (MKL) are required.
- To compile routines for the HYPRE (iterative linear solvers), the HYPRE Library [78] from Lawrence Livermore National Laboratory is required.
- To compile routines for the CVODE (finite-rate chemistry), the Sundials Library [25] from Lawrence Livermore National Laboratory is required.

Table 26.1: FDS source code files

File Name	Description
ccib.f90	Complex geometry physics routines
chem.f90	Finite-rate chemistry routines
cons.f90	Global arrays and constants
ctrl.f90	Definitions and routines for control functions
data.f90	Data for output quantities and thermo-physical properties
devc.f90	Derived type definitions and constants for devices
divg.f90	Compute the flow divergence
dump.f90	Output data dumps into files
fire.f90	Combustion routines
func.f90	Global functions and subroutines
geom.f90	Complex computational geometry engine
gsmv.f90	Complex geometry visualization routines
hvac.f90	Heating, Ventilation, and Air Conditioning
imkl.f90	Bindings for the Intel MKL library
init.f90	Initialize variables and Poisson solver
main.f90	Main program
mass.f90	Mass equation(s) and thermal boundary conditions
mesh.f90	Arrays and constants associated with each mesh
part.f90	Lagrangian particle transport and sprinkler activation
pois.f90	Poisson (pressure) solver
prec.f90	Specification of numerical precision
pres.f90	Spatial discretization of pressure (Poisson) equation
prop.f90	Gas and liquid property data
radi.f90	Radiation solver
rcal.f90	Functions needed for radiation solver, including RadCal
read.f90	Read input parameters
smvv.f90	Routines for computing and outputting 3D smoke and isosurfaces
soot.f90	Soot agglomeration and aerosol deposition
turb.f90	Turbulence models and manufactured solutions
type.f90	Derived type definitions
vege.f90	Experimental vegetation model
velo.f90	Momentum equations
wall.f90	Wall boundary conditions

Chapter 27

Output File Formats

The output from the code consists of the file `CHID.out`, plus various data files that are described below. Most of these output files are written out by the subroutines within `dump.f90`, and can easily be modified to accommodate various plotting packages.

27.1 Diagnostic Output (`.out`)

The file `CHID.out` contains diagnostic output, including an accounting of various important quantities, including CPU usage. Typically, diagnostic information is printed out every 100 time steps as follows:

```
Time Step 137431   December 27, 2015  00:29:49
Step Size:   0.563E-01 s, Total Time:   1800.04 s
Pressure Iterations:      1
Maximum Velocity Error:  0.30E-01 on Mesh  1 at (  0 44  3)
-----
Max CFL number:  0.94E+00 at (  1, 46,  1)
Max divergence:  0.13E+00 at ( 66, 12,  1)
Min divergence: -0.20E+00 at ( 66, 13,  1)
Max VN number:   0.51E+00 at (  1, 25, 18)
No. of Lagrangian Particles:      27
```

The `Time Step` indicates the total number of iterations. The date and time indicate the current wall clock time. The `STEP SIZE` indicates the size of the numerical time step. The `Total Time` indicates the total simulation time calculated up to that point. The `Pressure Iterations` are the number of iterations of the pressure solver for the corrector (second) half of the time step. The pressure solver iterations are designed to minimize the error in the normal component of velocity at solid walls or the interface of two meshes. The `Maximum Velocity Error` indicates this error and in which grid cell it occurs. `Max/Min divergence` is the max/min value of the function $\nabla \cdot \mathbf{u}$ and is used as a diagnostic when the flow is incompressible (i.e., no heating); `Max CFL number` is the maximum value of the CFL number, the primary time step constraint; `Max VN number` is the maximum value of the Von Neumann number, the secondary time step constraint. The `No. of Lagrangian Particles` refers to the number of particles in the current mesh.

Following the completion of a successful run, a summary of the CPU usage per subroutine is listed in the file called `CHID_cpu.csv` (Section 27.6). This is useful in determining where most of the computational effort is being placed.

27.2 Heat Release Rate and Related Quantities (`_hrr.csv`)

The heat release rate of the fire, mass production rate of gas species, and zonal background pressures are automatically written into a comma-delimited spreadsheet file called `CHID_hrr.csv` every `DT_HRR` s. Details of the integrated energy quantities (columns with units of kW) can be found in Section 22.10.1. The production rate (kg/s) of the tracked gas species are labeled `MLR_XXX` for “Mass Loss Rate.” A positive value means that this particular gas species is being generated at a boundary or by a particle; a negative value indicates depletion. The background pressure (Pa) of each pressure zone is discussed in Section 10.3.

27.3 Device Output Data (`_devc.csv`)

The output of devices (link temperatures, smoke obscuration, thermocouples, etc.) specified in the input file under the namelist group `DEVC` is output in comma-delimited format to a file called `CHID_devc.csv`.

If the number of device columns exceeds `DEVC_COLUMN_LIMIT`, the file will automatically be split into smaller files.

27.4 Control Output Data

Data associated with particular control functions specified in the input file under the namelist group `CTRL` is output in comma delimited format in a file called `CHID_ctrl.csv`. The format of the file is as follows:

```
s      , status , status , ... , status
Time  , ID(1)  , ID(2)  , ... , ID(N_CTRL)
T(1)  , -001   , 001    , ... , -001
      .
      .
```

where `N_CTRL` is the number of controllers, `ID(I)` is the user-defined ID of the `I`th control function, and plus or minus 1's represent the state `-1 = F` and `+1 = T` of the `I`th control function at the particular time. The files can be imported into Microsoft Excel or almost any other spreadsheet program. If the number of control columns exceeds `CTRL_COLUMN_LIMIT`, the file will automatically be split into smaller files. The frequency of output is controlled by either `DT_CTRL` or `RAMP_CTRL` on the `DUMP` line.

27.5 Device and Control Log File

Each time a device or control function changes its logical state, an event is recorded in a comma delimited format in a file called `CHID_devc_ctrl_log.csv`. A row is created for each state change. The format of the file is as follows:

```
Time (s) , Type      , ID      , State  , Value , Units
T(1)     , Type(1)  , ID(1)  , State(1), Val(1), Units(1)
      .
      .
```

where for the `I`th state change `T(I)` is the time of the state change, `Type(I)` is type of the item that changed state (either `DEVC` or `CTRL`), `ID(I)` is the ID of item, `State(I)` is the logical state of the item after the state change (`T` for true or `F` for false), `Val(I)` is item value at the state change, and `Units(I)` are the units associated with the value. No `Value` is written for a logical control function (e.g. a `FUNCTION_TYPE` or `ALL`, `ANY`, `ONLY`, `AT_LEAST`, `X_OF_N`, `KILL`, or `RESTART`). For `FUNCTION_TYPES` of `DEADBAND` and `CUSTOM` the input

value to the function is written. No `Units` value is written for a control function. If the device or control function does not have an `ID` defined, then the number of the device or control function is written (i.e, if the item was the 5th device in the input file, then 5 would be written).

27.6 CPU Usage Data

The file called `CHID_cpu.csv` records the amount of CPU time for each of the MPI processes.

```
Rank,MAIN,DIVG, ... , Total T_USED (s)
0, 2.052E+00, 1.058E+01, ... , 5.143+01
1, 2.432E+00, 1.062E+01, ... , 5.123+01
.
.
```

where `Rank` is the number of the MPI process (starting at 0), `MAIN`, `DIVG`, and so on, are major routines, and 'Total T_USED (s)' is the total CPU time consumed by that particular MPI process. Typically, the total time is similar. The time spent in `MAIN` is essentially overhead – time spent *not* working on the calculation. If you want to know if your work load is balanced, take a look at the time spent in `MAIN`. It should be similar for all MPI processes. If one of the MPI processes has a noticeably smaller value for `MAIN`, then that process is working on the core routines while the other processes sit idle in `MAIN`.

The `CHID_cpu.csv` file is printed out at the end of the simulation. To force it to be printed out periodically during the simulation, set `DT_CPU` or `RAMP_CPU` on the `DUMP` line. The latter parameter allows you to write out the files at specified times.

27.7 Time Step Data

The file called `CHID_steps.csv` records data on the size of time steps and the amount of elapsed CPU time.

```
Time Step,Wall Time,Step Size,Simulation Time,CPU Time
1,2020-04-29T10:23:13.631-04:00, 0.100E+00, 0.10000, 0.23438
.
.
```

where `Time Step` is the current time step, `Wall Time` is the date and time for the indicated time step, `Step Size` is the size of the current time step in seconds, `Simulation Time` is the current time of the simulation in seconds, and `CPU Time` is the elapsed CPU time since starting FDS. Note that the first time step will include time spent reading the input file and initializing FDS.

27.8 Gas Mass Data

The total mass of the various gas species at any instant in time is reported in the comma delimited file `CHID_mass.csv`. The file consists of several columns, the first column containing the time in seconds, the second contains the total mass of all the gas species in the computational domain in units of kg, the next lines contain the total mass of the individual species.

You must specifically ask that this file be generated, as it can potentially cost a fair amount of CPU time to generate. Set `MASS_FILE=T` on the `DUMP` line to create this output file.

27.9 Slice Files (.sf)

The slice files defined under the namelist group `SLCF` are named `CHID_M_N.sf` where `M` is the mesh index and `N` is the quantity index. The files are unformatted and written from `dump.f90` with the following lines:

```
WRITE(LUSF) QUANTITY
WRITE(LUSF) SHORT_NAME
WRITE(LUSF) UNITS
WRITE(LUSF) I1, I2, J1, J2, K1, K2
WRITE(LUSF) TIME
WRITE(LUSF) ( ( (QQ(I, J, K), I=I1, I2), J=J1, J2), K=K1, K2)
      .
      .
WRITE(LUSF) TIME
WRITE(LUSF) ( ( (QQ(I, J, K), I=I1, I2), J=J1, J2), K=K1, K2)
```

`QUANTITY`, `SHORT_NAME` and `UNITS` are character strings of length 30. The sextuplet `(I1, I2, J1, J2, K1, K2)` denotes the bounding mesh cell nodes. The sextuplet indices correspond to mesh cell nodes, or corners, thus the entire mesh would be represented by the sextuplet `(0, IBAR, 0, JBAR, 0, KBAR)`.

There is a short Fortran 90 program provided, called `fds2ascii.f90`, that can convert slice files into text files that can be read into a variety of graphics packages. The program combines multiple slice files corresponding to the same “slice” of the computational domain, time-averages the data, and writes the values into one file, consisting of a line of numbers for each node. Each line contains the physical coordinates of the node, and the time-averaged quantities corresponding to that node. In particular, the graphics package Tecplot reads this file and produces contour, streamline and/or vector plots. See Section 22.11 for more details about the program `fds2ascii`.

27.10 Plot3D Data (.xyz, .q)

Quantities over the entire mesh can be output in a format used by the graphics package Plot3D. The Plot3D data sets are single precision (32 bit reals), whole and unformatted. Note that there is blanking, that is, blocked out data points are not plotted. If the statement `WRITE_XYZ=T` is included on the `DUMP` line, then the mesh data is written out to a file called `CHID_M.xyz`

```
WRITE(LU13) IBAR+1, JBAR+1, KBAR+1
WRITE(LU13) ( ( (X(I), I=0, IBAR), J=0, JBAR), K=0, KBAR), &
      ( ( (Y(J), I=0, IBAR), J=0, JBAR), K=0, KBAR), &
      ( ( (Z(K), I=0, IBAR), J=0, JBAR), K=0, KBAR), &
      ( ( (IBLK(I, J, K), I=0, IBAR), J=0, JBAR), K=0, KBAR)
```

where `x`, `y` and `z` are the coordinates of the cell corners, and `IBLK` is an indicator of whether or not the cell is blocked. If the point `(x, y, z)` is completely embedded within a solid region, then `IBLK` is 0. Otherwise, `IBLK` is 1. Normally, the mesh file is not dumped.

The flow variables are written to a file called `CHID_M_<time>.q`, where `M` is the mesh number and `<time>` is the number of seconds when the data is output. The file is written with the lines

```
WRITE(LU14) IBAR+1, JBAR+1, KBAR+1
WRITE(LU14) ZERO, ZERO, ZERO, ZERO
WRITE(LU14) ( ( ( (QQ(I, J, K, N), I=0, IBAR), J=0, JBAR), K=0, KBAR), N=1, 5)
```

The five channels `N=1, 5` are by default the temperature ($^{\circ}\text{C}$), the u , v and w components of the velocity

(m/s), and the heat release rate per unit volume (kW/m³). Alternate variables can be specified with the input parameter `PLOT3D_QUANTITY(1:5)` on the `DUMP` line. Note that the data is interpolated at cell corners, thus the dimensions of the Plot3D data sets are one larger than the dimensions of the computational mesh.

Smokeview can display the Plot3D data. In addition, the Plot3D data sets can be read into some other graphics programs that accept the data format. This particular format is very convenient, and recognized by a number of graphics packages.

