

NIST Special Publication 1019

Fire Dynamics Simulator User's Guide



VTT Technical Research Centre of Finland

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

NIST Special Publication 1019

Fire Dynamics Simulator User's Guide

Fire Research Division
Engineering Laboratory
Gaithersburg, Maryland, USA

VTT Technical Research Centre of Finland
Espoo, Finland



June 13, 2012
FDS Version 6.0
SVN Repository Revision : 11024



U.S. Department of Commerce
John E. Bryson, Secretary

National Institute of Standards and Technology
Patrick D. Gallagher, Under Secretary of Commerce for Standards and Technology and Director

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, 259 pages (October 2007)
CODEN: NSPUE2

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 2007

For sale by the Superintendent of Documents, U.S. Government Printing Office
Internet: bookstore.gpo.gov – Phone: (202) 512-1800 – Fax: (202) 512-2250
Mail: Stop SSOP, Washington, DC 20402-0001

Authors

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 VTT Technical Research Centre of Finland. Its developers and contributors are listed below.

Principal Developers (in alphabetical order)

Jason Floyd, Hughes Associates, Inc., Baltimore, Maryland, USA
Glenn Forney, NIST
Simo Hostikka, VTT
Timo Korhonen, VTT
Randall McDermott, NIST
Kevin McGrattan, NIST

Contributors

Elizabeth Blanchard, Centre Scientifique et Technique du Bâtiment (CSTB), Paris, France
Susan Killian, hhpberlin, Germany
Charles Luo, Global Engineering and Materials, Inc., Princeton, New Jersey, USA
Anna Matala, VTT
William Mell, U.S. Forest Service, Seattle, Washington, USA
Christian Rogsch, Neustadt/Wstr., Germany
Topi Sikanen, VTT
Ben Trettel, University of Maryland, USA
Craig Weinschenk, NIST

About the Authors

Elizabeth Blanchard is a fire protection engineer at the French building agency CSTB. She holds a master of science degree in mathematical modeling and a doctorate in mechanics and thermal engineering. She is mainly involved at CSTB in the research program concerning water spray.

Jason Floyd is a Senior Engineer at Hughes Associates, Inc., in Baltimore, Maryland. He received a bachelors of science and Ph.D. in the Nuclear Engineering Program of the University of Maryland. After graduating, he won a National Research Council Post-Doctoral Fellowship at the Building and Fire Research Laboratory of NIST, where he developed the combustion algorithm within FDS. He is a principal developer of the combustion model and control logic within FDS.

Glenn Forney is a computer scientist in the Engineering Laboratory of NIST. He received a bachelors of science degree in mathematics from Salisbury State College in 1978 and a master of science and a doctorate in mathematics at Clemson University in 1980 and 1984. He joined the NIST staff 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, a software tool for visualizing Fire Dynamics Simulation data.

Simo Hostikka is a Senior Research Scientist 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.

Susan Kilian is a mathematician with numerics and scientific computing expertise. She received her diploma from the University of Heidelberg and received her doctorate from the Technical University of Dortmund in 2002. Since 2007 she has been a research scientist for hhpberlin, a fire safety engineering firm located in Berlin, Germany. Her research interests include high performance computing and the development of efficient parallel solvers for the pressure Poisson equation.

Charles Luo is a Senior Research Scientist at Global Engineering and Materials, Inc., in Princeton, New Jersey. He received a B.S. in theoretical and applied mechanics from the University of Science and Technology of China in 2002, and a doctorate in mechanical engineering from the State University of New York at Buffalo in 2010. His research interests include fire-structure interaction, immersed boundary methods, and fire response of composite and aluminum structures.

Anna Matala is a Research Scientist at VTT Technical Research Centre of Finland and a PhD candidate at Aalto University School of Science. She received her M.Sc. degree in Systems and Operations Research from Helsinki University of Technology in 2008. Her research concentrates on pyrolysis modelling and parameter estimation in fire simulations.

Randall McDermott joined the research staff of the Building and Fire Research Lab in 2008. He received a B.S. from the University of Tulsa in Chemical Engineering in 1994 and a doctorate at the University of

Utah in 2005. His research interests include subgrid-scale models and numerical methods for large-eddy simulation, adaptive mesh refinement, immersed boundary methods, and Lagrangian particle methods.

Kevin McGrattan is a mathematician in the Engineering Laboratory of NIST. He received a bachelors 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.

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.

Christian Rogsch received a Diploma degree (like M.Sc.) in Safety Engineering from the University of Wuppertal, Germany. He works on shared-memory parallelization (OpenMP) of the Fire Dynamics Simulator.

Topi Sikanen is a Research Scientist at VTT Technical Research Centre of Finland and a graduate student at Aalto University School of Science. He received his M.Sc. degree in Systems and Operations Research from Helsinki University of Technology in 2008. He works on the Lagrangian particle and liquid evaporation models.

Ben Trettel is a graduate student at the University of Maryland. He received a B.S. degree from the University of Maryland in Mechanical Engineering in 2011. He develops models for the transport of Lagrangian particles for the Fire Dynamics Simulator.

Craig Weinschenk joined the Fire Research Division as a National Research Council Postdoctoral Research Associate in 2011. He received a B.S. from Rowan University in Mechanical Engineering in 2006, an M.S. from the University of Texas-Austin in Mechanical Engineering in 2007, and a doctorate from the University of Texas-Austin in Mechanical Engineering in 2011. His research interests include numerical combustion, quadrature method of moments, 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 analyses performed using these tools.

Users are warned that FDS is intended for use only by those competent in the fields of fluid dynamics, thermodynamics, combustion, and heat transfer, 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.

At NIST, thanks to Dan Madrzykowski, Doug Walton, Bob Vettori, Dave Stroup, Steve Kerber and Nelson Bryner, who have used FDS and Smokeview as part of several investigations of fire fighter line of duty deaths. As part of these studies, 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 maintenance and development 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, Dave Stroup, and Jason Dreisbach of NRC, and Francisco Joglar of SAIC.

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.

Prof. 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.

Prof. Ian Thomas of Victoria University has also presented short courses on the use of FDS in Australia. His students have also performed some validation work on compartment fires.

Prof. Charles Fleischmann and his students at the University of Canterbury, New Zealand, have provided valuable assistance in improving the documentation and usability of the model.

James White Jr. of the Western Fire Center has provided valuable feedback on how to improve the functionality of the model in the area of forensic science.

Paul Hart of Swiss Re, GAP Services, and Pravinray Gandhi of Underwriters Laboratories provided useful suggestions about water droplet transport on solid objects.

Dr. Chris Lautenberger of University of California, Berkeley, has helped in development and improving the documentation of the pyrolysis models.

François Demouge of the Centre Scientifique et Technique du Bâtiment (CSTB) in France assisted with implementation of synthetic turbulence inflow boundary conditions.

Contents

Authors	i
About the Authors	iii
Preface	v
Disclaimer	vii
Acknowledgments	ix
I The Basics of FDS	1
1 Introduction	3
1.1 Features of FDS	3
2 Getting Started	5
2.1 How to Acquire FDS and Smokeview	5
2.2 Computer Hardware Requirements	5
2.3 Computer Operating System (OS) and Software Requirements	6
3 Running FDS	7
3.1 Starting an FDS Calculation	7
3.1.1 Starting an FDS Calculation (Single Processor Version)	7
3.1.2 Starting an FDS Calculation (Multiple Processor Version)	8
3.2 Monitoring Progress	10
4 User Support	13
4.1 The Version Number	13
4.2 Common Error Statements	14
4.3 Support Requests and Bug Tracking	15
4.4 Known Issues	16
II Writing an FDS Input File	19
5 The Basic Structure of an Input File	21
5.1 Naming the Job	21
5.2 Namelist Formatting	21
5.3 Input File Structure	22

6	Setting the Bounds of Time and Space	25
6.1	Naming the Job: The <code>HEAD</code> Namelist Group (Table 16.7)	25
6.2	Simulation Time: The <code>TIME</code> Namelist Group (Table 16.29)	25
6.2.1	Basics	25
6.2.2	Special Topic: Controlling the Time Step	26
6.2.3	Special Topic: Steady-State Applications	26
6.3	Computational Meshes: The <code>MESH</code> Namelist Group (Table 16.13)	27
6.3.1	Basics	27
6.3.2	Two-Dimensional and Axially-Symmetric Calculations	27
6.3.3	Multiple Meshes and Parallel Processing	28
6.3.4	Mesh Alignment	29
6.3.5	Mesh Stretching: The <code>TRNX</code> , <code>TRNY</code> and/or <code>TRNZ</code> Namelist Groups (Table 16.30)	31
6.3.6	Mesh Resolution	33
6.4	Miscellaneous Parameters: The <code>MISC</code> Namelist Group (Table 16.14)	34
6.4.1	Basics	34
6.4.2	Special Topic: Mean Forcing and Data Assimilation	34
6.4.3	Special Topic: Specified Force Field	35
6.4.4	Special Topic: Stopping and Restarting Calculations	35
6.4.5	Special Topic: Initializing a 3D Velocity Field	35
6.4.6	Special Topic: Defying Gravity	36
6.4.7	Special Topic: The Baroclinic Vorticity	37
6.4.8	Special Topic: Large Eddy Simulation Parameters	37
6.4.9	Special Topic: Numerical Stability Parameters	38
6.4.10	Adjusting the Time Step	40
6.5	Special Topic: Unusual Initial Conditions: The <code>INIT</code> Namelist Group (Table 16.10)	41
6.6	Special Topic: The Pressure Solver: The <code>PRES</code> Namelist Group (Table 16.18)	42
6.7	Special Topic: Setting Limits: The <code>CLIP</code> Namelist Group (Table 16.2)	44
7	Building the Model	45
7.1	Bounding Surfaces: The <code>SURF</code> Namelist Group (Table 16.27)	45
7.2	Creating Obstructions: The <code>OBST</code> Namelist Group (Table 16.16)	45
7.2.1	Basics	45
7.2.2	Repeated Obstructions: The <code>MULT</code> Namelist Group (Table 16.15)	47
7.2.3	Non-rectangular Geometry and Sloped Ceilings	48
7.3	Creating Voids: The <code>HOLE</code> Namelist Group (Table 16.8)	50
7.4	Applying Surface Properties: The <code>VENT</code> Namelist Group (Table 16.31)	51
7.4.1	Basics	51
7.4.2	Special <code>VENTs</code>	52
7.4.3	Controlling <code>VENTs</code>	53
7.4.4	Trouble-Shooting <code>VENTs</code>	53
7.4.5	Special Topic: Synthetic Turbulence Inflow Boundary Conditions	54
7.5	Coloring Obstructions, Vents, Surfaces and Meshes	56
7.5.1	Texture Mapping	56
8	Fire and Thermal Boundary Conditions	59
8.1	Basics	59
8.2	Surface Temperature and Heat Flux	61
8.2.1	Specified Solid Surface Temperature	61