27.11 Boundary Files (.bf)

The boundary files defined under the namelist group `BNDF` are named `CHID_M_N.bf` where `M` is the mesh index and `N` is the quantity index. The files are unformatted and written from `dump.f90` with the following lines:

```
WRITE(LUBF) QUANTITY
WRITE(LUBF) SHORT_NAME
WRITE(LUBF) UNITS
WRITE(LUBF) N_PATCH
WRITE(LUBF) I1, I2, J1, J2, K1, K2, IOR, OBST_INDEX, NM
WRITE(LUBF) I1, I2, J1, J2, K1, K2, IOR, OBST_INDEX, NM
.
.
WRITE(LUBF) TIME_1
WRITE(LUBF) ((QQ(I, J, K), I=11, I2), J=J1, J2), K=K1, K2)
WRITE(LUBF) ((QQ(I, J, K), I=11, I2), J=J1, J2), K=K1, K2)
.
.
WRITE(LUBF) TIME_2
WRITE(LUBF) ((QQ(I, J, K), I=11, I2), J=J1, J2), K=K1, K2)
WRITE(LUBF) ((QQ(I, J, K), I=11, I2), J=J1, J2), K=K1, K2)
.
.
```

`QUANTITY`, `SHORT_NAME` and `UNITS` are character strings of length 30. `N_PATCH` is the number of planes (or “patches”) that correspond to solid surfaces. The sextuplet `(I1, I2, J1, J2, K1, K2)` defines the cell indices of each patch. For example, a patch on the lower x boundary of the domain that spans cells 2 through 5 in the y direction and 7 through 10 in the z direction would have patch indices (0,0,1,5,6,10). Note that the boundary patches are 2-D; thus, one pair of cell indices is the same. `IOR` is an integer indicating the orientation of the patch ($\pm 1, \pm 2, \pm 3$). `OBST_INDEX` refers to the solid obstruction (`OBST`) to which the patch is affixed. Its value is 0 if the patch is affixed to an external boundary. `NM` is the number of the mesh within which the obstruction lies. It may be different than the mesh containing the boundary data; that is, `NM` is not necessarily equal to the mesh index `M` in the file name. `TIME_N` and `QQ` are four-byte reals.

27.12 Particle Data (.prt5)

Coordinates and specified quantities related to tracer particles, sprinkler droplets, and other Lagrangian particles are written to a FORTRAN unformatted (binary) file called `CHID_M.prt5` where `M` is the mesh index. Note that the format of this file has changed from previous versions (4 and below). The file consists of some header material, followed by particle data output every `DT_PART` seconds. The time increment `DT_PART` is specified on the `DUMP` line. It is `T_END/NFRAMES` by default. The header materials is written by the following FORTRAN code in the file called `dump.f90`.

```

WRITE(LUPF) ONE_INTEGER          ! Integer 1 to check Endian-ness
WRITE(LUPF) NINT(VERSION*100.)   ! FDS version number
WRITE(LUPF) N_PART               ! Number of PARTICle classes
DO N=1,N_PART
  PC => PARTICLE_CLASS(N)
  WRITE(LUPF) PC%N_QUANTITIES,ZERO_INTEGER ! ZERO_INTEGER is a place holder
  DO NN=1,PC%N_QUANTITIES
    WRITE(LUPF) CDATA(PC%QUANTITIES_INDEX(NN)) ! 30 character output quantity
    WRITE(LUPF) UDATA(PC%QUANTITIES_INDEX(NN)) ! 30 character output units
  ENDDO
ENDDO

```

Note that the initial printout of the number 1 is used by Smokeview to determine the Endian-ness of the file. The Endian-ness has to do with the particular way real numbers are written into a binary file. The version number is used to distinguish new versus old file formats. The parameter `N_PART` is not the number of particles, but rather the number of particle classes corresponding to the `PART` namelist groups in the input file. Every `DT_PART` seconds the coordinates of the particles and droplets are output as 4 byte reals (by default):

```

WRITE(LUPF) REAL(T,FB) ! Write out the time T as a 4 byte real
DO N=1,N_PART
  WRITE(LUPF) NPLIM ! Number of particles in the PART class
  WRITE(LUPF) (XP(I),I=1,NPLIM),(YP(I),I=1,NPLIM),(ZP(I),I=1,NPLIM)
  WRITE(LUPF) (TA(I),I=1,NPLIM) ! Integer "tag" for each particle
  IF (PC%N_QUANTITIES > 0) WRITE(LUPF) ((QP(I,NN),I=1,NPLIM),NN=1,PC%N_QUANTITIES)
ENDDO

```

The particle “tag” is used by Smokeview to keep track of individual particles and droplets for the purpose of drawing streamlines. It is also useful when parsing the file. The quantity data, `QP(I,NN)`, is used by Smokeview to color the particles and droplets. Note that it is now possible with the new format to color the particles and droplets with several different quantities.

27.13 Profile Files

The profile files defined under the namelist group `PROF` are named `CHID_prof_N.csv` and are written out formatted. These files are written out from `dump.f90` with the following line:

```

WRITE(LU_PROF) T,NWP+1,(X_S(I),I=0,NWP),(Q(I),I=0,NWP)

```

After the time `T`, the number of node points is given and then the node coordinates. These are written out at every time step because the wall thickness and the local solid phase mesh may change over time due to the solid phase reactions. Array `Q` contains the values of the output quantity, which may be wall temperature, density or component density.

27.14 3-D Smoke Files (.s3d)

3-D smoke files contain alpha values used by Smokeview to draw semi-transparent planes representing smoke and fire. FDS outputs 3-D smoke data at fixed time intervals. A *pseudo-code* representation of the 3-D smoke file is given by:

```

WRITE(LU_SMOKE3D) ONE,VERSION,0,NX-1,0,NY-1,0,NZ-1

```

```

.
.
WRITE(LU_SMOKE3D_SIZE,*) TIME,NCHARS_IN,NCHARS_OUT
WRITE(LU_SMOKE3D) TIME
WRITE(LU_SMOKE3D) NCHARS_IN,NCHARS_OUT
IF (NCHARS_OUT > 0) WRITE(LU_SMOKE3D) (BUFFER_OUT(I),I=1,NCHARS_OUT)

```

The first ONE is an endian flag. Smokeview uses this number to determine whether the computer creating the 3-D smoke file and the computer viewing the 3-D smoke file use the same or different byte swap (endian) conventions for storing floating point numbers. The opacity data is converted from 4 byte floating point numbers to one byte integers then compressed using run-length encoding (RLE). The compressed data is contained in BUFFER_OUT. Run-length encoding is a compression scheme where repeated “runs” of data are replaced with a number (number of repeats), and the value repeated. Four or more consecutive identical characters are represented by *#nc* where *#* is a special character denoting the beginning of a repeated sequence, *n* is the number of repeats and *c* is the character repeated. *n* can be up to 254 (255 is used to represent the *special* character). Characters not repeated four or more times are listed as is. For example, the character string *aaaaaabbbbbcc* is encoded as *#6a#5bcc*.

27.15 Iso-surface Triangulation Files (.iso)

Iso-surface files (.iso) are used to store one or more surfaces where the specified QUANTITY is a specified value. FDS outputs iso-surface data at fixed time intervals. These surfaces are defined in terms of vertices and triangles. A vertex consists of an (*x,y,z*) coordinate. A triangle consists of 3 connected vertices. These files are written out from *smvv.f90* using lines equivalent to the following:

```

! header

WRITE(LU_ISO) ONE
WRITE(LU_ISO) VERSION
WRITE(LU_ISO) NISO_LEVELS
IF (NISO_LEVELS>0) WRITE(LU_ISO) (ISO_LEVELS(I),I=1,NISO_LEVELS)
WRITE(LU_ISO) ZERO
WRITE(LU_ISO) ZERO, ZERO

! Write for each STIME

WRITE(LU_ISO) STIME, ZERO
WRITE(LU_ISO) N_VERT, N_FACE
IF (N_VERT_D>0) WRITE(LU_ISO) (Xvert(I),Yvert(I),Zvert(I),I=1,N_VERT)
IF (N_FACE_D>0) WRITE(LU_ISO) (FACE1(I),FACE2(I),FACE3(I),I=1,N_FACE)
IF (N_FACE_D>0) WRITE(LU_ISO) (ISO_LEVEL_INDICES(I),I=1,N_FACE)

```

where:

- ONE Integer set to 1. Smokeview uses this number to determine whether the computer creating the geometry file and the computer viewing the geometry file use the same or different byte swap (endian) conventions for storing floating point numbers.
- VERSION currently has value 1 and indicates the version number of this file format.
- NISO_LEVELS The number of iso-surface levels (values) defined at the beginning of the file.
- ISO_LEVELS Floating point values for iso-surface levels of QUANTITY stored at the beginning of the file.

- `STIME` is the FDS simulation time.
- `N_VERT`, `N_FACE` are the number of vertices and triangles in the iso-surface.
- `Xvert`, `Yvert`, `Zvert` are the vertex coordinates.
- `FACE1`, `FACE2`, `FACE3` are the vertex indices for each triangle. The indices are numbered relative to how vertices were written out earlier.
- `LEVEL_INDICES` are the iso-surface level indices for each triangle (points to `ISO_LEVELS`).
- `ZERO` Integer set to 0, left for compatibility and future use.

Data files associated with coloring iso-surfaces by a second `QUANTITY2` are described in section [27.17](#)

27.16 Geometry Boundary, Unstructured Slice Files (`.gcf`, `.gsf`)

Immersed geometry surfaces and unstructured slices are stored using a file format described in this section. These surfaces are defined in terms of vertices and triangles. Surface polygonal patches (CFACEs) resulting from geometry triangulation intersection with the FDS grid are divided in triangles and used to plot geometry and boundary data (`.gcf`). Unstructured slices (`.gsf`) derive from the `&SLCF` namelist when the slice crosses geometries, and are also split in triangles. Data files associated with these slices are described in section [27.17](#). These files are written out from `dump.f90` using lines equivalent to the following:

```
WRITE(LU_GEOM) ONE
WRITE(LU_GEOM) VERSION
WRITE(LU_GEOM) ZERO, ZERO, ONE
WRITE(LU_GEOM) STIME
WRITE(LU_GEOM) N_VERT, N_FACE, ZERO
IF (N_VERT>0 .AND. N_FACE>0) THEN
  WRITE(LU_GEOM) (Xvert(I),Yvert(I),Zvert(I),I=1,N_VERT)
  WRITE(LU_GEOM) (FACE1(I),FACE2(I),FACE3(I),I=1,N_FACE)
  WRITE(LU_GEOM) (LOCATION(I),I=1,N_FACE)
  IF ( ".gcf" ) THEN
    WRITE(LU_GEOM) (SURF_IND(I),I=1,N_FACE)
    WRITE(LU_GEOM) (GEOM_IND(I),I=1,N_FACE)
  ENDIF
ENDIF
ENDIF
```

where:

- `ONE` Integer set to 1.
- `VERSION` currently has value 2 and indicates the version number of this file format.
- `STIME` is the initial FDS simulation time.
- `N_VERT`, `N_FACE` are the number of vertices and faces.
- `Xvert`, `Yvert`, `Zvert` are the vertex coordinates.
- `FACE1`, `FACE2`, `FACE3` are the vertex indices for each triangle. The indices are numbered relative to how vertices were written out earlier.
- `LOCATION` Integers that define which edges of the triangles belong to polygonal patches boundaries.

- SURF_IND are the SURF indices for each triangle. Only written in .gcf files.
- GEOM_IND are the geometry indices for each triangle, for geometries listed in the input file sequence. Only written in .gcf files.
- ZERO Integer set to 0. Entry left for future use.

27.17 Isosurface, Geometry and Unstructured Slice Data Files (.viso, .be)

Isosurface, geometry boundary and unstructured slice data files contain a description of data values computed by FDS on triangulations stemming either from isosurfaces (.viso) or geometry boundary/unstructured slice (.be) files. These files are analogous to boundary files. The data written out to an unstructured data file must correspond to the triangulation written out in the corresponding geometry file (.iso for .viso; .gcf, .gsf for .be). These data files are written out from dump.f90 with lines equivalent to the following:

```
WRITE(FUNIT_DATA) ONE
WRITE(FUNIT_DATA) FILETYPE
WRITE(FUNIT_DATA) STIME
WRITE(FUNIT_DATA) ZERO, ZERO, N_VERT_VALS, N_FACE_VALS
IF (N_VERT_VALS>0) WRITE(FUNIT_DATA) (ValVert(I), I=1,N_VERT_VALS)
IF (N_FACE_VALS>0) WRITE(FUNIT_DATA) (ValFace(I), I=1,N_FACE_VALS)
```

- ONE has the value 1.
- FILETYPE currently has value 1 for isosurface and 2 for geometry.
- STIME is the FDS simulation time.
- N_VERT_VALS, N_FACE_VALS is the number of data values written out for vertices and faces.
- ValVert, ValFace vertex and face data. Currently both geometry and isosurface files export either face or node data in a single file.
- ZERO Integer set to 0, left for compatibility and future use.

27.18 Terrain Data Files (.ter)

For simulations involving outdoor terrain, files of the form CHID_M.ter are created for each mesh, M, with some terrain cutting through. The write statements are:

```
WRITE(LU_TERRAIN(NM)) REAL(ZS_MIN-1._EB,FB)
WRITE(LU_TERRAIN(NM)) M%IBP1,M%JBP1
WRITE(LU_TERRAIN(NM)) (M%XPLT(I), I=0,M%IBAR)
WRITE(LU_TERRAIN(NM)) (M%YPLT(J), J=0,M%JBAR)
WRITE(LU_TERRAIN(NM)) ((Z_TERRAIN(I,J), J=0,M%JBAR), I=0,M%IBAR)
```

NM is the mesh number, M is a pointer to the variables of the mesh NM, ZS_MIN is the lowest z value of the entire domain, IBP1 and JBP1 are the number of horizontal grid cells in the mesh plus 1, XPLT and YPLT are the four byte (FB) coordinates, and Z_TERRAIN(I, J) is the elevation at the point with indices (I, J). All real numbers are four bytes. The elevations are defined at cell corners.