8.2.2	Special Topic: Convective Heat Transfer Options	61
8.2.3	Special Topic: Adiabatic Surfaces	62
8.3	Heat Conduction in Solids	63
8.3.1	Structure of Solid Boundaries	63
8.3.2	Thermal Properties	64
8.3.3	Back Side Boundary Conditions	65
8.3.4	Initial and Back Side Temperature	65
8.3.5	Walls with Different Materials Front and Back	65
8.3.6	Special Topic: Non-Planar Walls and Targets	66
8.3.7	Special Topic: Solid Phase Numerical Gridding Issues	66
8.4	Pyrolysis Models	68
8.4.1	A Gas Burner with a Specified Heat Release Rate	68
8.4.2	Special Topic: A Radially-Spreading Fire	68
8.4.3	Solid Fuels that Burn at a Specified Rate	69
8.4.4	Solid Fuels that do NOT Burn at a Specified Rate	70
8.4.5	Liquid Fuels	78
8.4.6	Fuel Burnout	79
8.5	Testing Your Pyrolysis Model	82
8.6	Full-Scale Examples: Furniture	84
9	Ventilation	87
9.1	Simple Vents, Fans and Heaters	87
9.1.1	Simple Supply and Exhaust Vents	87
9.1.2	Total Mass Flux	88
9.1.3	Heaters	88
9.1.4	Louvered Vents	88
9.1.5	Special Topic: Tangential Velocity Boundary Conditions at Solid Surfaces	88
9.1.6	Species and Species Mass Flux Boundary Conditions	89
9.2	HVAC Systems: The HVAC Namelist Group (Table 16.9)	90
9.2.1	HVAC Duct Parameters	90
9.2.2	HVAC Node Parameters	91
9.2.3	HVAC Fan Parameters	92
9.2.4	HVAC Filter Parameters: HVAC_filter	97
9.2.5	HVAC Aircoil Parameters: HVAC_aircoil	97
9.2.6	Louvered HVAC Vents	99
9.3	Pressure-Related Effects: The ZONE Namelist Group (Table 16.31)	99
9.3.1	Specifying Pressure Zones	100
9.3.2	Leaks	102
9.3.3	Special Topic: Stack Effect	104
9.4	Special Topic: Pressure Boundary Conditions	105
9.5	Special Topic: Fires and Flows in the Outdoors	107
10	User-Specified Functions, Ramps and Tables	109
10.1	Time-Dependent Functions	109
10.2	Temperature-Dependent Functions	111
10.3	Tabular Functions	111
11	Chemistry	113

11.1	Specifying Gas Species: The SPEC Namelist Group	113
11.1.1	Basics	114
11.1.2	Special Topic: Gas and Liquid Properties	114
11.2	Specifying Gas Mixtures: The SMIX Namelist Group	118
11.2.1	Background Species	118
12	Combustion	119
12.1	Single-Step, Mixing-Controlled Combustion	119
12.1.1	Simple Chemistry Inputs	119
12.1.2	Heat of Combustion	121
12.1.3	Special Topic: Turbulent Combustion	122
12.1.4	Special Topic: Flame Extinction	122
12.2	Complex Stoichiometry	124
12.2.1	Complex Fuel Molecules	124
12.2.2	Two-Step Reaction Involving the Formation of CO	126
12.3	Finite Rate Combustion	128
12.3.1	Special Topic: Mixed-Mode Combustion	128
12.4	Aerosol Deposition	129
12.5	Other Reaction Parameters	131
12.5.1	Special Topic: Using the EQUATION input parameter	131
12.5.2	Special Topic: Reaction Rate Limiters	131
12.5.3	Special Topic: Diagnostic Parameters	131
13	Radiation	133
13.1	Basic Radiation Parameters: The RADI Namelist Group	133
13.1.1	Radiative Fraction	133
13.1.2	Spatial and Temporal Resolution of the Radiation Transport Equation	133
13.2	Radiative Absorption	134
13.2.1	RadCal Issues	134
13.2.2	Radiative Absorption and Scattering by Particles	134
13.2.3	Wide Band Model	134
14	Particles and Droplets	137
14.1	Basics	137
14.1.1	Output Quantities	137
14.1.2	Massless Particles	138
14.2	Particle and Droplet Insertion	139
14.2.1	Particles Introduced at a Solid Surface	139
14.2.2	Droplets Introduced at a Sprinkler or Nozzle	140
14.2.3	Particles or Droplets Introduced within a Volume	140
14.2.4	Controlling the Number of Particles and Droplets	141
14.3	Liquid Droplets	142
14.3.1	Droplet Thermal Properties	142
14.3.2	Radiative Properties	142
14.3.3	Size Distribution	143
14.3.4	Secondary breakup	143
14.3.5	Fuel Droplets	143
14.4	Solid Particles	146

14.4.1	Basics	146
14.4.2	Drag	146
14.4.3	Gas Generating Particles	147
14.4.4	Vegetation	147
14.4.5	Screens	149
14.5	Special Topic: Suppression by Water	150
14.5.1	Velocity on Solid Surfaces	150
14.5.2	Reduction of the Burning Rate	151
15	Devices and Control Logic	153
15.1	Device Location and Orientation: The <code>DEVC</code> Namelist Group (Table 16.5)	153
15.2	Device Output	154
15.3	Special Device Properties: The <code>PROP</code> Namelist Group (Table 16.20)	155
15.3.1	Sprinklers	155
15.3.2	Nozzles	158
15.3.3	Specified Entrainment (Velocity Patch)	159
15.3.4	Heat Detectors	160
15.3.5	Smoke Detectors	160
15.3.6	Beam Detection Systems	161
15.3.7	Aspiration Detection Systems	162
15.3.8	Electrical Cable Failure	163
15.4	Basic Control Logic	166
15.4.1	Creating and Removing Obstructions	166
15.4.2	Activating and Deactivating Vents	167
15.5	Advanced Control Functions: The <code>CTRL</code> Namelist Group	168
15.5.1	Control Functions: <code>ANY</code> , <code>ALL</code> , <code>ONLY</code> , and <code>AT_LEAST</code>	169
15.5.2	Control Function: <code>TIME_DELAY</code>	169
15.5.3	Control Function: <code>DEADBAND</code>	170
15.5.4	Control Function: <code>RESTART</code> and <code>KILL</code>	170
15.5.5	Control Function: <code>CUSTOM</code>	170
15.5.6	Control Function: Math Operations	171
15.5.7	Control Function: PID Control Function	172
15.5.8	Combining Control Functions: A Pre-Action Sprinkler System	172
15.5.9	Combining Control Functions: A Dry Pipe Sprinkler System	173
15.5.10	Example Case: <code>activate_vents</code>	173
15.6	Controlling a <code>RAMP</code>	174
15.6.1	Changing the Independent variable	174
15.6.2	Freezing the Output Value, Example Case: <code>freeze_hrr</code>	174
15.7	Visualizing FDS Devices Using Smokeview Objects	176
15.7.1	Static Smokeview Objects	176
15.7.2	Dynamic Smokeview Objects - Customized Using <code>&PROP</code> Parameters	178
15.7.3	Dynamic Smokeview Objects - Customized Using <code>&PROP</code> Parameters and Particle File Data	180
15.8	Output Control Parameters: The <code>DUMP</code> Namelist Group	182
15.9	Output File Types	184
15.9.1	Device Output: The <code>DEVC</code> Namelist Group	184
15.9.2	Quantities within Solids: The <code>PROP</code> Namelist Group	185
15.9.3	Animated Planar Slices: The <code>SLCF</code> Namelist Group	186