27.19 FDS GEOM I/O binary format (.bingeom)

The binary format for geometries read by FDS as described in section 7.3.9 is the following. Binary files read by the FDS released pre-compiled bundles must have *little endian* endianness.

```
WRITE(LU_BINGEOM) GEOM_TYPE
WRITE(LU_BINGEOM) N_VERTS, N_FACES, N_SURF_ID, N_VOLUS
WRITE(LU_BINGEOM) VERTS(1:3*N_VERTS)
WRITE(LU_BINGEOM) FACES(1:3*N_FACES)
WRITE(LU_BINGEOM) SURFS(1:N_FACES)
WRITE(LU_BINGEOM) VOLUS(1:4*N_VOLUS)
```

where:

- **GEOM_TYPE** Integer equal to 1 for CAD geometries, and 2 for terrains (top surfaces only). All FDS analytical geometries, including extruded POLYs, are written to binary as general CAD geometries.
- **N_VERTS, N_FACES, N_SURF_ID, N_VOLUS** number of vertices, triangles, surface types and volumes on the geometry.
- **VERTS** one dimensional array with x, y, z positions of the **N_VERTS** vertices.
- **FACES** one dimensional array with the n_1, n_2, n_3 node connectivities of the **N_FACES** triangles.
- **SURFS** one dimensional array with **SURF_IDS** of the **N_FACES** triangles.
- **VOLUS** one dimensional array with n_1, n_2, n_3, n_4 node connectivities of the **N_VOLUS** tetrahedrons. This array is currently unused in FDS and left for 3D solid heat transfer in the future.

27.20 Unstructured Geometry (.ge, .ge2)

Immersed geometric objects are stored for SMV reading using a file format with .ge extension described below. These objects are defined in terms of vertices and triangles. A vertex is represented as an (x, y, z) coordinate (three floating point values). A triangle is represented as three vertex (integer) indices. Tetrahedron volumes option in the following are not used by FDS, and left as an option for future functionality. An associated file format used to store data on the geometric object is described in the next section. These files are written out from `dump.f90, geom.f90` using lines equivalent to the following:

```
WRITE(LU_GEOM) ONE
WRITE(LU_GEOM) VERSION
WRITE(LU_GEOM) N_FLOATS, N_INTS, FIRST_FRAME_STATIC
IF (N_FLOATS>0) WRITE(LU_GEOM) (FLOAT_HEADER(I), I=1, N_FLOATS)
IF (N_INTS>0) WRITE(LU_GEOM) (INT_HEADER(I), I=1, N_INTS)

WRITE(LU_GEOM) STIME
WRITE(LU_GEOM) N_VERT, N_FACE, N_VOL
IF (N_VERT>0) WRITE(LU_GEOM) (Xvert(I), Yvert(I), Zvert(I), I=1, N_VERT)
IF (N_FACE>0) THEN
  WRITE(LU_GEOM) (FACE1(I), FACE2(I), FACE3(I), I=1, N_FACE)
  WRITE(LU_GEOM) (SURF_IND(I), I=1, N_FACE)
  WRITE(LU_GEOM) (Xtext(I), Ytext(I), I=1, 3*N_FACE)
ENDIF
IF (N_VOL>0) THEN
```



```

WRITE (LU_GEOM) (VOL1 (I) , VOL2 (I) , VOL3 (I) , VOL4 (I) , I=1, N_VOL)
WRITE (LU_GEOM) (MATL (I) , I=1, N_VOL)
ENDIF

```

- ONE has the value 1.
- VERSION currently has value 2 and indicates the version number of this file format.
- N_FLOATS, N_INTS, FIRST_FRAME_STATIC The number of floating point and integer data items stored at the beginning of the file. FIRST_FRAME_STATIC for future use is currently set to 1.
- FLOAT_HEADER, INT_HEADER Floating point and integer data stored at the beginning of the file.
- STIME is the FDS initial simulation time.
- N_VERT, N_FACE, N_VOL are the number of vertices, faces and volumes.
- Xvert, Yvert, Zvert are the vertex coordinates.
- FACE1, FACE2, FACE3 are the vertex indices for each face (triangle). The indices range from 1 to N_VERT.
- SURF_IND are the SURF indices for each face.
- Xtext, Ytext are the texture coordinates.
- VOL1, VOL2, VOL3, VOL4 are the vertex indices for each volume (tetrahedron). The indices are numbered from 1 to N_VERT. Not used currently.
- MATL are the MATL indices for each volume. Not used by FDS currently.

The .ge2 geometry files contain the actual geometry index as read in the .fds input file, written as:

```

IF (N_FACE>0) THEN
  WRITE (LU_GEOM2) N_FACE
  WRITE (LU_GEOM2) (GEOM_IDS (I) , I=1, N_FACES)
ENDIF

```

This information allows for complete area and boundary condition information per geometry in SMV.

27.21 FDS HVAC I/O binary format

The HVAC outputs under the namelist group HVAC are written out unformatted to the file CHID.hvac. These files are written out from dump.f90 with the following lines:

```

WRITE (LU_HVAC) N_NODE_OUT, N_NODE_QUANTITY, N_DUCT_OUT, N_DUCT_QUANTITY
WRITE (LU_HVAC) DUCT_CELL (1:N_DUCT_OUT)

WRITE (LU_HVAC) TIME
DO N=1, N_NODE_OUT
  WRITE (LU_HVAC) OUTVAL_N (1:N_NODE_QUANTITY)
ENDDO
DO N=1, N_DUCT_OUT
  IF (DUCT (N) %N_CELLS > 0) THEN

```

```

      DO N=1, DUCT(N) %N_CELLS
        WRITE (LU_HVAC) OUTVAL_D(1:N_DUCT_QUANTITY)
      ENDDO
    ELSE
      WRITE (LU_HVAC) OUTVAL_D(1:N_DUCT_QUANTITY)
    ENDIF
  ENDDO

```

where `DUCT_CELL` is the number of mass transport cells for a duct (if a duct doesn't have mass transport, then the value 1 is written), and the second block of output is written for each `DT_HVAC` output interval.

27.22 File Extension Glossary

Table [27.1](#) provides a quick reference to file extensions used by FDS and Smokeview.

Table 27.1: File Extension Reference Table

File extension	Short Description	Reference
.be	Data values for .gcf, .gsf, .iso files	Section 27.17
.bf	Boundary files	Section 27.11
.bingeom	Binary geometry files	Section 27.19
.bnd	Min/Max Bound files	Smokeview User's Guide [2]
.fds	FDS input files	FDS User's Guide Part II
.gbf	Geometry boundary files	To be deprecated
.gcf	CFACE geometry files	Section 27.16
.ge	Geometry vertices and faces	Section 27.20
.ge2	Geometry face properties	Section 27.20
.gsf	Unstructured slice files	Section 27.16
.ini	Smokeview configuration files	Smokeview User's Guide [2]
.iso	Isosurface files	Section 27.15
.out	FDS output log files	Section 27.1
.prt5.gbnd	Global bounds of all particle files	
.prt5	Particle files	Section 27.12
.q	Plot3D files	Section 27.10
.restart	FDS restart files	Section 5.6
.s3d	3D smoke files	Section 27.14
.sf	Slice files	Section 27.9
.sinfo	Slice parameter files	Smokeview User's Guide [2]
.smv	Smokeview files	Smokeview User's Guide [2]
.ssf	Smokeview script files	Smokeview User's Guide [2]
.stop	If present, stops a simulation gracefully	Section 5.6
.sz	Size files for Smokeview memory allocation	
.ter	Terrain data files	Section 27.18
.xyz	Plot3D mesh files	Section 27.10
_devc.csv	Device files	Section 27.3
_git.txt	Git revision	
_hrr.csv	Heat Release Rate files	Section 27.2

Bibliography

- [1] K. McGrattan, S. Hostikka, R. McDermott, J. Floyd, C. Weinschenk, and K. Overholt. *Fire Dynamics Simulator, Technical Reference Guide*. National Institute of Standards and Technology, Gaithersburg, Maryland, USA, and VTT Technical Research Centre of Finland, Espoo, Finland, sixth edition, September 2013. Vol. 1: Mathematical Model; Vol. 2: Verification Guide; Vol. 3: Validation Guide; Vol. 4: Software Quality Assurance. v, 63, 80, 109, 164, 181, 325
- [2] G.P. Forney. *Smokeyview, A Tool for Visualizing Fire Dynamics Simulation Data, Volume I: User's Guide*. National Institute of Standards and Technology, Gaithersburg, Maryland, USA, sixth edition, May 2013. v, 3, 7, 312, 350, 419, 428, 439, 483
- [3] K. McGrattan, S. Hostikka, R. McDermott, J. Floyd, C. Weinschenk, and K. Overholt. *Fire Dynamics Simulator, Technical Reference Guide, Volume 1: Mathematical Model*. National Institute of Standards and Technology, Gaithersburg, Maryland, USA, and VTT Technical Research Centre of Finland, Espoo, Finland, sixth edition, September 2013. 3, 84, 103, 129, 131, 179, 195, 196, 197, 199, 206, 212, 221, 271, 291, 322, 324, 327, 336, 384, 395, 406, 409, 449
- [4] K. McGrattan, S. Hostikka, R. McDermott, J. Floyd, C. Weinschenk, and K. Overholt. *Fire Dynamics Simulator, Technical Reference Guide, Volume 2: Verification*. National Institute of Standards and Technology, Gaithersburg, Maryland, USA, and VTT Technical Research Centre of Finland, Espoo, Finland, sixth edition, September 2013. 3, 8, 120, 345
- [5] K. McGrattan, S. Hostikka, R. McDermott, J. Floyd, C. Weinschenk, and K. Overholt. *Fire Dynamics Simulator, Technical Reference Guide, Volume 3: Validation*. National Institute of Standards and Technology, Gaithersburg, Maryland, USA, and VTT Technical Research Centre of Finland, Espoo, Finland, sixth edition, September 2013. 3, 46, 158
- [6] W. Grosshandler. *RadCal: A Narrow Band Model for Radiation Calculations in a Combustion Environment*. NIST Technical Note 1402, National Institute of Standards and Technology, Gaithersburg, Maryland, 1993. 4, 167
- [7] B. Chapman, G. Jost, and R. van der Pas. *Using OpenMP – Portable Shared Memory Parallel Programming*. MIT Press, Cambridge, Massachusetts, 2007. 4, 10
- [8] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI – Portable Parallel Programming with the Message-Passing Interface*. MIT Press, Cambridge, Massachusetts, 2nd edition, 1999. 4, 10
- [9] T.L. Bergman, A.S. Lavine, F.P. Incropera, and D.P. DeWitt. *Fundamentals of Heat and Mass Transfer*. John Wiley and Sons, New York, 7th edition, 2011. 78, 79, 494, 495
- [10] H.S. Mickely, R.C. Ross, and A.L. Squyers. *Heat, Mass, and Momentum Transfer for Flow Over a Flat Plate with Blowing or Suction*. Technical Note 3208, National Advisory Committee for Aeronautics, Washington, DC, July 1954. 80

- [11] R. Taylor and R. Krishna. *Multicomponent Mass Transfer*. Wiley-Interscience, 1993. 80
- [12] N. Jarrin. *Synthetic Inflow Boundary Conditions for the Numerical Simulation of Turbulence*. PhD thesis, The University of Manchester, Manchester M60 1QD, United Kingdom, 2008. 129, 130
- [13] B. Ralph and R. Carvel. Coupled hybrid modelling in fire safety engineering; a literature review. *Fire Safety Journal*, 100:157 – 170, 2018. 132
- [14] F. Bisetti, G. Blanquart, and H. Pitsch. Direct numerical simulation of soot formation in turbulent non-premixed flames. Technical report, Stanford Center for Turbulence Research, 2008. 165
- [15] Sebastian Ferreyro Fernandez. *ADVANCED SOOT AND RADIATION MODELS FOR LAMINAR AND TURBULENT FLAMES*. PhD thesis, The Pennsylvania State University, 2018. 165
- [16] D. Purser and J. Purser. HCN Yields and Fate of Fuel Nitrogen for Materials under Different Combustion Conditions in the ISO 19700 Tube Furnace and Large-scale Fires. In *Fire Safety Science – Proceedings of the Ninth International Symposium*, pages 1117–1128. International Association of Fire Safety Science, 2008. 177
- [17] C. Beyler. *SFPE Handbook of Fire Protection Engineering*, chapter Flammability Limits of Premixed and Diffusion Flames. Springer, New York, 5th edition, 2016. 179, 181, 182
- [18] M.M. Khan, A. Tewarson, and M. Chaos. *SFPE Handbook of Fire Protection Engineering*, chapter Combustion Characteristics of Materials and Generation of Fire Products. Springer, New York, 5th edition, 2016. 185, 186, 188, 199, 203, 204
- [19] C.K. Westbrook and F.L. Dryer. Simplified Reaction Mechanisms for the Oxidation of Hydrocarbon Fuels in Flames. *Combustion Science and Technology*, 27:31–43, 1981. 192, 193
- [20] J. Andersen, C.L. Rasmussen, T. Giselsson, and P. Glarborg. Global Combustion Mechanisms for Use in CFD Modeling Under Oxy-Fuel Conditions. *Energy & Fuels*, 23(3):1379–1389, 2009. 193
- [21] G. P. Smith, D. M. Golden, M. Frenklach, N. W. Moriarty, B. Eiteneer, M. Goldenberg, C. T. Bowman, R. K. Hanson, S. Song, W. C. Gardiner Jr., V. V. Lissianski, and Z. Qin. Gri-mech 3.0. http://www.me.berkeley.edu/gri_mech/, 1999. 194, 195
- [22] R. J. Kee, F. M. Rupley, and J. A. Miller. CHEMKIN-II: A Fortran chemical kinetics package for the analysis of gas-phase chemical kinetics. Technical Report SAND89-8009, Sandia National Laboratories, 1989. 194
- [23] R. J. Kee, F. M. Rupley, J. A. Miller, M. E. Coltrin, J. F. Grcar, E. Meeks, H. K. Moffat, A. E. Lutz, G. DixonLewis, M. D. Smooke, J. Warnatz, G. H. Evans, R. S. Larson, R. E. Mitchell, L. R. Petzold, W. C. Reynolds, M. Caracotsios, W. E. Stewart, P. Glarborg, C. Wang, and O. Adigun. CHEMKIN Collection Release 3.6, The Chemkin Thermodynamic Database. Technical report, Reaction Design, Inc., San Diego, CA, 2000. 194
- [24] D. G. Goodwin, H. K. Moffat, I. Schoegl, R. L. Speth, and B. W. Weber. Cantera: An object-oriented software toolkit for chemical kinetics, thermodynamics, and transport processes. <https://www.cantera.org>, 2023. Version 3.0.0. 194
- [25] A. C. Hindmarsh, R. Serban, C. J. Balos, D. J. Gardner, D. R. Reynolds, and C. S. Woodward. User documentation for ccode. <https://sundials.readthedocs.io/en/latest/ccode/index.html>, 2023. v6.7.0. 194, 197, 469