15.9.4	Animated Boundary Quantities: The BNDF Namelist Group	187
15.9.5	Animated Isosurfaces: The ISOOF Namelist Group	187
15.9.6	Plot3D Static Data Dumps	187
15.9.7	SMOKE3D: Realistic Smoke and Fire	188
15.10	Special Output Quantities	189
15.10.1	Heat Release Rate	189
15.10.2	Visibility and Obscuration	189
15.10.3	Layer Height and the Average Upper and Lower Layer Temperatures	190
15.10.4	Thermocouples	191
15.10.5	Heat Fluxes and Thermal Radiation	191
15.10.6	Droplet Output Quantities	192
15.10.7	Interfacing with Structural Models	195
15.10.8	Useful Solid Phase Outputs	196
15.10.9	Fractional Effective Dose (FED) and Fractional Irritant Concentration (FIC)	196
15.10.10	Spatially-Integrated Outputs	198
15.10.11	Temporally-Integrated Outputs	200
15.10.12	Wind and the Pressure Coefficient	200
15.10.13	Near-wall Grid Resolution	201
15.10.14	Dry Volume and Mass Fractions	201
15.10.15	Gas Velocity	201
15.10.16	Enthalpy	202
15.10.17	Computer Performance	202
15.10.18	Output File Precision	202
15.10.19	<i>A Posteriori</i> Mesh Quality Metrics	202
15.11	Extracting Numbers from the Output Data Files	207
15.12	Summary of Frequently-Used Output Quantities	208
15.13	Summary of Infrequently-Used Output Quantities	212
15.14	Summary of HVAC Output Quantities	213
16	Alphabetical List of Input Parameters	215
16.1	BNDF (Boundary File Parameters)	216
16.2	CLIP (MIN/MAX Clipping Parameters)	216
16.3	CSVF (Comma Delimited Output Files)	216
16.4	CTRL (Control Function Parameters)	216
16.5	DEVC (Device Parameters)	217
16.6	DUMP (Output Parameters)	218
16.7	HEAD (Header Parameters)	219
16.8	HOLE (Obstruction Cutout Parameters)	219
16.9	HVAC (HVAC System Definition)	220
16.10	INIT (Initial Conditions)	221
16.11	ISOOF (Isosurface Parameters)	221
16.12	MATL (Material Properties)	222
16.13	MESH (Mesh Parameters)	222
16.14	MISC (Miscellaneous Parameters)	223
16.15	MULT (Multiplier Function Parameters)	225
16.16	OBST (Obstruction Parameters)	225
16.17	PART (Lagrangian Particles/Droplets)	226
16.18	PRES (Pressure Solver Parameters)	227

16.19	PROF (Wall Profile Parameters)	227
16.20	PROP (Device Properties)	228
16.21	RADI (Radiation Parameters)	229
16.22	RAMP (Ramp Function Parameters)	229
16.23	REAC (Reaction Parameters)	230
16.24	SLCF (Slice File Parameters)	231
16.25	SMIX (Species Mixture (Lumped Species) Parameters)	231
16.26	SPEC (Species Parameters)	232
16.27	SURF (Surface Properties)	233
16.28	TABL (Table Parameters)	234
16.29	TIME (Time Parameters)	235
16.30	TRNX, TRNY, TRNZ (MESH Transformations)	235
16.31	VENT (Vent Parameters)	235
16.32	ZONE (Pressure Zone Parameters)	236
16.33	Modifications of Input Parameters from FDS 5 to FDS 6	237
III	FDS and Smokeview Development Tools	241
17	The FDS/Smokeview Repository	243
18	Compiling FDS	245
18.1	FDS Source Code	245
19	Output File Formats	247
19.1	Diagnostic Output	247
19.2	Heat Release Rate and Related Quantities	248
19.3	Device Output Data	248
19.4	Control Output Data	249
19.5	Gas Mass Data	249
19.6	Slice Files	249
19.7	Plot3D Data	250
19.8	Boundary Files	250
19.9	Particle Data	251
19.10	Profile Files	252
19.11	3-D Smoke Files	252
19.12	Geometry, Isosurface Files	252
19.13	Geometry Data Files	254
	Bibliography	257

List of Figures

6.1	An example of a multiple-mesh geometry.	28
6.2	Rules governing the alignment of meshes.	30
6.3	Piecewise-Linear Mesh Transformation.	32
6.4	Polynomial Mesh Transformation.	32
6.5	Axi-symmetric helium plume	38
6.6	Simple example of flow through a duct.	43
6.7	Results of the dancing eddies test case.	43
7.1	An example of the multiplier function.	48
7.2	Simple example of <code>SAWTOOTH=.FALSE.</code>	49
8.1	Simple demonstration of pyrolysis model.	72
8.2	A more complicated demonstration of the pyrolysis model.	74
8.3	Input parameters for sample case pyrolysis_2	75
8.4	Input parameters for sample case water_ice_water	76
8.5	Freezing and melting of water	77
8.6	Input parameters for sample case ethanol_pan	78
8.7	Output of box_burn_away test cases.	81
8.8	Output of room_fire test case.	85
9.1	An example simplifying a complex duct.	95
9.2	Example of a fan curve.	96
9.3	Output of the fan_test example.	96
9.4	Example of a jet fan	96
9.5	Results for the HVAC_filter sample case	98
9.6	Duct Aircoil	99
9.7	Output of pressure_rise test case.	101
9.8	Output of zone_break test cases.	102
9.9	Stack effect	105
9.10	Example of positive pressure at a tunnel entrance.	106
11.1	Example of gas filling.	115
12.1	Output of door_crack test case.	123
12.2	Product species mass fractions for model PVC example.	126
12.3	Wall soot deposition for propane flame with deposition example.	130
14.1	Simple test of particle mass flux.	139
14.2	HRR of spray burner.	144
14.3	Example of specified gas mass production from particles.	148

14.4	Example of burning vegetation.	149
14.5	Example of water cascading over solid obstructions.	150
15.1	Output of the flow rate test case.	159
15.2	Example of a beam detector.	162
15.3	Output of aspiration_detector test case.	164
15.4	Output of the <code>control_test_2</code> case.	173
15.5	Example of a vent controls.	174
15.6	Example of freezing the output of a <code>RAMP</code>	175
15.7	Output of the <code>bucket_test</code> case.	193
15.8	(Left) Resolved signal, MTR is small. (Right) Unresolved signal, MTR is close to unity. . .	203
15.9	Haar mother wavelet on the interval $[0,1]$	205
15.10	Averages and coefficients for local Haar wavelet transforms on four typical signals. . . .	206

List of Tables

4.1	FDS features with known issues or problems.	17
5.1	Namelist Group Reference Table	24
7.1	Sample of Color Definitions (A complete list is included on the website)	57
10.1	Parameters used to control time-dependence.	111
11.1	Optional Gas Species	117
15.1	Suggested Values for Smoke Detector Model.	161
15.2	Control function types for CTRL	168
15.3	Single Frame Static Objects	177
15.4	Dual Frame Static Objects	177
15.4	Dual Frame Static Objects (continued)	178
15.5	Dynamic Objects - Customized using SMOKEVIEW_PARAMETERS on a &PROP line .	179
15.5	Dynamic Objects (continued)	180
15.6	Dynamic Objects - Customized using SMOKEVIEW_PARAMETERS on a &PROP line and Particle File Data	181
15.6	Dynamic Objects (continued)	182
15.7	Output quantities available for PDPA output.	194
15.8	Coefficients used for the computation of irritant effects of gases.	197
15.9	Output quantities.	209
15.10	Output quantities.	212
15.11	Output quantities.	213
16.1	Boundary File Parameters	216
16.2	MIN/MAX Clipping Parameters	216
16.3	Comma Delimited Output Files	216
16.4	Control Function Parameters	216
16.5	Device Parameters	217
16.6	Output Parameters	218
16.7	Header Parameters	219
16.8	Obstruction Cutout Parameters	219
16.9	Initial Conditions	220
16.10	Initial Conditions	221
16.11	Isosurface Parameters	221
16.12	Material Properties	222
16.13	Mesh Parameters	223
16.14	Miscellaneous Parameters	223

16.15	Multiplier Function Parameters	225
16.16	Obstruction Parameters	225
16.17	Lagrangian Particles/Droplets	226
16.18	Pressure Solver Parameters	227
16.19	Wall Profile Parameters	227
16.20	Device Properties	228
16.21	Radiation Parameters	229
16.22	Ramp Function Parameters	230
16.23	Reaction Parameters	230
16.24	Slice File Parameters	231
16.25	Species Parameters	231
16.26	Species Parameters	232
16.27	Surface Properties	233
16.28	Table Parameters	234
16.29	Time Parameters	235
16.30	MESH Transformations	235
16.31	Vent Parameters	235
16.32	Pressure Zone Parameters	236
16.33	List of changes to input parameters for FDS 6	238
16.34	List of changes to input parameters for FDS 6	239
18.1	Source Code Files	246

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, 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 [1]. Verification and Validation of the model are discussed in the FDS Verification [3] and Validation [4] 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 the dynamic Smagorinsky form of Large Eddy Simulation (LES). It is possible to perform a Direct Numerical Simulation (DNS) if the underlying numerical mesh is fine enough. LES is the default mode of operation.

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 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

narrow-band model. 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 It is possible to run an FDS calculation on more than one computer using the Message Passing Interface (MPI). Details can be found in [Section 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).

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. Other tools related to FDS and Smokeview can be found at the web site.

2.1 How to Acquire FDS and Smokeview

Detailed instructions on how to download executables, manuals, source-code and related utilities, can be found on the FDS-SMV Website <http://fire.nist.gov/fds>. The typical FDS/Smokeview distribution consists of an installation package or compressed archive, which is available for MS Windows, Mac OS X, and Linux. For other operating systems, consult the web site.

If you ever want to keep an older version of FDS and Smokeview, copy the installation directory to some other place so that it is not overwritten during the updated installation.

2.2 Computer Hardware Requirements

FDS requires a fast CPU¹ and a substantial amount of random-access memory (RAM) to run efficiently. For minimum specifications, the system should have a 1 GHz CPU, and at least 512 MB RAM. 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. It is not unusual for the output of a single calculation to consume more than 1 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 512 MB. 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.

For Multi-Mesh calculations, the MPI version of FDS will operate over standard 100 Mbps networks. A Gigabit or 1000 Mbps network will further reduce latency and improve data transfer rates between nodes.

¹Central Processing Unit

2.3 Computer Operating System (OS) and Software Requirements

The goal of making FDS and Smokeview publicly available has been to enable practicing fire protection engineers to perform fairly sophisticated fire simulations at a reasonable cost. Thus, FDS and Smokeview have been designed for computers running Microsoft Windows, Mac OS X, and various implementations of Unix/Linux.

MS Windows An installation package is available for Windows operating system. It is not recommended to run FDS/Smokeview under any version of MS Windows released prior to Windows 2000.

Mac OS X A zip archive is available for Intel architectures. Mac OS X 10.4.x or better is recommended, versions of OS X prior to 10.4.x are not officially supported. Users can always download the latest version of FDS source and compile FDS for other versions of OS X (See Appendix 18 for details).

Linux Pre-compiled executables are available that can be installed in an appropriate directory. Note that the installation package is simply an archive and no path variables are set. 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 Appendix 18 for details). If Smokeview does not work on the Linux workstation, you can use the Windows version to view FDS output.

Unix There are no pre-compiled versions of FDS for the various flavors of Unix. However, the advice for Linux applies equally as well to Unix.

FDS in Parallel For those wishing to use multiple computers to run a single FDS calculation, MPI (Message Passing Interface) must be installed on each of the computers within the network that will be used for FDS computations.

Chapter 3

Running FDS

This chapter describes the procedure to run an FDS calculation. The primary requirement for any calculation is an FDS input file. The creation of an input file is covered in detail in Part II. If you are new to FDS and Smokeview, it is strongly suggested that you start with an existing data file, run it as is, and then make the appropriate changes to the input file for the desired scenario. Sample input files are included as part of the standard installation. 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.

3.1 Starting an FDS Calculation

FDS can be run from the command prompt, or with a third party Graphical User Interface (GUI). In the discussion to follow, it is assumed that FDS is being run from the command prompt. FDS can be run on a single computer, using only one CPU, or it can be run on multiple computers and use multiple CPUs. For any operating system, there are two FDS executable files. The single CPU 32 bit Windows executable is called **fds_win_32.exe**. The parallel executable is called **fds_mpi_win_32.exe**. The letters “mpi” in the filename denote Message Passing Interface (MPI), which will be discussed below.

Note that the input file for both single and parallel versions of FDS are the same. In fact, it is recommended that before embarking on parallel processing, you should run your input file in serial mode to ensure that it is properly set up.

3.1.1 Starting an FDS Calculation (Single Processor Version)

Sample input files are provided with the program for new users who are encouraged to first run a sample calculation before attempting to write an input file. Input files are typically given names that help identify the particular case, followed by the suffix `.fds`. Suppose you want to run an input file called **job_name.fds**. You can start FDS from a DOS or Linux/Unix command prompt as follows:

MS Windows

Open up a Command Prompt window (click Start, then Run, then type “cmd”), and change directories (“cd”) to where the input file for the case is located, then run the code by typing at the command prompt

```
fds_win_32 job_name.fds
```

On a 64 bit machine, substitute 64 for 32. The character string `job_name` is usually designated within the input file as the `CHID`. It is recommended that the name of the input file and the `CHID` be the same so that all of the files associated with a given calculation have a consistent name. The progress of a simulation is indicated by diagnostic output that is written out onto the screen. Detailed diagnostic information is automatically written to a file **CHID.out**, where `CHID` is a character string, usually the same as `job_name`, designated in the input file. Screen output can be redirected to a file via the alternative command

```
fds_win_32 job_name.fds > job_name.err
```

Note that it is also possible to associate the extension “`fds`” with the FDS executable directly, thereby making FDS run by double-clicking on the input file. If you do this, note that error messages will be written to the file called `job_name.out`. Also, if you associate the input file with the FDS executable, be careful not to accidentally double-click on the input file when trying to edit it. This action will cause previously generated output files to be over-written.

Mac OS X, Unix, Linux

Depending on the type of installation, you may need to set various path or environment variables in order to invoke FDS without a full path reference to the executable. The easiest way to do this is via an “alias” in your shell start-up script. For the example below, it is assumed that `fds` is an alias for the specific executable used. You may also need to “`chmod + x`” to make the file executable. Once this is done, run FDS from the command line by typing:

```
fds job_name.fds
```

The input parameters are read from the file **job_name.fds**, and error statements and other diagnostics are written out to the screen. To run the job in the background:

```
fds job_name.fds >& job_name.err &
```

Note that in the latter case, the screen output is stored in the file **job_name.err** and the detailed diagnostics are saved automatically in a file **CHID.out**, where `CHID` is a character string, usually the same as `job_name`, designated in the input file. It is preferable to run jobs in the background so as to free the console for other uses.

3.1.2 Starting an FDS Calculation (Multiple Processor Version)

Running FDS across a network using multiple processors and multiple banks of memory (RAM) is more complicated than running the single processor version. More is required of the user to make the connections between the machines as seamless as possible. This involves creating accounts for a given user on each machine, sharing directories, increasing the speed of the network, making each machine aware of the others, *etc.* Some of these details are handled by the parallel-processing software, others are not. Undoubtedly the procedure will be simplified in years to come, but for the moment, parallel-processing is still relatively new and requires more expertise in terms of understanding both the operating system and the network connections of a given set of computers.

FDS uses MPI (Message-Passing Interface) [5] to allow multiple computers to run a single FDS job. The main idea is that you must break up the FDS domain into multiple meshes, and then the flow field in each mesh is computed as a different *process*. Note the subtle difference between these terms – a *process* does not have the same meaning as a *processor*. 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. The *processor* refers to the computer hardware. A single *processor* may run multiple *processes*, for example. The computation

on a given FDS mesh is thought of as an individual *process*, and MPI handles the transfer of information between these *processes*. Usually, each mesh is assigned its own *process* in a parallel calculation, although it is also possible assign multiple meshes to a single *process*. In this way, large meshes can be computed on dedicated *processors*, while smaller meshes can be clustered together in a single *process* running on a single *processor*, without the need for MPI message passing between themselves.

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. As a user, 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 five mesh FDS case is run in parallel, the first printout (usually to the screen unless otherwise directed) is:

```
Process 4 of 4 is running on fire65
Process 3 of 4 is running on fire64
Process 2 of 4 is running on fire63
Process 0 of 4 is running on fire61
Process 1 of 4 is running on fire62
Mesh 1 is assigned to Process 0
Mesh 2 is assigned to Process 1
Mesh 3 is assigned to Process 2
Mesh 4 is assigned to Process 3
Mesh 5 is assigned to Process 4
```

This means that 5 processes (numbered 0 to 4) have started on the computers named fire61, fire62, *etc.*, and that each mesh is being computed as an individual process on the individual computers. Each computer has its own memory (RAM), and MPI is the protocol by which information is passed from process to process during the calculation. Note that these computers may have multiple processors, and each processor may have multiple “cores.” You have control over how many processes get assigned to each computer, but you may or may not have control over how the processes are handled by a given computer. That depends on the operating system and the particular version of MPI. For example, the computer named fire62 happens to have two quad-core processors, and all five meshes could have been assigned to run as five individual processes all on fire62. Whether or not this is the best strategy is still a subject of research and heavily dependent on the technical specifications of the OS and hardware.

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.

The way FDS is executed in parallel depends on which implementation of MPI has been installed. At NIST, the parallel version of FDS is presently run on Windows PCs connected by the Local Area Network (LAN, 100 Mbps) or on a cluster of Linux PCs linked together with a dedicated, fast (1000 Mbps) network. The Windows computers use MPICH2, a free implementation of MPI from Argonne National Laboratory, USA.

MPICH2

With MPICH2, a parallel FDS calculation can be invoked either from the command line or by using a Graphical User Interface (GUI). After the MPICH2 libraries are installed on each computer and the necessary directories are shared, FDS is run using the command issued from one of the computers

```
mpiexec -file config.txt
```

where **config.txt** is a text file containing the name and location of the FDS executable, name of the FDS input file, the working directory, and the names of the various computers that are to run the job. For example, the **config.txt** file might look like this for a job run at NIST with computers named fire_1, fire_2, and fire_3:

```
exe \\fire_1.nist.gov\NIST\FDS\fds_mpi_win_32.exe job_name.fds
dir \\fire_1.nist.gov\Projects\
hosts
fire_1.nist.gov 2
fire_2.nist.gov 1
fire_3.nist.gov 2
```

The numbers following the “host” machines represent the number of threads to run on that particular machine. In this example, 5 threads are run for an FDS calculation that has 5 meshes. The **exe** and **dir** directories need to be shared, with the latter having read and write permissions.

All the computers must be able to access the executable and the working directory on **fire_1**. This is achieved under Windows by “sharing.” Under Unix/Linux and OS X, the process involves cross-mounting the file systems of the various machines.

Open MPI

Both at VTT and NIST, we use Open MPI, an open source MPI-2 implementation that is developed and maintained by a consortium of academic, research, and industry partners. With Open MPI, FDS is run using the command

```
mpirun -np 5 fds_mpi_linux_64 job_name.fds
```

where the 5 indicates that 5 processors are to be used. In this case, the executable **fds_mpi_linux_64** is located in the working directory, but you can invoke it by its full path name. To make the process run in the background

```
mpirun -np 5 fds_mpi_linux_32 job_name.fds >& job_name.err &
```

The file **job_name.err** contains what is normally printed out to the screen.

Note that there are several other implementations of MPI, some free, some not. Support for the software varies, thus FDS has been designed to run under any of the more popular versions without too much user intervention. However, keep in mind that parallel processing is a relatively new area of computer science, and there are bound to be painful growth spurts in the years ahead.

3.2 Monitoring Progress

Diagnostics for a given calculation are written into a file called **CHID.out**. The CPU usage and simulation time are written here, so you can see how far along the program has progressed. At any time during a calculation, Smokeview can be run and the progress can be checked visually. To stop a calculation before its scheduled time, either kill the process, or preferably 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 be hours or days long, there is a restart feature in FDS. Details of how to use this feature are given in Section [6.4.4](#). 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 dumping of restart files by using control functions as described in Section [15.5](#).