- [26] W. P. Jones and R. P. Lindstedt. Global reaction schemes for hydrocarbon combustion. *Combustion and Flame*, 73:233–249, 1988. [195](#)
- [27] J.R. Hartman, A.P. Beyler, S. Riahi, and C.L. Beyer. Smoke oxidation kinetics for application to prediction of clean burn patterns. *Fire and Materials*, 36:177–184, 2012. [200](#)
- [28] C.L. Beyer. *SFPE Handbook of Fire Protection Engineering*, chapter Fire Hazard Calculations for Large Open Hydrocarbon Fires. Springer, New York, 5th edition, 2016. [203](#), [204](#)
- [29] R. Buch, A. Hamins, K. Konishi, D. Mattingly, and T. Kashiwagi. Radiative emission fraction of pool fires burning silicone fluids. *Combustion and flame*, 108(1-2):118–126, 1997. [204](#)
- [30] Y.J. Kim and A. Trouvé. Evaluation of angular resolution requirements in the finite-volume method solution of the radiative transfer equation. *Fire Safety Journal*, 141:103971, 2023. [206](#)
- [31] A. Gupta, K. Meredith, G. Agarwal, S. Thumularu, Y. Xin, M. Chaos, and Y. Wang. CFD Modeling of Fire Growth in Rack-Storages of Cartoned Unexpanded Plastic (CUP) Commodity. FM Global Open Source CFD Fire Modeling Workshop, 2015. [208](#)
- [32] W. E. Ranz and W. R. Marshall. Evaporation from drops - Part II. *Chemical Engineering Progress*, 48:173–180, March 1952. [213](#)
- [33] S.S. Sazhin. Advanced models of fuel droplet heating and evaporation. *Progress in Energy and Combustion Science*, 32:162–214, 2006. [213](#)
- [34] C. Theobald. The Effect of Nozzle Design on the Stability and Performance of Turbulent Water Jets. *Fire Safety Journal*, 4:1–13, 1981. [218](#)
- [35] T. Bartzanas, T. Boulard, and C. Kittas. Numerical simulation of the airflow and temperature distribution in a tunnel greenhouse equipped with insect-proof screens in the openings. *Computers and Electronics in Agriculture*, 34:207–221, 2002. [225](#)
- [36] P. Andersson and P. Van Hees. Performance of Cables Subjected to Elevated Temperatures. In *Fire Safety Science – Proceedings of the Eighth International Symposium*, pages 1121–1132. International Association of Fire Safety Science, 2005. [225](#)
- [37] S.P. Nowlen, F.J. Wyant, and K.B. McGrattan. Cable Response to Live Fire (CAROLFIRE). NUREG/CR 6931, United States Nuclear Regulatory Commission, Washington, DC, April 2008. [226](#)
- [38] S.R. Hanna, G.A. Briggs, and R.P. Hosker. Handbook on Atmospheric Diffusion. DOE/TIC 11223, U.S. Department of Energy, 1982. Available through the National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia. [240](#)
- [39] T.C. Brown, R.T. Cederwall, S.T. Chan, D.L. Ermak, R.P. Koopman, K.C. Lamson, J.W. McClure, and L.K. Morris. Falcon Series Data Report: 1987 LNG Vapor Barrier Verification Field Trials. Data Report GRI-89/0138, Lawrence Livermore National Laboratory, June 1990. [241](#)
- [40] A.J. Dyer. A review of flux profile relationships. *Boundary-Layer Meteorology*, 7:363–372, 1974. [241](#)
- [41] R.B. Stull. *Meteorology for Scientists and Engineers*. Brooks/Cole, Pacific Grove, California, 2nd edition, 2000. [242](#), [245](#), [246](#)

- [42] S. Basu and A. Lacser. A Cautionary Note on the Use of Monin-Obukhov Similarity Theory in Very High Resolution Large-Eddy Simulations. *Boudary-Layer Meteorol*, 163:351–355, 2017. [242](#)
- [43] J.H. Klote and J.A. Milke. *Design of Smoke Management Systems*. American Society of Heating Refrigeration, and Air-Conditioning Engineers and Society of Fire Protection Engineers, Atlanta, Georgia, 1992. [251](#)
- [44] B. Porterie, J.L. Consalvi, A. Kaiss, and J.C. Loraud. Predicting Wildland Fire Behavior and Emissions Using a Fine-Scale Physical Model. *Numerical Heat Transfer, Part A*, 47:571–591, 2005. [255](#), [256](#), [258](#)
- [45] D. Morvan and J.L. Dupuy. Modeling the propagation of a wildfire through a Mediterranean shrub using a multiphase formulation. *Combustion and Flame*, 138(3):199–210, 2004. [255](#), [256](#)
- [46] M. Houssami, E. Mueller, A. Filkov, J.C. Thomas, N. Skowronski, M.R. Gallagher, K. Clark, R. Kremens, and A. Simeoni. Experimental procedures characterizing firebrand generation in wildland fires. *Fire Technology*, 52(3):731–751, 2016. [255](#), [256](#)
- [47] S. Ma, F. He2, D. Tian, D. Zou, Z. Yan, Y. Yang, T. Zhou, K. Huang, H. Shen5, and J. Fang. Variations and determinants of carbon content in plants: a global synthesis. *Biogeosciences*, 15:693–702, 2018. <https://doi.org/10.5194/bg-15-693-2018>. [256](#)
- [48] S.J. Ritchie, K.D. Steckler, A. Hamins, T.G. Cleary, J.C. Yang, and T. Kashiwagi. The Effect of Sample Size on the Heat Release Rate of Charring Materials. In *Fire Safety Science - Proceedings of the 5th International Symposium*, pages 177–188. International Association For Fire Safety Science, 1997. [256](#), [258](#)
- [49] W. Mell, A. Maranghides, R. McDermott, and S. Manzello. Numerical simulation and experiments of burning Douglas fir trees. *Combustion and Flame*, 156:2023–2041, 2009. [256](#)
- [50] N. Boonmee and J.G. Quintiere. Glowing ignition of wood: the onset of surface combustion. *Proceedings of the Combustion Institute*, 30:2303–2310, 2005. [257](#), [258](#)
- [51] S.R. Turns. *An Introduction to Combustion*. McGraw-Hill, New York, 2nd edition, 1996. [257](#)
- [52] A.M. Grishin. *Mathematical modeling of forest fires and new methods of fighting them*. Publishing House of the Tomsk State University, 1997. [258](#)
- [53] T. Kashiwagi and H. Nambu. Global kinetic constants for thermal oxidative degradation of a cellulosic paper. *Combustion and Flame*, 88:345–368, 1992. [258](#)
- [54] C.R. Boardman, M.A. Dietenberger, and D.R. Weise. Specific heat capacity of wildland foliar fuels to 434 °C. *Fuel*, 292:120396, 2021. <https://doi.org/10.1016/j.fuel.2021.120396>. [257](#)
- [55] V. Tihay, F. Morandini, P. Santoni, Y. Perez-Ramirez, and T. Barboni. Combustion of forest litters under slope conditions: Burning rate, heat release rate, convective and radiant fractions for different loads. *Combustion and Flame*, 161:3237–3248, 2014. [260](#)
- [56] R. Falkenstein-Smith, K. McGrattan, B. Toman, and M. Fernandez. Measurement of the Flow Resistance of Vegetation. NIST Technical Note 2039, National Institute of Standards and Technology, Gaithersburg, Maryland, March 2019. <https://doi.org/10.6028/NIST.TN.2039>. [262](#)

- [57] S.L. Manzello, J.R. Shields, T.G. Cleary, A. Maranghides, W.E. Mell, J.C. Yang, Y. Hayashi, D. Nii, and T. Kurita. On the development and characterization of a firebrand generator. *Fire Safety Journal*, 43:258–268, 2008. 266
- [58] Y. Perez-Ramirez, P.A. Santoni, J.B. Tramoni, F. Bosseur, and W.E. Mell. Examination of WFDS in Modeling Spreading Fires in a Furniture Calorimeter. *Fire Technology*, 53:1795–1832, 2017. 268
- [59] R. Siegel and J. R. Howell. *Thermal Radiation Heat Transfer*. Taylor & Francis, New York, 4th edition, 2002. 271
- [60] A.S. Bova, W.E. Mell, and C.M. Hoffman. A comparison of level set and marker methods for the simulation of wildland fire front propagation. *International Journal of Wildland Fire*, 25(2):229–241, 2015. <http://dx.doi.org/10.1071/WF13178>. 273, 274, 277
- [61] M.A. Finney. Fire Area Simulator – Model, Development and Evaluation. Research Paper RMRS-RP-4 Revised, United States Forest Service, Rocky Mountain Research Station, Missoula, Montana, February 2004. http://www.fs.fed.us/rm/pubs/rmrs_rp004.pdf. 273
- [62] R.C. Rothermel. A Mathematical Model for Predicting Fire Spread in Wildland Fuels. Research Paper INT-115, Intermountain Forest and Range Experiment Station, USDA Forest Service, Ogden, Utah, January 1972. <http://www.treesearch.fs.fed.us/pubs/32533>. 273, 274, 275
- [63] F.A. Albini. Estimating Wildfire Behavior and Effects. Research Paper INT-30, Intermountain Forest and Range Experiment Station, USDA Forest Service, Ogden, Utah, 1976. https://www.fs.fed.us/rm/pubs_int/int_gtr030.pdf. 273, 274, 275
- [64] G.D. Richards. An elliptical growth model of forest fire fronts and its numerical solution. *International Journal for Numerical Methods in Engineering*, 30:1163–1179, 1990. 273
- [65] G. Heskestad and R.G. Bill. Quantification of Thermal Responsiveness of Automatic Sprinklers Including Conduction Effects. *Fire Safety Journal*, 14:113–125, 1988. 285
- [66] T. Cleary, A. Chernovsky, W. Grosshandler, and M. Anderson. Particulate Entry Lag in Spot-Type Smoke Detectors. In *Fire Safety Science – Proceedings of the Sixth International Symposium*, pages 779–790. International Association for Fire Safety Science, 1999. 292
- [67] M.J. Hurley, editor. *SFPE Handbook of Fire Protection Engineering*. Springer, New York, 5th edition, 2016. 292
- [68] J.W. Deardorff. Stratocumulus-capped mixed layers derived from a three-dimensional model. *Boundary-Layer Meteorol.*, 18:495–527, 1980. 322
- [69] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000. 322, 397, 398
- [70] J. Smagorinsky. General Circulation Experiments with the Primitive Equations. I. The Basic Experiment. *Monthly Weather Review*, 91(3):99–164, March 1963. 322
- [71] M. Germano, U. Piomelli, P. Moin, and W. Cabot. A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A*, 3(7):1760–1765, 1991. 322
- [72] P. Moin, K. Squires, W. Cabot, and S. Lee. A dynamic subgrid-scale model for compressible turbulence and scalar transport. *Phys. Fluids A*, 3(11):2746–2757, 1991. 322

- [73] B. Vreman. An eddy-viscosity subgrid-scale model for turbulent shear flow: Algebraic theory and applications. *Physics of Fluids*, 16(10):3670–3681, 2004. [322](#)
- [74] F. Nicoud and F. Ducros. Subgrid-Scale Stress Modelling Based on the Square of the Velocity Gradient Tensor. *Flow, Turbulence, and Combustion*, 62:183–200, 1999. [322](#)
- [75] D.C. Wilcox. *Turbulence Modeling for CFD*. DCW Industries, Inc., 2nd edition, 1998. [322](#)
- [76] P.L. Roe. Characteristics-based schemes for the euler equations. *Ann. Rev. Fluid Mech.*, 18:337, 1986. [325](#)
- [77] G. Zhou. *Numerical simulations of physical discontinuities in single and multi-fluid flows for arbitrary Mach numbers*. PhD thesis, Chalmers Univ. of Tech., Goteborg, Sweden, 1995. [325](#)
- [78] HYPRE: Scalable Linear Solvers and Multigrid Methods. [HYPRE Website](#). Lawrence Livermore National Laboratories. [336](#), [337](#), [469](#)
- [79] Y. Xin. Baroclinic Effects on Fire Flow Field. In *Proceedings of the Fourth Joint Meeting of the U.S. Sections of the Combustion Institute*. Combustion Institute, Pittsburgh, Pennsylvania, March 2005. [342](#)
- [80] Bruce R. Munson, Donald F. Young, and Theodore H. Okiishi. *Fundamentals of Fluid Mechanics*. John Wiley and Sons, 1990. [345](#)
- [81] R. I. Harris. Gumbel re-visited—a new look at extreme value statistics applied to wind speeds. *Journal of Wind Engineering and Industrial Aerodynamics*, 59:1–22, 1996. [358](#)
- [82] Pamela P. Walatka and Pieter G. Buning. PLOT3D User’s Manual, version 3.5. NASA Technical Memorandum 101067, NASA, 1989. [366](#)
- [83] G.W. Mulholland. *SFPE Handbook of Fire Protection Engineering*, chapter Smoke Production and Properties. National Fire Protection Association, Quincy, Massachusetts, 3rd edition, 2002. [373](#), [374](#)
- [84] G.W. Mulholland and C. Croarkin. Specific Extinction Coefficient of Flame Generated Smoke. *Fire and Materials*, 24:227–230, 2000. [374](#)
- [85] M.L. Janssens and H.C. Tran. Data Reduction of Room Tests for Zone Model Validation. *Journal of Fire Science*, 10:528–555, 1992. [375](#)
- [86] Y.P. He, A. Fernando, and M.C. Luo. Determination of interface height from measured parameter profile in enclosure fire experiment. *Fire Safety Journal*, 31:19–38, 1998. [376](#)
- [87] S. Welsh and P. Rubini. Three-dimensional Simulation of a Fire-Resistance Furnace. In *Fire Safety Science – Proceedings of the Fifth International Symposium*. International Association for Fire Safety Science, 1997. [376](#)
- [88] A.V. Murthy, I. Wetterlund, and D.P. DeWitt. Characterization of an Ellipsoidal Radiometer. *Journal of Research of the National Institute of Standards and Technology*, 108(2):115–124, March-April 2003. [381](#)
- [89] U. Wickström, D. Duthinh, and K.B. McGrattan. Adiabatic Surface Temperature for Calculating Heat Transfer to Fire Exposed Structures. In *Proceedings of the Eleventh International Interflam Conference*. Interscience Communications, London, 2007. [382](#)

- [90] M. Malendowski. Analytical Solution for Adiabatic Surface Temperature (AST). *Fire Technology*, 53:413–420, 2017. [382](#)
- [91] D.A. Purser. *SFPE Handbook of Fire Protection Engineering*, chapter Combustion Toxicity. Springer, New York, 5th edition, 2016. [389](#), [390](#)
- [92] J.G. Slowik, K. Stainken, P. Davidovits, L.R. Williams, J.T. Jayne, C.E. Kolb, D.R. Worsnop, Y. Rudich, P.F. DeCarlo, and J.L. Jimenez. Particle Morphology and Density Characterization by Combined Mobility and Aerodynamic Diameter Measurements. Part 2: Application to Combustion-Generated Soot Aerosols as a Function of Fuel Equivalence Ratio. *Aerosol Science and Technology*, 38:1206–1222, 2004. [393](#)
- [93] S.B. Pope. Ten questions concerning the large-eddy simulation of turbulent flows. *New Journal of Physics*, 6:1–24, 2004. [394](#)
- [94] R. McDermott, G. Forney, K. McGrattan, and W. Mell. Fire Dynamics Simulator 6: Complex Geometry, Embedded Meshes, and Quality Assessment. In J.C.F. Pereira and A. Sequeira, editors, *V European Conference on Computational Fluid Dynamics*, Lisbon, Portugal, 2010. ECCOMAS. [395](#), [397](#)
- [95] Y. Nievergelt. *Wavelets Made Easy*. Birkhäuser, 1999. [395](#)
- [96] K. Schneider and O. Vasilyev. Wavelet methods in computational fluid dynamics. *Annu. Rev. Fluid Mech.*, 42:473–503, 2010. [395](#)
- [97] B. McBride, S. Gordon, and M. Reno. Coefficients for Calculating Thermodynamic and Transport Properties of Individual Species. NASA Technical Memorandum 4513, National Aeronautics and Space Administration, Cleveland, OH, 1993. [494](#), [495](#)
- [98] B. McBride, M. Zehe, and S. Gordon. NASA Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species. NASA/TP 2002-211556, National Aeronautics and Space Administration, Cleveland, OH, September 2002. [494](#), [495](#)
- [99] R. Svhela. Estimated Viscosities and Thermal Conductivity of Gases at High Temperatures. NASA Technical Report R-132, National Aeronautics and Space Administration, Cleveland, OH, 1962. [494](#), [495](#)
- [100] M.W. Chase. *NIST-JANAF Thermochemical Tables*. Journal of Physical and Chemical Reference Data, Monograph No. 9. American Chemical Society, Woodbury, New York, 4th edition, 1998. [494](#), [495](#)
- [101] A. Faghri and Y. Zhang. *Transport Phenomena in Multiphase Systems*. Academic Press, Cambridge, MA, 1st edition, 2006. [494](#), [495](#)
- [102] J.E. Hardy, J.O. Hylton, and T.E. McKnight. Empirical Correlations for Thermal Flowmeters Covering a Wide Range of Thermal-Physical Properties. In *National Conference of Standards labs (NCSL) 1999 Workshop and Symposium*, 1999. Charlotte, North Carolina. [494](#), [495](#)
- [103] I. Martiez. *Termodinamica Basica y Aplicada*. Dossat, Editorial S.A., Madrid, Spain, 1992. [494](#), [495](#)

Appendix A

Predefined Species

This appendix lists all predefined species in FDS. If a species name ends in 'p' or 'm', this indicates the positive or negative charge state of that species. If a species ends in 's', this indicates the excited state of that species. This is typically written with a '*' as in CH*. A 'Y' in the Liquid column indicates that liquid properties are predefined, and a 'Y' in the Gibbs column indicates that Gibbs energy data is predefined. Two tables are given, one containing species commonly of interest in fire, common toxic species, and reference species; and a second table containing all species.

Table A.1: Pre-defined gas and liquid species [97, 98, 99, 100, 9, 101, 102, 103]

Species	Chemical Formula	Temp. Range (K)	ΔH_f (kJ/mol)	σ (Å)	ϵ/k (K)	Liquid	Gibbs	Pr	RadCal Surrogate
LJ AIR	Air	200 -20000	-0.13	3.711	78.60		Y	0.71	
ARGON	Ar	200 -20000	0.00	3.542	93.30	Y	Y	0.67	
SOOT	C	200 -6000	0.00	3.798	71.40		Y	0.71	SOOT
JP-10	C10H16	200 -6000	-86.86	4.892	231.60		Y	0.71	N-HEPTANE
N-DECANE	C10H22	200 -1500	-249.70	4.892	231.60	Y		0.71	N-HEPTANE
NAPHTHALENE	C10H8	200 -6000	150.58	5.698	480.00		Y	0.71	TOLUENE
JET-A	C12H23	273 -6000	-249.66	4.892	231.60		Y	0.71	N-HEPTANE
DODECANE	C12H26	288 -1500	-187.80	4.892	231.60	Y		0.71	N-HEPTANE
ACETYLENE	C2H2	200 -6000	228.20	4.033	231.80		Y	0.78	ETHYLENE
ACROLEIN	C2H3CHO	298 -3000	-74.47	5.176	357.00	Y	Y	0.71	MMA
ETHYLENE	C2H4	200 -6000	52.50	4.163	224.70	Y	Y	0.83	ETHYLENE
ETHANOL	C2H5OH	200 -6000	-234.95	4.530	362.60	Y	Y	0.84	METHANOL
ETHANE	C2H6	200 -6000	-83.85	4.443	215.70	Y	Y	0.84	ETHANE
PROPYLENE	C3H6	90 -6000	20.00	4.678	289.90	Y	Y	0.82	PROPYLENE
ACETONE	C3H6O	200 -6000	-217.15	4.600	560.20	Y	Y	0.87	MMA
PROPANE	C3H8	90 -6000	-104.68	5.118	237.10	Y	Y	0.73	PROPANE
ISOPROPANOL	C3H8O	190 -6000	-272.70	4.549	576.70	Y	Y	0.71	METHANOL
PROPANOL	C3H8O	200 -6000	-255.20	4.549	576.70		Y	0.71	METHANOL
ISOBUTANE	C4H10	200 -6000	-134.99	5.287	330.10		Y	0.83	PROPANE
BUTANE	C4H10	200 -6000	-125.79	4.687	531.40	Y	Y	0.83	PROPANE
N-PENTANE	C5H12	200 -6000	-146.76	5.784	341.10	Y	Y	0.79	N-HEPTANE
CYCLOPENTENE	C5H6	200 -6000	134.30	5.180	357.00		Y	0.71	
CYCLOHEXANE	C6H12	200 -6000	-123.30	6.182	297.10		Y	0.71	
N-HEXANE	C6H14	200 -6000	-166.92	5.949	399.30	Y	Y	0.79	N-HEPTANE
BENZENE	C6H6	200 -6000	82.88	5.439	412.30		Y	1.50	TOLUENE
N-HEPTANE	C7H16	200 -6000	-187.78	4.701	205.78	Y	Y	0.83	N-HEPTANE
TOLUENE	C7H8	178 -6000	50.17	5.698	480.00	Y	Y	0.71	TOLUENE
N-OCTANE	C8H18	200 -6000	-208.75	4.892	231.60	Y	Y	0.64	N-HEPTANE
DIFLUOROMETHANE	CH2F2	200 -6000	-452.30	4.080	318.00		Y	0.71	
CHLOROFLUORMETHANE	CH2FC1	200 -6000	-265.70	4.480	318.00		Y	0.71	
METHYL ETHER	CH3OCH3	200 -6000	-184.11	4.307	395.00		Y	0.72	METHANOL
METHANOL	CH3OH	200 -6000	-200.94	3.626	481.80		Y	0.95	METHANOL
METHANE	CH4	200 -6000	-74.60	3.758	148.60	Y	Y	0.70	METHANE
CHLOROFORM	CHCl3	200 -6000	-102.70	5.389	340.20		Y	0.73	
DIFLUOROCHLOROMETHANE	CHF2C1	200 -6000	-482.80	4.680	261.00		Y	0.72	
FLUOROFORM	CHF3	200 -6000	-693.30	4.330	240.00		Y	0.72	
CHLORINE	Cl2	200 -6000	468.39	4.217	316.00	Y	Y	0.75	
CARBON MONOXIDE	CO	200 -20000	-110.54	3.690	91.70	Y	Y	0.73	CARBON MONOXIDE
CARBON DIOXIDE	CO2	200 -20000	-393.51	3.941	195.20		Y	0.75	CARBON DIOXIDE
HYDROGEN	H2	200 -20000	0.00	2.827	59.70	Y	Y	0.69	
WATER VAPOR	H2O	273 -600	-241.83	2.641	809.10	Y	Y	1.00	WATER VAPOR
HYDROGEN PEROXIDE	H2O2	200 -6000	-135.88	4.196	289.30	Y	Y	0.71	
HYDROGEN SULFIDE	H2S	200 -6000	-20.60	3.623	301.10	Y	Y	0.70	

Table A.1: Optional gas and liquid species (continued).

Species	Formula	Temp. Range (K)	ΔH_f (kJ/mol)	σ (Å)	ϵ/k (K)	Liquid	Gibbs	Pr	RadCal Surrogate
FORMALDEHYDE	HCHO	200 -6000	-108.58	3.590	498.00	Y	Y	0.71	METHANOL
HYDROGEN CHLORIDE	HCl	200 -6000	-92.31	3.339	344.70	Y	Y	0.75	
HYDROGEN CYANIDE	HCN	200 -6000	133.08	3.638	569.40	Y	Y	0.70	
HELIUM	He	200 -20000	0.00	2.551	10.22	Y	Y	0.68	
HYDROGEN FLUORIDE	HF	200 -20000	-273.30	3.148	330.00	Y	Y	0.71	
MERCURY	Hg	200 -20000	61.38	2.969	750.00		Y	0.67	
MERCURY p	Hg	298 -20000	1074.64	2.969	750.00		Y	0.67	
HYDROGEN IODIDE	HI	200 -6000	26.36	4.211	288.70		Y	0.69	
IODINE	I	200 -20000	106.76	4.320	210.70		Y	0.67	
NITROGEN	N2	200 -20000	0.00	3.798	71.40	Y	Y	0.71	
NITROUS OXIDE	N2O	182 -6000	81.60	3.828	232.40	Y	Y	0.74	
NEON	Ne	200 -20000	0.00	2.820	32.80		Y	0.67	
AMMONIA	NH3	200 -6000	-45.94	2.900	558.30	Y	Y	0.87	
NITRIC OXIDE	NO	110 -20000	91.27	3.492	116.70	Y	Y	0.74	
NITROGEN DIOXIDE	NO2	290 -6000	34.19	3.922	204.88	Y	Y	6.10	
OXYGEN	O2	60 -20000	0.00	3.467	106.70	Y	Y	0.71	
OZONE	O3	200 -6000	141.80	3.875	208.40		Y	1.48	
SULFUR HEXAFLUORIDE	SF6	200 -6000	-1219.40	5.128	222.10		Y	0.77	
SULFUR DIOXIDE	SO2	200 -6000	-296.81	4.112	335.40	Y	Y	0.91	
URANIUM HEXAFLUORIDE	UF6	200 -20000	-2148.64	5.967	236.80		Y	0.71	
XENON	Xe	300 -20000	0.00	4.047	231.00		Y	0.67	
ISOCTANE	C8H18	200 -6000	-224.01	4.892	231.60	Y	Y	0.64	N-HEPTANE

Table A.2: Pre-defined gas and liquid species [97, 98, 99, 100, 9, 101, 102, 103]

Species	Chemical Formula	Temp. Range (K)	ΔH_f (kJ/mol)	σ (Å)	ϵ/k (K)	Liquid	Gibbs	Pr	RadCal Surrogate
SODIUM HYDROXIDE2	(NaOH)2	298 -4000	-607.66	4.840	1279.00		Y	0.71	
LJ AIR	Air	200 -20000	-0.13	3.711	78.60		Y	0.71	
ALUMINUM	Al	200 -20000	330.00	2.655	2750.00		Y	0.67	
ALUMINUMm	Al	298 -20000	281.09	2.655	2750.00		Y	0.67	
ALUMINUMp	Al	298 -20000	913.02	2.655	2750.00		Y	0.67	
DIALUMINUM	Al2	200 -6000	501.30	2.940	2750.00		Y	0.70	
ALUMINUM OXIDE	Al2O3	200 -6000	-546.89	3.186	557.49		Y	1.00	SOOT
ALUMINUM MONOCHLORIDE	AlCl	200 -6000	-51.01	3.578	932.00		Y	0.70	
ALUMINUM MONOCHLORIDEp	AlCl	298 -6000	861.85	3.578	932.00		Y	0.70	
ALUMINUM CHLORIDE	AlCl3	200 -6000	-584.68	5.127	472.00		Y	0.73	
ALUMINUM MONOFLUORIDE	AlF	200 -6000	-264.06	3.148	556.00		Y	0.70	
ALUMINUM MONOFLUORIDEp	AlF	298 -6000	692.23	3.148	556.00		Y	0.70	
ALUMINUM FLUORIDE	AlF3	200 -6000	-1209.28	4.198	1846.00		Y	0.73	
ALUMINUM NITRIDE	AlN	200 -6000	438.83	3.369	2682.00		Y	0.70	

Table A.2: Optional gas and liquid species (continued).