Chapter 4

User Support

It is not unusual over the course of a project to run into various problems, some related to FDS, some related to your computer. FDS is not a typical PC application. It is an intensive calculation that can push your 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. It is difficult to predict at the start of a simulation just how long and how much memory will be required. 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 include an "Issue Tracker," that allows you to report bugs, request new features, and ask specific clarifying questions, and "Group Discussions," which support more general topics than just specific problems. 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

If you encounter problems with FDS, it is crucial that you submit, along with a description of the problem, the FDS version number. Each release of FDS comes with a version number like 5.2.6, 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 effect 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 SVN number (useful to us). The SVN number refers to the Subversion repository number of the source code. It allows us to go back in time

and recover the exact source code files that were used to build that 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 mis-typed 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 right type of information is being provided for each parameter, like whether one real number is expected, or several integers, or whatever. 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 zeros are zeros and O’s are O’s. Make sure l’s are not !’s. Make sure apostrophes are used to designate character strings. Make sure the text file on a Unix/Linux machine was not created on a DOS machine, and *vice versa*. Make sure that all the parameters listed are still being used – new versions of FDS often drop or change parameters to force you to re-examine old input files.

Numerical Instability Errors: It is possible that during an FDS calculation the flow velocity at some location in the domain can increase due to numerical error causing 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 the user some means by which to see 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, high speed flow through a small opening, a sudden change in the heat release rate, or any number of sudden changes to the flow field. There are various ways to solve the problem, depending on the situation. Try to diagnose and fix the problem before reporting it. It is difficult for anyone but the originator of the input file 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. In fact, for a typical 32 bit Windows PC with 4 GB RAM, only 2 GB will be available to FDS, based on user feedback. If you want to run bigger cases, consider buying a computer with a 64 bit

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 DT_LIMITING_RATIO.

operating system or break up the calculation into multiple meshes and use parallel processing. 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 your 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.

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:

```
fortrtl: 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=.FALSE.
```

on the `DUMP` line of the input file (see Section 15.8). 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.

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. 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 need to be investigated first by the user before being reported. 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 problem input file and hopefully diagnose the problem.

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

4.4 Known Issues

As a result of collecting feedback from FDS users over roughly a decade, we have identified a number of features in FDS that can be problematic for a variety of reasons. Table [4.1](#) lists these features that are either under development, or that have been cited a number of times by users who have observed spurious behavior, inconsistent or inaccurate results, fragility, and so on. For those interested in FDS model development, this list is ripe for further research. For those who use FDS for engineering applications, these may be features to avoid until they can be made more reliable and robust.

Table 4.1: Parameters or features known to have problems related to accuracy, numerical stability, robustness, sensitivity, and so on.

Feature	Description	Symptom of Problem	Recommendation
Liquid Fuels	Pyrolysis model of evaporating liquid fuel	Inaccuracies found in comparison to experiments; physical and numerical sensitivity	Research usage only
Solid Fuels	Pyrolysis model of solid fuel	Results in FDS 5.4 and beyond different than previous versions	Read Section 8.4.4

Part II

Writing an FDS Input File

Chapter 5

The Basic Structure of an Input File

5.1 Naming the Job

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 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 `/` 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. Some machines do not recognize the spaces or the length of the tab stops. Comments and notes can be written into the file so long as nothing comes before the `&` except a space and nothing comes between the ampersand `&` and the slash `/` except appropriate parameters corresponding to that particular namelist group.

¹ASCII – American Standard Code for Information Interchange. There are 256 characters that make up the standard ASCII text.

²A *namelist* is a Fortran input record.

The parameters in the input file can be integers ($T_END=5400$), real numbers ($CO_YIELD=0.008$), groups of real numbers or integers ($XYZ=6.04, 0.28, 3.65$) or ($IJK=90, 36, 38$), character strings:

```
CHID='WTC_05_v5'
```

groups of character strings:

```
SURF_IDS='burner','STEEL','BRICK'
```

or logical parameters:

```
POROUS_FLOOR=.FALSE.
```

A logical parameter is either `.TRUE.` or `.FALSE.` – the periods are a Fortran convention. Character strings that are listed in this User's Manual must be copied exactly as written – the code is case sensitive and underscores *do* matter.

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 element 1 through 3, inclusive.

Note that character strings can be enclosed either by apostrophes or 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.

Note that depending on compiler and operating system, 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. It does not appear that current Fortran compilers support the UTF-8 encoding standard for reading Namelist inputs.

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 to the file parameters that are to change from their 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.

```
&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      = '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=.TRUE. /
...
&BNDF QUANTITY='GAUGE HEAT FLUX' /
...
&DEVC XYZ=6.04,0.28,3.65, QUANTITY='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 resolution to determine if the geometry 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.

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

Table 5.1: Namelist Group Reference Table

Group Name	Namelist Group Description	Reference Section	Parameter Table
BNDF	Boundary File Output	15.9.4	16.1
CLIP	Min/Max Clipping Parameters	6.7	16.2
CTRL	Control Function Parameters	15.5	16.4
DEVC	Device Parameters	15.1	16.5
DUMP	Output Parameters	15.8	16.6
HEAD	Input File Header	6.1	16.7
HOLE	Obstruction Cutout	7.3	16.8
HVAC	Heating, Vent., Air Cond.	9.2	16.9
INIT	Initial Condition	6.5	16.10
ISOF	Isosurface File Output	15.9.5	16.11
MATL	Material Property	8.3	16.12
MESH	Mesh Parameters	6.3	16.13
MISC	Miscellaneous	6.4	16.14
MULT	Multiplier Parameters	7.2.2	16.15
OBST	Obstruction	7.2	16.16
PART	Lagrangian Particle	14	16.17
PRES	Pressure Solver Parameters	6.6	16.18
PROF	Profile Output	15.9.2	16.19
PROP	Device Property	15.3	16.20
RADI	Radiation	13.1	16.21
RAMP	Ramp Profile	10	16.22
REAC	Reactions	12	16.23
SLCF	Slice File Output	15.9.3	16.24
SMIX	Species Mixtures	11.2	16.25
SPEC	Species Parameters	11.1	16.26
SURF	Surface Properties	7.1	16.27
TABL	Tabulated Particle Data	10.3	16.28
TIME	Simulation Time	6.2	16.29
TRNX	Mesh Stretching	6.3.5	16.30
VENT	Vent Parameters	7.4	16.31
ZONE	Pressure Zone Parameters	9.3	16.32

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 `HEAD` Namelist Group (Table 16.7)

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` is a string of 30 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_v5.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 60 characters or less that describes the simulation. It is simply descriptive text that is passed to various output files.

6.2 Simulation Time: The `TIME` Namelist Group (Table 16.29)

`TIME` is the name of a group of parameters time 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 seconds.

```
&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.

If you want the time line to start at a number other than zero, you can 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.

Time-based RAMPs are evaluated using the actual time if the RAMP activation time is the same as `T_BEGIN`; 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, `T_BEGIN` should be set slightly less than 10 s.

6.2.2 Special Topic: 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 $5(\delta x \delta y \delta z)^{\frac{1}{3}} / \sqrt{gH}$ 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. Note that by default the time step is never allowed to increase above its initial value. To allow this to happen, set `RESTRICT_TIME_STEP=.FALSE.`

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

One additional parameter in the `TIME` group is `SYNCHRONIZE`, a logical flag (`.TRUE.` or `.FALSE.`) indicating that in a multi-mesh computation the time step for each mesh should be the same, thus ensuring that each mesh is processed each iteration. More details can be found in Section 6.3.3. The default value of `SYNCHRONIZE` is `.TRUE.`

Finally, if you want to prevent FDS from automatically changing the time step, set `LOCK_TIME_STEP=.TRUE.`

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.

6.2.3 Special Topic: Steady-State Applications

Occasionally, there are applications in which only the steady-state solution (in a time-averaged sense) is desired. However, the time necessary to heat the walls to steady-state can make the cost of the calculation prohibitive. In these situations, if you specify a `TIME_SHRINK_FACTOR` of, say, 10, the specific heats of the various materials is reduced by a factor of 10, speeding up the heating of these materials roughly by 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.

6.3 Computational Meshes: The MESH Namelist Group (Table 16.13)

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. If it is desired that the mesh cells in a particular direction not be uniform in size, then the namelist groups TRNX, TRNY and/or TRNZ may be used to alter the uniformity of the mesh (See Section 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.

Note that 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.

Because an important part of the calculation uses a Poisson solver based on Fast Fourier Transforms (FFTs) in the y and z directions, the second and third dimensions of the mesh should each be of the form $2^l 3^m 5^n$, where l , m and n are integers. For example, $64 = 2^6$, $72 = 2^3 3^2$ and $108 = 2^2 3^3$ are good mesh cell divisions, but 37, 99 and 109 are not. The first number of mesh cell divisions (the I in IJK) does not use FFTs and need not be given as a product of small numbers. However, you should experiment with different values of divisions to ensure that those that are ultimately used do not unduly slow down the calculation.

Here is a list of numbers between 1 and 1024 that can be factored down to 2's, 3's and 5's:

2	3	4	5	6	8	9	10	12	15	16	18	20	24	25
27	30	32	36	40	45	48	50	54	60	64	72	75	80	81
90	96	100	108	120	125	128	135	144	150	160	162	180	192	200
216	225	240	243	250	256	270	288	300	320	324	360	375	384	400
405	432	450	480	486	500	512	540	576	600	625	640	648	675	720
729	750	768	800	810	864	900	960	972	1000	1024				

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=.TRUE. to the MESH line, and the coordinate x is then interpreted as the radial coordinate r . 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 in which $r = XB(1) = 0$. 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 (**helium_2d**) is given in Section 6.4.7.