Species	Formula	Temp. Range (K)	ΔH_f (kJ/mol)	σ (Å)	ϵ/k (K)	Liquid	Gibbs	Pr	RadCal Surrogate
ALUMINUM MONOXIDE	AlO	200 -20000	67.32	3.204	542.00		Y	0.69	
ALUMINUM SULFIDE	AlS	200 -6000	232.68	3.730	1526.00		Y	0.70	
ARGON	Ar	200 -20000	0.00	3.542	93.30	Y	Y	0.67	
ARGONp	Ar	298 -20000	1526.78	3.542	93.30		Y	0.67	
BORON	B	200 -20000	575.60	2.265	3331.00		Y	0.67	
BORONm	B	298 -20000	542.63	2.265	3331.00		Y	0.67	
BORONp	B	298 -20000	1382.32	2.265	3331.00		Y	0.67	
BORON DIMER	B2	200 -6000	857.37	2.420	3331.00		Y	0.70	
DIBORANE	B2H6	200 -6000	36.60	4.821	213.20		Y	0.72	
DIBORON TRIOXIDE	B2O3	200 -6000	-835.38	4.158	2092.00		Y	0.62	
BORON TRIBORIDE	BBr3	200 -6000	-205.30	5.439	430.00		Y	0.73	
BORON MONOCHLORIDE	BCl	200 -6000	183.17	3.318	10126.00		Y	0.69	
BORON MONOCHLORIDEp	BCl	298 -6000	1234.28	3.318	10126.00		Y	0.69	
DICHLOROBORON	BCl2	200 -6000	-60.88	4.222	682.00		Y	0.71	
DICHLOROBORONp	BCl2	298 -6000	672.32	4.222	682.00		Y	0.71	
BORON CHLORIDE	BCl3	200 -6000	-404.50	5.127	337.70		Y	0.72	
BERYLLIUM ATOM	Be	200 -20000	324.00	2.618	3603.00		Y	0.67	
BERYLLIUM ATOMP	Be	298 -20000	1229.70	2.618	3603.00		Y	0.67	
BERYLLIUM ATOMpp	Be	298 -20000	2993.00	2.618	3603.00		Y	0.67	
BERYLLIUM	Be2	200 -6000	637.54	2.891	2.89		Y	0.71	
BERYLLIUM DIBORIDE	BeBr2	200 -6000	-234.06	4.235	936.00		Y	0.71	
BERYLLIUM MONOCHLORIDE	BeCl	200 -6000	56.69	3.554	1067.00		Y	0.70	
BERYLLIUM CHLORIDE	BeCl2	200 -6000	-361.54	4.190	936.00		Y	0.72	
BERYLLIUM MONOFLUORIDE	BeF	200 -6000	-170.62	3.124	637.00		Y	0.69	
BERYLLIUM FLUORIDE	BeF2	200 -6000	-796.59	3.452	1266.00		Y	0.70	
BERYLLIUM DIODIDE	BeI2	200 -6000	-64.76	4.955	1019.00		Y	0.71	
BORON FLUORIDE	BF	200 -6000	-106.93	2.888	612.00		Y	0.69	
BERYLLIUM FLUORIDE _m	BF2	298 -6000	-733.80	3.452	1266.00		Y	0.68	
BERYLLIUM FLUORIDE _p	BF2	298 -6000	322.59	3.452	1266.00		Y	0.68	
DIFLUOROBORYL RADICAL	BF2	200 -6000	-499.43	3.543	3999.00		Y	0.71	
BORON FLUORIDE	BF3	200 -6000	-1136.00	4.198	186.30		Y	0.72	
BORON MONOXIDE	BO	200 -20000	20.41	2.944	566.00		Y	0.69	
BORON MONOXIDE _m	BO	298 -6000	-277.79	2.944	566.00		Y	0.69	
BROMINE ATOM	Br	200 -20000	111.87	3.672	236.60		Y	0.67	
BROMINE ATOM _m	Br	298 -20000	-219.00	3.672	236.60		Y	0.67	
BROMINE ATOM _p	Br	298 -20000	1257.93	3.672	236.60		Y	0.67	
BROMINE	Br2	200 -6000	30.91	4.296	507.90		Y	0.70	
BROMINE MONOFLUORIDE	BrF	200 -6000	-58.85	3.826	239.00		Y	0.70	
BORON TRIFLUORIDE	BrF3	200 -6000	-255.60	4.366	481.70		Y	0.71	
BORON MONOXIDE	BrO	200 -6000	125.80	3.882	233.00		Y	0.71	
CARBON ATOM	C	200 -20000	716.68	3.385	30.60		Y	0.67	
CARBON ATOM _m	C	298 -20000	588.31	3.385	30.60		Y	0.67	
CARBON ATOM _p	C	298 -20000	1809.44	3.385	30.60		Y	0.67	

Table A.2: Optional gas and liquid species (continued).

Species	Formula	Temp. Range (K)	ΔH_f (kJ/mol)	σ (Å)	ε/k (K)	Liquid	Gibbs	Pr	RadCal Surrogate
SOOT	C	200 -6000	0.00	3.798	71.40		Y	0.71	SOOT
JP-10	C10H16	200 -6000	-86.86	4.892	231.60		Y	0.71	N-HEPTANE
N-DECANE	C10H22	200 -1500	-249.70	4.892	231.60	Y		0.71	N-HEPTANE
NAPHTHALENE	C10H8	200 -6000	150.58	5.698	480.00		Y	0.71	TOLUENE
JET-A	C12H23	273 -6000	-249.66	4.892	231.60		Y	0.71	N-HEPTANE
DODECANE	C12H26	288 -1500	-187.80	4.892	231.60	Y		0.71	N-HEPTANE
DICARBON	C2	200 -20000	830.46	3.913	78.80		Y	0.71	
DICARBONm	C2	298 -6000	480.77	3.913	78.80		Y	0.71	
DICARBONp	C2	298 -20000	2004.78	3.913	78.80		Y	0.71	
ETHYNYL RADICAL	C2H	200 -6000	566.20	4.100	209.00		Y	0.71	
ACETYLENE	C2H2	200 -6000	228.20	4.033	231.80		Y	0.78	ETHYLENE
VINYL RADICAL	C2H3	200 -6000	299.69	4.100	209.00		Y	0.71	
ACROLEIN	C2H3CHO	298 -3000	-74.47	5.176	357.00	Y	Y	0.71	MMA
ETHYLENE	C2H4	200 -6000	52.50	4.163	224.70	Y	Y	0.83	ETHYLENE
ETHYL RADICAL	C2H5	200 -6000	118.66	4.302	252.30		Y	0.71	
ETHANOL	C2H5OH	200 -6000	-234.95	4.530	362.60	Y	Y	0.84	METHANOL
ETHANE	C2H6	200 -6000	-83.85	4.443	215.70	Y	Y	0.84	ETHANE
CYANONGEN	C2N2	200 -6000	309.10	4.361	3486.00		Y	0.72	
CYCLOPROPENYLIDENE	C3H2	298 -5000	25.34	3.970	436.00		Y	0.71	
1-PROPENYL	C3H3	200 -6000	450.00	4.760	252.00		Y	0.71	
CYCLOPROPENE	C3H4	200 -6000	277.10	4.760	252.00		Y	0.71	
ALLENE	C3H4	200 -6000	190.92	4.760	252.00		Y	0.71	
PROPENE	C3H4	200 -6000	184.90	4.761	251.80		Y	0.71	
ALLYL RADICAL	C3H5	200 -6000	163.59	4.982	266.80		Y	0.71	
CYCLOPROPANE	C3H6	200 -6000	53.30	4.807	248.90		Y	0.72	
PROPYLENE	C3H6	90 -6000	20.00	4.678	289.90	Y	Y	0.82	PROPYLENE
ACETONE	C3H6O	200 -6000	-217.15	4.600	560.20	Y	Y	0.87	MMA
PROPYL RADICAL	C3H7	200 -6000	100.50	4.982	266.80		Y	0.71	
ISOPROPYL RADICAL	C3H7	200 -6000	93.30	4.982	266.80		Y	0.71	
PROPANE	C3H8	90 -6000	-104.68	5.118	237.10	Y	Y	0.73	PROPANE
ISOPROPANOL	C3H8O	190 -6000	-272.70	4.549	576.70	Y	Y	0.71	METHANOL
PROPANOL	C3H8O	200 -6000	-255.20	4.549	576.70		Y	0.71	METHANOL
ISOBUTANE	C4H10	200 -6000	-134.99	5.287	330.10		Y	0.83	PROPANE
BUTANE	C4H10	200 -6000	-125.79	4.687	531.40	Y	Y	0.83	PROPANE
BUTADIENE	C4H2	200 -6000	450.00	5.180	357.00		Y	0.71	
i-C4H3	C4H3	298 -3000	498.75	5.180	357.00		Y	0.71	
n-C4H3	C4H3	298 -3000	531.81	5.180	357.00		Y	0.71	
CYCLOBUTADIENE	C4H4	200 -6000	385.00	5.180	357.00		Y	0.71	
i-C4H5	C4H5	298 -3000	323.86	5.180	357.00		Y	0.71	
n-C4H5	C4H5	298 -3000	357.33	5.180	357.00		Y	0.71	
1-BUTYNE	C4H6	200 -6000	165.20	5.180	357.00		Y	0.71	
2-BUTYNE	C4H6	200 -6000	145.70	5.180	357.00		Y	0.71	
BUTADIENE	C4H6	200 -6000	110.00	5.180	357.00		Y	0.71	

Table A.2: Optional gas and liquid species (continued).

Species	Formula	Temp. Range (K)	ΔH_f (kJ/mol)	σ (Å)	ε/k (K)	Liquid	Gibbs	Pr	RadCal Surrogate
CYCLOBUTENE	C4H6	200 -6000	156.70	5.180	357.00		Y	0.71	
C4H7	C4H7	298 -3000	184.40	5.176	357.00		Y	0.71	
1-BUTENE	C4H8	200 -6000	-0.54	5.176	357.00		Y	0.71	
N-PENTANE	C5H12	200 -6000	-146.76	5.784	341.10	Y	Y	0.79	N-HEPTANE
PYRILIUM	C5H4O	298 -3000	45.81	5.290	464.80		Y	0.71	
CYCLOPENTADIENIDE	C5H5	200 -6000	265.68	5.180	357.00		Y	0.71	
PENTANOL	C5H4OH	298 -6000	80.12	5.290	464.80		Y	0.71	
PYRILIUM	C5H5O	298 -3000	40.57	5.290	464.80		Y	0.71	
CYCLOPENTENE	C5H6	200 -6000	134.30	5.180	357.00		Y	0.71	
CYCLOHEXANE	C6H12	200 -6000	-123.30	6.182	297.10		Y	0.71	
N-HEXANE	C6H14	200 -6000	-166.92	5.949	399.30	Y	Y	0.79	N-HEPTANE
TRIACETYLENE	C6H2	200 -6000	670.00	5.180	357.00		Y	0.71	PROPYLENE
C6H3	C6H3	298 -3000	730.99	5.180	357.00		Y	0.71	
BENZINE	C6H4	298 -3000	443.53	5.290	464.80	Y	Y	0.71	
BUTALENE	C6H4	298 -3000	516.76	5.349	412.30		Y	0.71	
PHENYL	C6H5	200 -6000	337.20	5.290	464.80		Y	0.71	
PHENOLATE	C6H5O	200 -6000	47.70	5.290	464.80		Y	0.71	
PHENOL	C6H5OH	200 -6000	-96.40	5.290	464.80		Y	0.71	
BENZENE	C6H6	200 -6000	82.88	5.439	412.30		Y	1.50	TOLUENE
N-HEPTANE	C7H16	200 -6000	-187.78	4.701	205.78	Y	Y	0.83	N-HEPTANE
TOLUENE	C7H8	178 -6000	50.17	5.698	480.00	Y	Y	0.71	TOLUENE
N-OCTANE	C8H18	200 -6000	-208.75	4.892	231.60	Y	Y	0.64	N-HEPTANE
ISO-OCTANE	C8H18	200 -6000	-224.01	4.892	231.60	Y	Y	0.64	N-HEPTANE
CARBON TETRABROMIDE	CB4	200 -6000	79.50	6.120	442.00		Y	0.73	
CHLOROMETHYLIDYNE	CCl	200 -6000	432.61	4.065	157.80		Y	0.70	
DICHLOROMETHYLENE	CCl2	200 -6000	222.94	4.692	213.00		Y	0.71	
TRICHLOROMETHYL RADICAL	CCl3	200 -6000	71.13	5.320	268.00		Y	0.71	
CARBON TETRACHLORIDE	CCl4	200 -6000	-95.60	5.947	322.70		Y	0.73	
CADMIUM	Cd	200 -20000	111.80	2.606	1227.00		Y	0.67	
FLUOROMETHYLIDYNE	CF	200 -6000	242.30	3.635	94.20		Y	0.69	
FLUOROMETHYLIDYNEp	CF	298 -6000	1145.56	3.635	94.20		Y	0.69	
DIFLUOROMETHYLENE	CF2	200 -6000	-186.60	3.977	108.00		Y	0.71	
DIFLUOROMETHYLENEp	CF2	298 -6000	949.34	3.977	108.00		Y	0.71	
DICHLORODIFLUOROMETHANE	CF2Cl2	200 -6000	-490.80	5.250	253.00		Y	0.73	
TRIFLUOROMETHYL RADICAL	CF3	200 -6000	-467.40	4.320	121.00		Y	0.72	
TRIFLUOROMETHYL RADICAL-p	CF3	298 -6000	423.62	4.320	121.00		Y	0.72	
BROMOTRIFLUORMETHANE	CF3Br	200 -6000	-648.80	5.010	235.00		Y	0.73	
CHLOROTRIFLUORMETHANE	CF3Cl	200 -6000	-704.20	4.960	188.00		Y	0.73	
TETRAFLUOROMETHANE	CF4	200 -6000	-933.12	4.662	134.00		Y	0.72	
TRICHLOROMONOFUOROMETHANE	CFCl3	200 -6000	-283.70	5.440	334.00		Y	0.73	
METHYLIDYNE	CH	200 -20000	597.37	3.370	68.60		Y	0.69	
METHYLIDYNEp	CH	298 -20000	1630.57	3.370	68.60		Y	0.69	
METHYLENE	CH2	200 -6000	390.36	3.800	144.00		Y	0.71	