6.3.3 Multiple Meshes and Parallel Processing

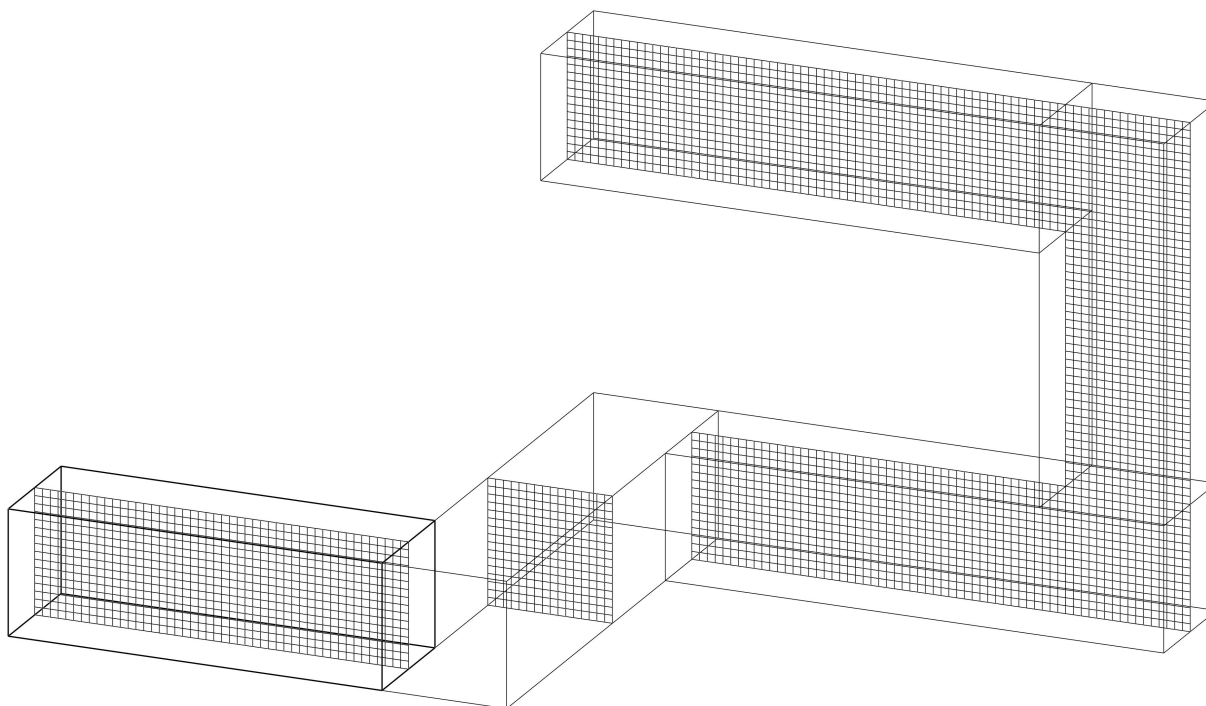


Figure 6.1: An example of a multiple-mesh geometry.

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.

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 available processors. In general, it is better to run multiple mesh cases with the parallel version of FDS if you have the computers available, but be aware that two computers will not necessarily finish the job in half the time as one. For the parallel version to work well, there has to be a comparable number of cells in each mesh, or otherwise most of the computers will sit idle waiting for the one with the largest mesh to finish processing each time step. You can use multiple meshes even when running the serial version of FDS, in which case one CPU will serially process each mesh, one by one. Why do this? For one, if you set `SYNCHRONIZE=.FALSE.` on the `TIME` line, then in each mesh, the governing equations will be solved with a time step based on the flow speed within that particular mesh. Because each mesh can have different time steps, this technique can save CPU time by requiring relatively coarse meshes to be updated only when necessary. Coarse meshes are best used in

regions where temporal and spatial gradients of key quantities are small or unimportant. Be aware, however, that unsynchronized time steps are more likely to lead to numerical instabilities.

By default, the time steps in each mesh are synchronized. With this setting, all meshes are active each iteration. For a single-processor, multiple mesh calculation, this strategy reduces and may even eliminate any benefit seen by using multiple meshes. However, in a parallel calculation, if a particular mesh is inactive during an iteration because it is not ready to be updated, then the processor assigned to that mesh is also inactive. Forcing the mesh to be updated with a smaller than ideal time step does not cost anything since that processor would have been idle anyway. The benefit is that there is a tighter connection between meshes. It is also possible to synchronize the time step in only a select set of meshes. To do this, add `SYNCHRONIZE=.TRUE.` to the appropriate `MESH` lines and then add `SYNCHRONIZE=.FALSE.` to the `TIME` line.

Usually in a multi-mesh calculation, each mesh is assigned its own process, and each process its own processor. However, it is possible to assign more than one mesh to a single process, and it is possible to assign more than one process to a single processor. 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.

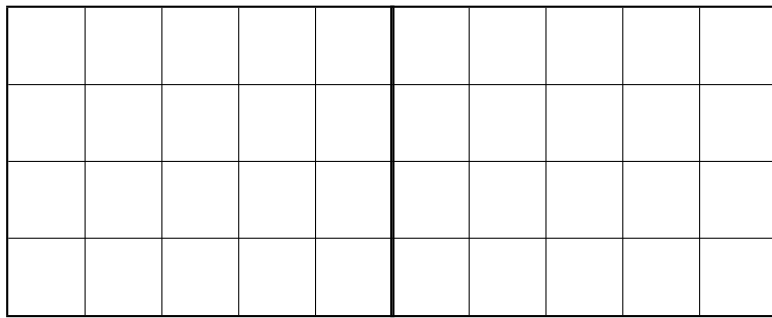
For cases involving many meshes, you might want to assign them colors using either the character string `COLOR` or the integer triplet `RGB`. You may also want to consider using the multiplying feature to easily create a 3-D array of meshes. See Section 7.2.2 for details.

Some parallel computing environments do not have a centralized file system, in which case FDS must write the output files for each process to a separate disk. If your computing cluster does not have a `SHARED_FILE_SYSTEM`, then set this parameter to `.FALSE.` on the `MISC` line.

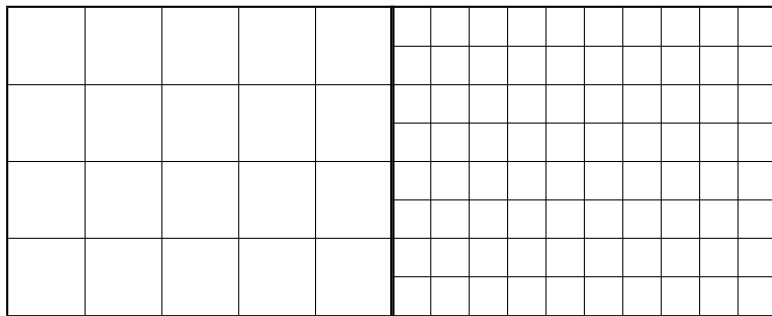
6.3.4 Mesh Alignment

Whether the calculation is to be run on a single processor, or on multiple processors, 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. 6.2. The following rules of thumb should also be followed when setting up a multiple mesh calculation:

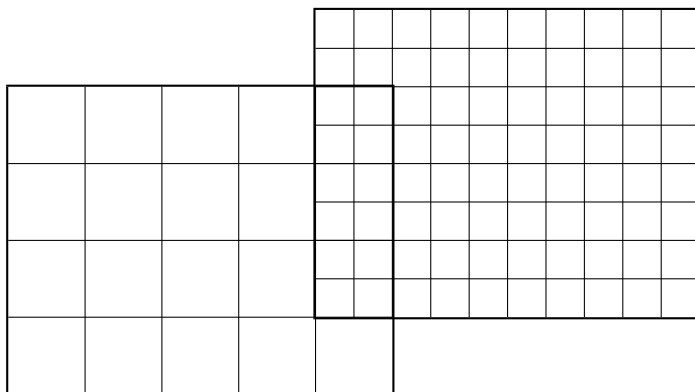
- Avoid putting mesh boundaries where critical action is expected, especially fire. Sometimes fire spread from mesh to mesh cannot be avoided, but if at all possible try to keep mesh interfaces relatively free of complicated phenomena since the exchange of information across mesh boundaries is not yet as accurate as cell to cell exchanges within one mesh.



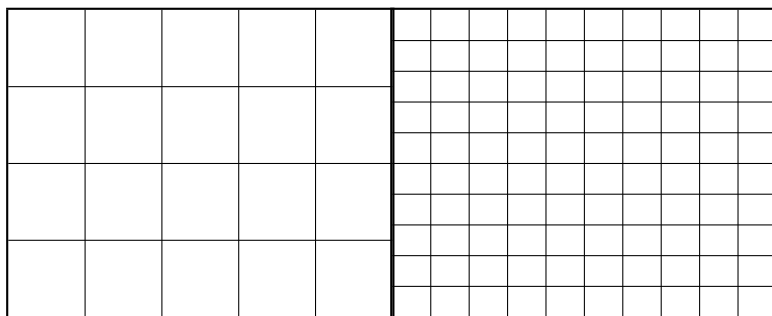
This is the ideal kind of mesh to mesh alignment.



This is allowed so long as there are an integral number of fine cells abutting each coarse cell.



This is allowed, but of questionable value.



This is no longer allowed in FDS 5.1 and higher.

Figure 6.2: 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 Section 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.

Accuracy of the Parallel 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, try to move the mesh boundaries away from areas of activity. 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.

Efficiency of the Parallel Calculation

When running a case with multiple meshes in parallel, the efficiency of the calculation can be checked as follows: (1) Set `SYNCHRONIZE=.TRUE.` on the `TIME` line, (2) Let the program run several hundred time steps, (3) Calculate the difference in wall clock time between two 100 iteration print outs in the file **CHID.out** (see Section 19.1). Divide the time difference by 100. This is the average elapsed wall clock time per time step, (4) Look at the `CPU/step` for each mesh. The largest value should be less than, but close to, the average elapsed wall clock time. The efficiency of the parallel calculation is the maximum `CPU/step` divided by the average wall clock time per step. If this number is between 90 % and 100 %, the parallel code is working well.

6.3.5 Mesh Stretching: The `TRNX`, `TRNY` and/or `TRNZ` Namelist Groups (Table 16.30)

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 two 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 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.

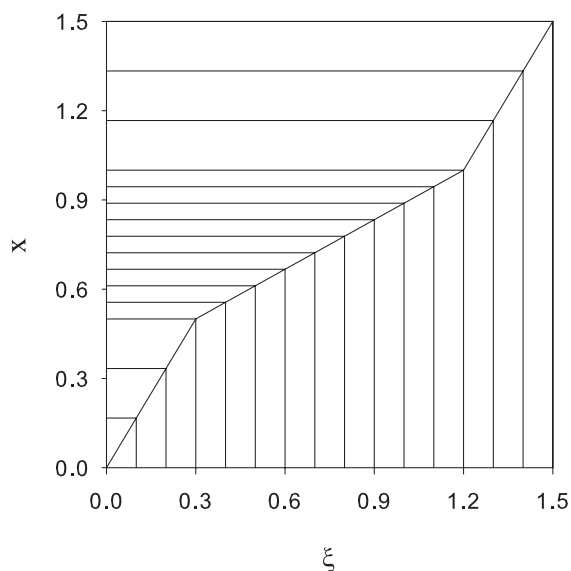


Figure 6.3: Piecewise-Linear Mesh Transformation.

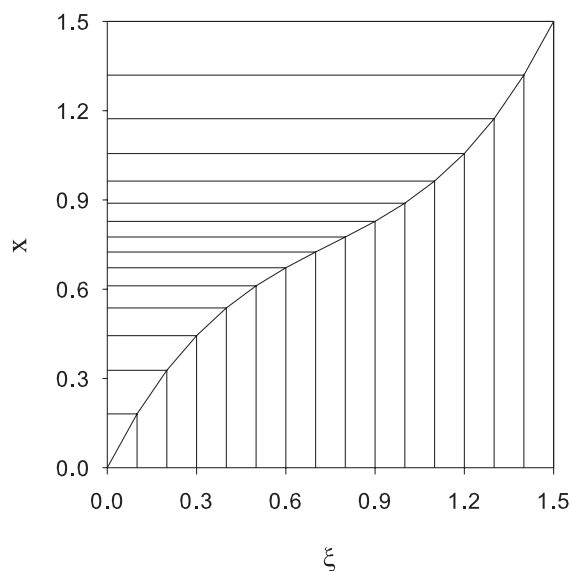


Figure 6.4: Polynomial Mesh Transformation.

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 6.3 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:

```
&TRNX CC=0.30, PC=0.50, MESH_NUMBER=2 /
&TRNX CC=1.20, PC=1.00, MESH_NUMBER=2 /
```

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 backwards 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.

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

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

Figure 6.4 gives an example of a polynomial transformation, for which the parameters are specified (assuming that this is the third mesh):

```
&TRNX IDERIV=0, CC=0.75, PC=0.75, MESH_NUMBER=3 /
&TRNX IDERIV=1, CC=0.75, PC=0.50, MESH_NUMBER=3 /
```

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.

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

$$D^* = \left(\frac{\dot{Q}}{\rho_\infty c_p T_\infty \sqrt{g}} \right)^{\frac{2}{5}} \quad (6.1)$$

and δx is the nominal size of a mesh cell¹. 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 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 source².

There are a number of special output quantities that provide local measures of grid resolution. See Section 15.10.19 for details.

¹The 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.

²For the validation study sponsored by the U.S. Nuclear Regulatory Commission [6], the $D^*/\delta x$ values ranged from 4 to 16.

6.4 Miscellaneous Parameters: The MISC Namelist Group (Table 16.14)

MISC is the namelist group of global miscellaneous input parameters. It contains parameters that do not logically fit into any other category.

6.4.1 Basics

Only one MISC line should be entered in the data file. For example, the input line

```
&MISC TMPA=25. /
```

sets the ambient temperature at 25 °C.

The MISC parameters vary in scope and degree of importance. Here is a partial list of MISCellaneous parameters. Others are described where necessary throughout this guide.

DNS A logical parameter that, if `.TRUE.`, directs FDS to perform a Direct Numerical Simulation, as opposed to the default Large Eddy Simulation (LES). This feature is appropriate only for simulations that use mesh cells that are on the order of a millimeter or less in size, or for diagnostic purposes.

GVEC The 3 components of gravity, in m/s^2 . The default is `GVEC=0, 0, -9.81`.

NOISE 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. To turn this off, set `NOISE=.FALSE.` To control the amount of noise, set `NOISE_VELOCITY`. Its default value is 0.005 m/s.

OVERWRITE If `.FALSE.` FDS checks for the existence of `CHID.out` and stops execution if it exists.

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.

U0, V0, W0 Initial values of the gas velocity in each of the coordinate directions. Normally, these are all 0 m/s, but there are a few applications where it is convenient to start the flow immediately, like in an outdoor simulation involving wind.

6.4.2 Special Topic: Mean Forcing and Data Assimilation

A situation that occurs often in atmospheric flows is that initial and boundary conditions are not well defined. Usually, one knows only that the mean wind is 10 m/s in the northeast direction, for example. More generally, there may be weather stations located at specific locations within the domain which continuously gather wind speed and direction. The process of steering the solution of the mass, momentum, and energy equations to match the statistics of the data gathered at the weather stations is known as *data assimilation*. This branch of modeling is early in its development, but very sophisticated (translation: *complicated*) methods already exist [7] and are employed in operational weather forecasting models.

In FDS, the user may invoke the most rudimentary of data assimilation techniques, a method called *nudging*. In brief, we add a mean forcing term to the momentum equation to nudge the solution toward a desired result. Currently, FDS can only affect the mean flow velocities. To turn on this capability, set the logical `MEAN_FORCING(1:3)=.TRUE.` on MISC. When this is set, FDS will drive the mean velocity toward the value of `U0`, `V0`, or `W0` (also set on MISC). For example, to set a northeast wind at 10 m/s use

```
&MISC MEAN_FORCING(1:2)=.TRUE.,.TRUE., U0=7.07, V0=7.07 /
```

For an outdoor flow, all other boundaries (except the ground) should be set to OPEN.

6.4.3 Special Topic: Specified Force Field

Similar to the MEAN_FORCING feature, the user may specify a constant and uniform force per unit volume by setting FORCE_VECTOR(1:3) on MISC. This is useful, for example, in specifying a mean pressure drop in a duct. In the absence of other forces, the force vector F_i affects the momentum equation by

$$\frac{\partial u_i}{\partial t} = F_i/\rho \quad (6.2)$$

6.4.4 Special Topic: Stopping and Restarting Calculations

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. 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 final Plot3D file and 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=.TRUE. 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, add RESTART=.TRUE. 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 old. 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 the file **CHID.out** that was created by the original job. All of the other output files from the original run are appended as well.

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 no restart files are created unless you gracefully stop a job by creating a dummy file called **CHID.stop**.

It is also possible to use the new control function feature (see Section 15.5) to stop a calculation or dump a restart file when the computation reaches some measurable condition such as a 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. 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.

6.4.5 Special Topic: Initializing a 3D Velocity Field

It may be useful to start a calculation from an established flow field. Usually this can be accomplished with the normal restart functionality. But in some circumstances restart may be fragile, or the user may want

to specify a profile throughout the entire domain. For such situations we have added the ability to read the velocity field information from a comma-delimited (.csv) file. The user has the option of creating the velocity file using FDS or creating their own. To generate the a velocity initialization file with FDS, in the input file add a DUMP line with a UVW_TIMER (time in seconds). The timer will accept up to 10 values, and will write velocity files for each mesh and each timer index to the working directory. For example, if the user wants to write the simulation velocity field at 10 minutes into the run, they would add the following:

```
&DUMP UVW_TIMER(1)=600 /
```

FDS will then write **CHID_uvw_nm.csv** for each time index and mesh. The format for this file is

```
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)
    ENDDO
  ENDDO
ENDDO
```

The user may read in the 3D velocity field using a CSVF line. For example:

```
&CSVF UVWFILE='my_velocity_field.csv' /
```

If multiple meshes are involved, it is assumed that the CSVF lines are provided in the input file in the same order as the meshes.

6.4.6 Special Topic: Defying Gravity

Most users of FDS assume that the acceleration of 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 the Space Shuttle or International Space Station, 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 Section 10, 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 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 .

6.4.7 Special Topic: The Baroclinic Vorticity

The pressure term in the momentum transport equation solved by FDS is decomposed as follows:

$$\frac{1}{\rho} \nabla \tilde{p} = \nabla \left(\frac{\tilde{p}}{\rho} \right) - \tilde{p} \nabla \left(\frac{1}{\rho} \right) \quad (6.3)$$

The pressure term is written like this so that a separable elliptic partial differential equation can be solved for the “total” pressure, $\mathcal{H} \equiv |\mathbf{u}|^2/2 + \tilde{p}/\rho$, using a direct solver. The second term is calculated based on the pressure field from the previous time step, a slight approximation necessary to render the pressure equation separable. This term 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=.FALSE.` 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. [8].

Example Case: Flowfields/helium_2d

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 Figure 6.5.

6.4.8 Special Topic: Large Eddy Simulation Parameters

By default FDS uses the Deardorff [9, 10] turbulent viscosity,

$$(\mu_{\text{LES}}/\rho) = C_v \Delta \sqrt{k_{\text{sgs}}} \quad (6.4)$$

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 [1]). Options for the turbulent viscosity model are (set `TURBULENCE_MODEL` on `MISC`):

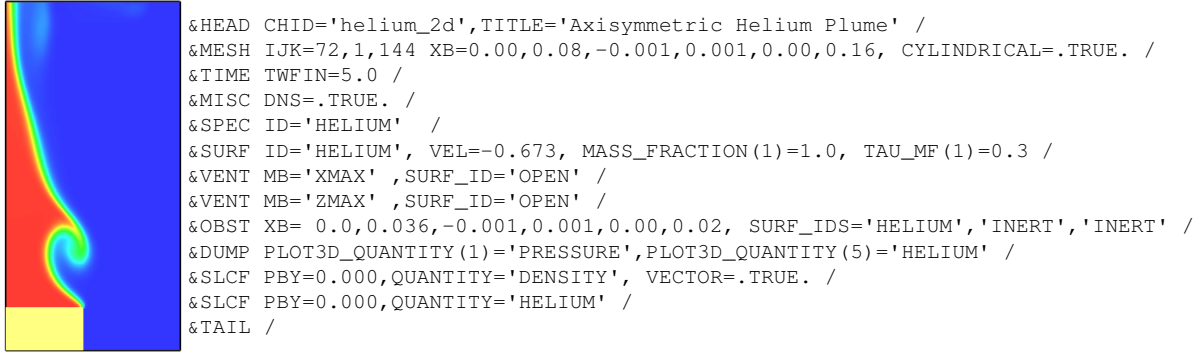


Figure 6.5: Simulation of a helium plume.