Table A.2: Optional gas and liquid species (continued).

Species	Formula	Temp. Range (K)	ΔH_f (kJ/mol)	σ (Å)	ε/k (K)	Liquid	Gibbs	Pr	RadCal Surrogate
METHYLENES	CH2	200 -3500	429.89	3.800	144.00		Y	0.71	
VYNILOXY RADICAL	CH2CHO	298 -5000	25.34	3.970	436.00		Y	0.71	
METHYLENE CHLORIDE	CH2Cl2	200 -6000	-95.00	4.898	356.30		Y	0.72	
BROMOCHLOROMETHANE	CH2ClBr	200 -6000	-45.00	4.880	410.00		Y	0.72	
KETENE	CH2CO	200 -6000	-49.58	3.970	436.00		Y	0.71	
DIFLUROMETHANE	CH2F2	200 -6000	-452.30	4.080	318.00		Y	0.71	
CHLOROFLUORMETHANE	CH2FCl	200 -6000	-265.70	4.480	318.00		Y	0.71	
DIODOMETHANE	CH2I2	200 -6000	117.57	5.160	449.20		Y	0.72	
HYDROXYMETHYL RADICAL	CH2OH	200 -6000	-17.80	3.690	417.00		Y	0.71	
HYDROXYMETHYL RADICALp	CH2OH	298 -6000	716.40	3.690	417.00		Y	0.71	
METHYL RADICAL	CH3	200 -6000	146.66	3.800	144.00		Y	0.71	
MEHTYL BROMIDE	CH3Br	200 -6000	-37.74	4.118	449.20		Y	0.71	
NEOPENTANE	CH3C(CH3)2CH3	200 -6000	-167.92	6.464	193.40		Y	0.71	PROPANE
CYCLOPROPYL	CH3CCH2	298 -6000	253.55	4.982	266.80		Y	0.71	
ETHANAL	CH3CHO	200 -6000	-166.19	3.970	436.00		Y	0.71	
CHLOROMETHANE	CH3Cl	200 -6000	-81.87	4.182	350.00		Y	0.71	
METHYL FLUORIDE	CH3F	200 -6000	-237.70	3.730	333.00		Y	0.70	
IODOMETHANE	CH3I	200 -6000	13.77	4.230	519.00		Y	0.71	
METHOXY RADICAL	CH3O	200 -6000	13.00	3.690	417.00		Y	0.71	
METHYL ETHER	CH3OCH3	200 -6000	-184.11	4.307	395.00		Y	0.72	METHANOL
METHANOL	CH3OH	200 -6000	-200.94	3.626	481.80		Y	0.95	METHANOL
METHANE	CH4	200 -6000	-74.60	3.758	148.60	Y	Y	0.70	METHANE
TRIBROMOMETHANE	CHBr3	200 -6000	16.74	5.330	559.00		Y	0.73	
BROMODICHLOROMETHANE	CHCl2Br	200 -6000	-45.00	5.250	427.00		Y	0.73	
CHLOROFORM	CHCl3	200 -6000	-102.70	5.389	340.20		Y	0.73	
DIFLUOROCHLOROMETHANE	CHF2Cl	200 -6000	-482.80	4.680	261.00		Y	0.72	
FLUOROFORM	CHF3	200 -6000	-693.30	4.330	240.00		Y	0.72	
CHFCIBr	CHFCIBr	200 -6000	-230.00	5.130	345.00		Y	0.71	
CHLORINE	Cl2	200 -6000	468.39	4.217	316.00	Y	Y	0.75	
CHLORINE ATOM	Cl	200 -20000	121.30	3.613	130.80		Y	0.67	
CHLORINE ATOMm	Cl	298 -20000	-233.96	3.613	130.80		Y	0.67	
CHLORINE ATOMp	Cl	298 -20000	1378.80	3.613	130.80		Y	0.67	
CHLORINE	Cl2	200 -6000	0.00	4.217	316.00		Y	0.70	
CYANOGEN CHLORIDE	CICN	200 -6000	134.20	4.047	338.70		Y	0.71	
CHLORINE FLUORIDE	ClF	200 -6000	-55.70	3.668	203.40		Y	0.70	
CHLORINE TRIFLUORIDE	ClF3	200 -6000	-164.60	4.288	335.70		Y	0.73	
MONOCHLORINE MONOXIDE	ClO	200 -6000	101.62	3.842	184.00		Y	0.69	
CYANO RADICAL	CN	200 -20000	438.68	3.856	75.00		Y	0.69	
CYANO RADICALm	CN	298 -6000	63.89	3.856	75.00		Y	0.69	
CYANO RADICALp	CN	298 -20000	1798.89	3.856	75.00		Y	0.69	
CARBON MONOXIDE	CO	200 -20000	-110.54	3.690	91.70	Y	Y	0.73	CARBON MONOXIDE
CARBON MONOXIDEp	CO	298 -20000	1247.79	3.690	91.70		Y	0.73	CARBON MONOXIDE
CARBON DIOXIDE	CO2	200 -20000	-393.51	3.941	195.20		Y	0.75	CARBON DIOXIDE

Table A.2: Optional gas and liquid species (continued).

Species	Formula	Temp. Range (K)	ΔH_f (kJ/mol)	σ (Å)	ϵ/k (K)	Liquid	Gibbs	Pr	RadCal Surrogate
CARBON DIOXIDEp	CO2	298 -20000	944.69	3.941	195.20		Y	0.75	CARBON DIOXIDE
CARBONYL SULFIDE	COS	200 -6000	-141.70	4.130	336.00		Y	0.71	
CARBON MONOPHOSPHIDE	CP	200 -6000	520.16	4.400	227.00		Y	0.69	
CARBON MONOSULFIDE	CS	200 -6000	278.55	4.216	199.40		Y	0.69	
CARBON DISULFIDE	CS2	200 -6000	116.70	4.483	467.00		Y	7.15	
FLUORINE ATOM	F	200 -20000	79.38	2.980	112.60		Y	0.67	
FLUORINE ATOMm	F	298 -20000	-255.09	2.980	112.60		Y	0.67	
FLUORINE ATOMp	F	298 -20000	1766.82	2.980	112.60		Y	0.67	
FLUORINE	F2	200 -6000	0.00	3.357	112.60		Y	0.68	
CYANONGEN FLUROIDE	FCN	200 -6000	34.33	3.578	168.00		Y	0.71	
HYDROGEN ATOM	H	200 -20000	218.00	2.708	37.00		Y	0.67	
HYDROGEN ATOMm	H	298 -20000	139.03	2.708	37.00		Y	0.67	
HYDROGEN ATOMp	H	298 -20000	1536.25	2.708	37.00		Y	0.67	
HYDROGEN	H2	200 -20000	0.00	2.827	59.70	Y	Y	0.69	
HYDROGENm	H2	298 -6000	235.17	2.827	59.70		Y	0.69	
HYDROGENp	H2	298 -20000	1494.67	2.827	59.70		Y	0.69	
PROPADIENONE	H2C4O	298 -4000	228.43	5.180	357.00		Y	0.71	
METHYLENE-AMIDOGEN	H2CN	298 -6000	247.33	3.630	569.00		Y	0.71	
WATER VAPOR	H2O	273 -600	-241.83	2.641	809.10	Y	Y	1.00	WATER VAPOR
WATER VAPORp	H2O	298 -20000	981.60	2.641	809.10		Y	1.00	WATER VAPOR
HYDROGEN PEROXIDE	H2O2	200 -6000	-135.88	4.196	289.30	Y	Y	0.71	
HYDROGEN SULFIDE	H2S	200 -6000	-20.60	3.623	301.10	Y	Y	0.70	
OXONIUM	H3O	298 -20000	598.00	3.150	106.20		Y	0.71	
HYDROGEN BROMIDE	HBr	200 -6000	-36.29	3.353	449.00	Y	Y	0.69	
KETENYL RADICAL	HCCO	200 -6000	176.57	2.500	150.00		Y	0.71	
ETHYNOL	HCCOH	298 -5000	153.92	3.970	436.00		Y	0.71	
FORMALDEHYDE	HCHO	200 -6000	-108.58	3.590	498.00	Y	Y	0.71	METHANOL
HYDROGEN CHLORIDE	HCl	200 -6000	-92.31	3.339	344.70	Y	Y	0.75	
HYDROGEN CYANIDE	HCN	200 -6000	133.08	3.638	569.40	Y	Y	0.70	
HCNN	HCNN	298 -5000	462.12	2.500	150.00		Y	0.71	
FULMINIC ACID	HCNO	298 -5000	171.04	3.828	232.40		Y	0.71	
FORMYL RADICAL	HCO	200 -6000	42.40	3.590	498.00		Y	0.71	
FORMYL RADICALp	HCO	298 -20000	833.03	3.590	498.00		Y	0.71	
FORMIC ACID	HCOOH	200 -6000	-378.57	3.970	436.00		Y	0.71	
HELIUM	He	200 -20000	0.00	2.551	10.22	Y	Y	0.68	
HELIUMp	He	298 -20000	2378.52	2.551	10.22		Y	0.68	
HYDROGEN FLUORIDE	HF	200 -20000	-273.30	3.148	330.00	Y	Y	0.71	
MERCURY	Hg	200 -20000	61.38	2.969	750.00		Y	0.67	
MERCURYp	Hg	298 -20000	1074.64	2.969	750.00		Y	0.67	
MERCURIC BROMIDE	HgBr2	200 -6000	-91.31	5.088	686.20		Y	0.73	
HYDROGEN IODIDE	HI	200 -6000	26.36	4.211	288.70		Y	0.69	
ISOCYANIC ACID	HNCO	200 -6000	-118.06	3.828	232.40		Y	0.71	
NITROSYL HYDRIDE	HNO	200 -6000	102.03	3.495	116.70		Y	0.71	

Table A.2: Optional gas and liquid species (continued).

Species	Formula	Temp. Range (K)	ΔH_f (kJ/mol)	σ (Å)	ϵ/k (K)	Liquid	Gibbs	Pr	RadCal Surrogate
NITROUS ACID	HNO2	200 -6000	-78.45	3.828	232.40		Y	0.71	
NITRIC ACID	HNO3	200 -6000	-133.91	4.175	378.40		Y	0.71	
HYDROPEROXY RADICAL	HO2	200 -6000	12.02	3.020	106.50		Y	0.71	
HYDROPEROXY RADICALm	HO2	298 -6000	-97.92	3.020	106.50		Y	0.71	
CYANIC ACID	HOCN	298 -5000	-11.80	3.828	232.40		Y	0.71	
IODINE	I	200 -20000	106.76	4.320	210.70		Y	0.67	
IODINEm	I	298 -20000	-194.60	4.320	210.70		Y	0.67	
IODINEp	I	298 -20000	1121.35	4.320	210.70		Y	0.67	
IODINE2	I2	200 -6000	62.42	5.160	474.20		Y	0.70	
KRYPTONm	K	298 -20000	34.42	3.655	178.90		Y	1.43	
KRYPTONp	K	298 -20000	514.01	3.655	178.90		Y	1.43	
KRYPTON	Kr	200 -20000	0.00	3.655	178.90		Y	0.67	
LITHIUM	Li	200 -20000	159.30	2.850	1899.00		Y	0.67	
LITHIUMm	Li	298 -20000	93.47	2.850	1899.00		Y	0.67	
LITHIUMp	Li	298 -20000	685.72	2.850	1899.00		Y	0.67	
LITHIUM DIMER	Li2	200 -6000	215.90	3.200	1899.00		Y	0.70	
LITHIUM DIMERp	Li2	298 -6000	721.61	3.200	1899.00		Y	0.70	
LITHIUM OXIDE	Li2O	200 -6000	-167.34	3.561	1827.00		Y	0.71	
LITHIUM OXIDEp	Li2O	298 -6000	439.10	3.561	1827.00		Y	0.71	
LITHIUM BROMIDE	LiBr	200 -6000	-151.16	3.748	1815.00		Y	0.70	
LITHIUM CHLORIDE	LiCl	200 -6000	-193.78	3.078	1919.00		Y	0.70	
LITHIUM FLUORIDE	LiF	200 -6000	-340.94	3.278	2305.00		Y	0.70	
LITHIUM IODIDE	LiI	200 -6000	-85.27	4.180	1726.00		Y	0.71	
LITHIUM MONOXIDE	LiO	200 -6000	72.91	3.334	450.00		Y	0.69	
MAGNESIUM	Mg	200 -20000	147.10	2.926	1614.00		Y	0.67	
MAGNESIUMp	Mg	298 -20000	891.05	2.926	1614.00		Y	0.67	
MAGNESIUM DIMER	Mg2	200 -6000	286.51	3.301	1614.00		Y	0.71	
MAGNESIUM MONOCHLORIDE	MgCl	200 -6000	-54.70	3.759	714.00		Y	0.70	
MAGNESIUM MONOCHLORIDEp	MgCl	298 -20000	646.34	3.759	714.00		Y	0.70	
MAGNESIUM CHLORIDE	MgCl2	200 -6000	-399.17	34.000	1988.00		Y	0.72	
MAGNESIUM MONOFLUORIDE	MgF	200 -6000	-232.27	4.340	1988.00		Y	0.70	
MAGNESIUM MONOFLUORIDEp	MgF	298 -20000	516.87	4.340	1988.00		Y	0.70	
MAGNESIUM FLUORIDE	MgF2	200 -6000	-735.50	3.329	426.00		Y	0.72	
MAGNESIUM FLUORIDEp	MgF2	298 -20000	582.69	3.623	2964.00		Y	0.71	
NITROGEN ATOM	N	200 -20000	472.68	3.298	71.40		Y	0.71	
NITROGEN ATOMm	N	298 -20000	473.54	3.298	71.40		Y	0.71	
NITROGEN ATOMP	N	298 -20000	1882.13	3.298	71.40		Y	0.71	
NITROGEN	N2	200 -20000	0.00	3.798	71.40	Y	Y	0.71	
NITROGENm	N2	298 -20000	148.18	3.798	71.40		Y	0.71	
NITROGENp	N2	298 -20000	1509.51	3.798	71.40		Y	0.71	
DIIMIDE	N2H2	200 -6000	211.86	3.798	71.40		Y	0.71	
HYDRAZINYL RADICAL	N2H3	298 -5000	153.92	3.900	200.00		Y	0.71	
HYDRAZINE	N2H4	200 -6000	95.18	4.230	205.00		Y	0.71	

Table A.2: Optional gas and liquid species (continued).

Species	Formula	Temp. Range (K)	ΔH_f (kJ/mol)	σ (Å)	ϵ/k (K)	Liquid	Gibbs	Pr	RadCal Surrogate
NITROUS OXIDE	N2O	182 -6000	81.60	3.828	232.40	Y	Y	0.74	
NITROUS OXIDE _p	N2O	298 -6000	1332.96	3.828	232.40		Y	0.74	
SODIUM	Na	200 -20000	107.50	3.567	1357.00		Y	0.67	
SODIUM _m	Na	298 -20000	48.45	3.567	1357.00		Y	0.67	
SODIUM _p	Na	298 -20000	609.54	3.567	1357.00		Y	0.67	
DISODIUM	Na2	200 -6000	142.34	4.156	1375.00		Y	0.70	
DISODIUM OXIDE	Na2O	200 -6000	-16.56	4.358	1827.00		Y	0.71	
DISODIUM OXIDE _p	Na2O	298 -6000	520.83	4.358	1827.00		Y	0.71	
DISODIUM PEROXIDE	Na2O2	200 -6000	-123.93	4.530	1208.00		Y	0.71	
SODIUM BROMIDE	NaBr	200 -6000	-145.93	4.226	1963.00		Y	0.71	
SODIUM CHLORIDE	NaCl	200 -6000	-181.54	4.186	1898.00		Y	0.70	
SODIUM CYANIDE	NaCN	200 -6000	94.27	4.395	2088.00		Y	0.71	
SODIUM FLUORIDE	NaF	200 -6000	-295.16	3.756	2333.00		Y	0.70	
SODIUM HYDRIDE	NaH	200 -6000	140.84	3.542	93.30		Y	0.71	
SODIUM BICARBONATE	NaHCO3	298 -3000	-937.30	4.530	1257.00		Y	0.71	
SODIUM IODIDE	NaI	200 -6000	-90.64	4.658	1856.00		Y	0.70	
SODIUM OXIDE	NaO	200 -6000	106.51	3.812	383.00		Y	0.70	
SODIUM SUPEROXIDE	NaO2	298 -3000	106.52	3.812	383.00		Y	0.71	
SODIUM HYDROXIDE	NaOH	200 -6000	-191.00	3.804	1962.00		Y	0.71	
SODIUM HYDROXIDE _p	NaOH	298 -20000	683.86	3.804	1962.00		Y	0.71	
ISOCYANATO RADICAL	NCO	200 -6000	131.85	3.828	232.40		Y	0.71	
NEON	Ne	200 -20000	0.00	2.820	32.80		Y	0.67	
NEON _p	Ne	298 -20000	2086.97	2.820	32.80		Y	0.67	
NITROGEN TRIFLUORIDE	NF3	200 -6000	-131.70	4.154	175.00		Y	0.72	
IMIDOGEN	NH	200 -20000	357.03	3.312	65.30		Y	0.69	
IMIDOGEN _p	NH	298 -20000	1665.79	3.312	65.30		Y	0.69	
DINITROGEN MONOHYDRIDE	NH2	200 -6000	189.13	2.650	80.00		Y	0.71	
AMMONIA	NH3	200 -6000	-45.94	2.900	558.30	Y	Y	0.87	
NNH	NNH	250 -4000	245.07	3.798	71.40		Y	0.71	
NITRIC OXIDE	NO	110 -20000	91.27	3.492	116.70	Y	Y	0.74	
NITRIC OXIDE _p	NO	298 -20000	990.81	3.492	116.70		Y	0.74	
NITROGEN DIOXIDE	NO2	290 -6000	34.19	3.922	204.88	Y	Y	6.10	
NITROGEN DIOXIDE _m	NO2	298 -6000	-200.04	3.922	204.88		Y	6.10	
NITRATE RADICAL	NO3	200 -6000	71.13	4.175	378.40		Y	0.71	
NITRATE RADICAL _m	NO3	298 -6000	-310.78	4.175	378.40		Y	0.71	
NITROSYL CHLORIDE	NOCl	200 -6000	52.70	4.112	395.30		Y	1.42	
OXYGEN ATOM	O	200 -20000	249.18	3.050	106.60		Y	0.67	
OXYGEN ATOM _m	O	298 -20000	101.85	3.050	106.60		Y	0.67	
OXYGEN ATOM _p	O	298 -20000	1568.79	3.050	106.60		Y	0.67	
OXYGEN	O2	60 -20000	0.00	3.467	106.70	Y	Y	0.71	
OXYGEN _m	O2	298 -6000	-48.03	3.467	186.70		Y	0.71	
OXYGEN _p	O2	298 -20000	1171.83	3.467	186.70		Y	0.71	
OZONE	O3	200 -6000	141.80	3.875	208.40		Y	1.48	

Table A.2: Optional gas and liquid species (continued).

Species	Formula	Temp. Range (K)	ΔH_f (kJ/mol)	σ (Å)	ϵ/k (K)	Liquid	Gibbs	Pr	RadCal Surrogate
HYDROXYL RADICAL	OH	200 -20000	37.28	3.147	79.80		Y	0.69	
HYDROXYL RADICAL _m	OH	298 -6000	-145.26	3.147	79.80		Y	0.69	
HYDROXYL RADICAL _p	OH	298 -20000	1299.21	3.147	79.80		Y	0.69	
PHOSPHORUS	P	200 -20000	316.50	4.115	653.00		Y	0.67	
PHOSPHORUS _m	P	298 -20000	238.83	4.115	653.00		Y	0.67	
PHOSPHORUS _p	P	298 -20000	1336.45	4.115	653.00		Y	0.67	
PHOSPHORUS DIMER	P ₂	200 -6000	144.00	4.887	653.00		Y	0.70	
TETRAPHOSPHORUS	P ₄	200 -6000	58.90	5.455	711.00		Y	0.73	
PHOSPHORUS CHLORIDE	PCl	200 -6000	134.62	4.552	454.00		Y	0.71	
PHOSPHORUS TRICHLORIDE	PCl ₃	200 -6000	-289.50	5.240	419.00		Y	0.73	
PHOSPHORUS MONOFLUORIDE	PF	200 -6000	-47.94	4.120	271.00		Y	0.69	
PHOSPHORUS MONOFLUORIDE _m	PF	298 -20000	-164.05	4.120	271.00		Y	0.69	
PHOSPHORUS MONOFLUORIDE _p	PF	298 -20000	901.52	4.120	271.00		Y	0.69	
PHOSPHORUS TRIFLUORIDE	PF ₃	200 -6000	-957.40	4.360	203.30		Y	0.72	
PHOSPHINE	PH ₃	200 -6000	5.44	3.981	251.50		Y	0.70	
PHOSPHORUS MONONITRIDE	PN	200 -6000	171.49	4.432	216.00		Y	0.69	
PHOSPHORUS MONOXIDE	PO	200 -6000	-27.86	4.177	264.00		Y	0.69	
PHOSPHORUS MONOXIDE _m	PO	298 -6000	-140.07	4.177	264.00		Y	0.69	
PHOSPHORUS MONOSULFIDE	PS	200 -6000	150.43	4.730	744.00		Y	0.70	
SULFUR	S	200 -20000	277.17	3.839	847.00		Y	0.69	
SULFUR _m	S	298 -20000	70.37	3.839	847.00		Y	0.69	
SULFUR _p	S	298 -20000	1282.50	3.839	847.00		Y	0.69	
SULFUR DIMER	S ₂	200 -6000	128.60	4.519	847.00		Y	0.69	
SULFUR DIMER _m	S ₂	298 -6000	-37.13	4.519	847.00		Y	0.69	
DIFLUORODISULFANE	S ₂ F ₂	200 -6000	-401.41	4.702	205.60		Y	0.71	
SULFUR HEXAFLUORIDE	SF ₆	200 -6000	-1219.40	5.128	222.10		Y	0.77	
SULFUR HEXAFLUORIDE _m	SF ₆	298 -20000	-1341.88	5.128	222.10		Y	0.77	
SILICON	Si	200 -20000	450.00	2.910	3036.00		Y	0.67	
SILICON _m	Si	298 -20000	308.82	2.910	3036.00		Y	0.67	
SILICON _p	Si	298 -20000	1242.51	2.910	3036.00		Y	0.67	
SILICON DIMER	Si ₂	200 -6000	580.20	3.280	3036.00		Y	0.70	
CLOROSIYLIDYNE	SiCl	200 -6000	142.36	3.748	980.00		Y	0.71	
SILICON TETRACHLORIDE	SiCl ₄	200 -6000	-662.20	5.977	390.20		Y	0.74	
SILICON FLUORIDE	SiF	200 -6000	-25.23	3.318	585.00		Y	0.70	
SILICON TETRAFLUORIDE	SiF ₄	200 -6000	-1615.78	4.888	171.90		Y	0.73	
SILANE	SiH ₄	200 -6000	34.70	4.084	287.60		Y	0.71	
SILICON MONOXIDE	SiO	200 -6000	-98.84	3.374	569.00		Y	0.69	
SILICON DIOXIDE	SiO ₂	200 -6000	-322.07	3.706	2954.00		Y	0.72	
SILICON MONOSULFIDE	SiS	200 -6000	108.19	3.900	1432.00		Y	0.70	
STANNIC MONOBROMIDE	SnBr	200 -6000	75.64	6.388	563.70		Y	0.71	
STANNIC BROMIDE	SnBr ₄	200 -6000	-324.22	6.388	563.70		Y	0.74	
TIN(IV)CHLORIDE	SnCl ₄	200 -6000	-478.47	6.202	420.00		Y	0.74	
SULFUR MONOXIDE	SO	200 -6000	4.76	3.993	301.00		Y	0.50	

Table A.2: Optional gas and liquid species (continued).

Species	Formula	Temp. Range (K)	ΔH_f (kJ/mol)	σ (Å)	ϵ/k (K)	Liquid	Gibbs	Pr	RadCal Surrogate
SULFUR MONOXIDE _m	SO	298 -6000	-105.97	3.993	301.00		Y	0.50	
SULFUR DIOXIDE	SO2	200 -6000	-296.81	4.112	335.40	Y	Y	0.91	
SULFUR DIOXIDE _m	SO2	298 -6000	-408.61	4.112	335.40		Y	0.91	
URANIUM HEXAFLUORIDE _m	UF6	298 -20000	-2691.31	5.967	236.80		Y	0.71	
URANIUM HEXAFLUORIDE	UF6	200 -20000	-2148.64	5.967	236.80		Y	0.71	
XENON	Xe	300 -20000	0.00	4.047	231.00		Y	0.67	
XENON _p	Xe	298 -20000	1176.55	4.047	231.00		Y	0.67	
ZINC	Zn	200 -20000	130.40	2.891	2.89		Y	0.67	
ZINC _p	Zn	298 -20000	1043.00	2.891	2.89		Y	0.67	