TURBULENCE_MODEL	Description	Coefficient
'CONSTANT SMAGORINSKY'	Constant coefficient Smagorinsky model [11]	C_SMAGORINSKY
'DYNAMIC SMAGORINSKY'	Dynamic Smagorinsky model [12, 13]	None
'DEARDORFF'	Deardorff model [9, 10]	C_DEARDORFF
'VREMAN'	Vreman's eddy viscosity model [14]	C_VREMAN

The other diffusive parameters, the thermal conductivity and material diffusivity, are related to the turbulent viscosity by

$$k_{LES} = \frac{\mu_{LES} c_p}{Pr_t} \quad ; \quad (\rho D)_{LES} = \frac{\mu_{LES}}{Sc_t} \quad (6.5)$$

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 $Pr_t = 0.5$, and $Sc_t = 0.5$ 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 [1].

To return to the non-dynamic form of LES used in FDS versions 1-5, set `TURBULENCE_MODEL='CONSTANT SMAGORINSKY'` on the `MISC` line.

6.4.9 Special Topic: Numerical Stability Parameters

In FDS (which uses a explicit time advancement schemes), stability criteria may be understood in terms of using the time step to maintain physically realizable conditions. Below we examine the necessary conditions for stability in the presence of advection, diffusion, and expansion of the velocity and scalar fields.

The Courant-Friedrichs-Lewy (CFL) Constraint

The well-known CFL constraint given by

$$CFL = \delta t \frac{\|\mathbf{u}\|}{\delta x} < 1 \quad (6.6)$$

places a restriction on the time step due to the advection velocity. The limits for the CFL are set by `CFL_MIN` and `CFL_MAX` on `MISC`. Physically, the constraint says that a fluid element should not traverse more than one cell within a time step. 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 δx , the time

step needs to be in concert with the CFL. If one were to employ an implicit scheme for purpose of taking time steps say 10 times larger than the CFL limit, the smallest resolvable turbulent motions would then be roughly 10 times the grid spacing, which would severely limit the benefit of the LES. In most cases, if one wishes 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. Three choices for velocity norm are available in FDS (set on MISC):

CFL_VELOCITY_NORM=0 (FDS 5 default, least restrictive, corresponds to L_∞ norm of velocity vector)

$$\frac{\|\mathbf{u}\|}{\delta x} = \max \left(\frac{|u|}{\delta x}, \frac{|v|}{\delta y}, \frac{|w|}{\delta z} \right) \quad (6.7)$$

CFL_VELOCITY_NORM=1 (FDS 6 default, most restrictive, corresponds to L_1 norm of velocity vector)

$$\frac{\|\mathbf{u}\|}{\delta x} = \frac{|u|}{\delta x} + \frac{|v|}{\delta y} + \frac{|w|}{\delta z} \quad (6.8)$$

CFL_VELOCITY_NORM=2 (L_2 norm of velocity vector)

$$\frac{\|\mathbf{u}\|}{\delta x} = \sqrt{(u/\delta x)^2 + (v/\delta y)^2 + (w/\delta z)^2} \quad (6.9)$$

The Von Neumann Constraint

The Von Neumann constraint is given by

$$\text{VN} = \delta t \max[(\mu/\rho), D_\alpha] \sum_i \frac{1}{\delta x_i^2} < \frac{1}{2} \quad (6.10)$$

The limits for VN may be adjusted using VN_MIN and VN_MAX 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 , $V_{\alpha,i} = -D_\alpha \partial Y_\alpha / \partial x_i$, 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 1D. Here subscripts indicate grid indices and ν is the diffusivity.

$$u_i^{n+1} = u_i^n + \frac{\delta t \nu}{\delta x^2} (u_{i-1}^n - 2u_i^n + u_{i+1}^n) \quad (6.11)$$

Very simply, notice that if $\delta t \nu / \delta x^2 = 1/2$ 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”, else the neighboring cell values may be shifting 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.

Realizable Mass Density Constraint

In an explicit Euler update of the continuity equation, if the time increment is too large the computational 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:

$$\delta t < \frac{\rho^n}{\bar{\mathbf{u}}^n \cdot \nabla \rho^n + \rho^n \nabla \cdot \mathbf{u}^n} \quad (6.12)$$

We can argue that the case we are most concerned with is when ρ^n is near zero. A reasonable approximation to (6.12) then becomes (time location suppressed)

$$\delta t < \frac{\rho}{\bar{u}_i \left(\frac{\rho-0}{\delta x_i} \right) + \rho \nabla \cdot \mathbf{u}} = \left[\frac{\bar{u}_i}{\delta x_i} + \nabla \cdot \mathbf{u} \right]^{-1} \quad (6.13)$$

Eq. (6.13) basically adds the effect of thermal expansion to the CFL constraint and provides a reason to prefer `CFL_VELOCITY_NORM=1` as the basis for the time step restriction.

Heat Transfer Constraint

Note that the heat transfer coefficient, h , has units of $\text{W}/(\text{m}^2 \text{ K})$. Thus, a velocity scale may be formed from $h/(\rho c_p)$. 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 $\delta t \leq \delta x \rho c_p / h$. This check may be skipped by setting `CHECK_HT=.FALSE.` on `MISC`.

Gravitational Constraint

A time scale restriction based on gravitational acceleration can be formed from $\delta t \leq \sqrt{\delta z / g}$, where δz is the grid size in the direction of gravity. This check may be omitted by setting `CHECK_GR=.FALSE.` on `MISC`.

6.4.10 Adjusting the Time Step

In the call to `CHECK_STABILITY` both the CFL and VN numbers are compared with `CFL_MAX [1.0]`, `CFL_MIN [0.8]`, `VN_MAX [0.5]`, and `VN_MIN [0.4]`, respectively (default values shown in brackets). To be clear, the CFL constraint is now given by

$$\text{CFL} = \delta t \left(\frac{\|\mathbf{u}\|}{\delta x} + |\nabla \cdot \mathbf{u}| \right) \quad (6.14)$$

If either the current CFL (augmented for both heat transfer and gravity) or VN is too large then the new time step is set to 90% of the allowable value. If both CFL and VN are below their minimum values then the current time step is increased by 10%.

Resetting the stability parameters is not recommended except in very special circumstances, as they can lead to simulations failing due to numerical instabilities.

If you want to prevent FDS from automatically changing the time step, set `LOCK_TIME_STEP=.TRUE.` 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.

6.5 Special Topic: Unusual Initial Conditions: The `INIT` Namelist Group (Table 16.10)

Usually, 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 some rectangular region of the domain. If so, add lines of the form

```
&INIT XB=0.5,0.8,2.1,3.4,2.5,3.6, TEMPERATURE=30. /
```

Here, within the region whose bounds are given by the sextuplet `XB`, the initial temperature shall be 30 °C instead of the ambient. This construct can also be used for `DENSITY` or pairs of `SPEC_ID(N)` and `MASS_FRACTION(N)` where `N` is an ordinal index starting from 1. Note that `N` is not necessarily indicative of the order in which the species are listed in the input file. Make sure that you specify all species (components of `MASS_FRACTION(N)`) on the same `INIT` line.

The `INIT` construct may be useful in examining the influence of stack effect in a building, where the temperature is different inside and out.

The `INIT` line may also be used to specify a volumetric heat source term. For example,

```
&INIT XB=0.5,0.8,2.1,3.4,2.5,3.6, HRRPUV=1000. /
```

indicates that the region bounded by `XB` shall generate 1000 kW/m³. This feature is mainly useful for diagnostics, or to model a fire in a very simple way.

Note that a solid obstruction can be given an initial temperature via the parameter `TMP_INNER` on the `SURF` line. An initial velocity can be prescribed via `U0`, `V0`, and `W0` on the `MISC` line.

6.6 Special Topic: The Pressure Solver: The `PRES` Namelist Group (Table 16.18)

FDS uses a low-Mach number formulation of the Navier-Stokes equations. One of the consequences of this is that the speed of sound is assumed infinite, and that the pressure throughout the computational domain is affected, instantaneously, by local changes in the flow field. A simple example of this is when air is pushed through a tunnel. If the tunnel has forced flow at one end and an opening at the other, the volume flow at the opening is the same as that which is forced at the other end. Without any heat addition, the air is assumed incompressible. Information is passed through the tunnel instantaneously in the model via a solution of a linear system of equations for the pressure. For a single mesh, the solution of this Poisson equation for the pressure is very accurate. However, for multiple meshes, there is potentially a delay in information passing throughout the domain because the Poisson equation is solved on each individual mesh, without any influence from the larger computational domain. The details of the numerics can be found in the FDS Technical Reference Guide.

Another limitation of the pressure solver is that at solid surfaces that are not part of the boundary of the computational domain, the pressure solver enforces a no-flux boundary condition. However, it is not perfect, and it is possible to have a non-zero normal velocity at a solid surface. For most applications, this velocity is so small that it has a negligible effect on the solution.

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. A good value to start with is $\delta x/10$, 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 per time step is given by `MAX_PRESSURE_ITERATIONS`, also on the `PRES` line. Its default value is 10000.

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.

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 case called **duct_flow**, 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 6.6 displays the computed inflow and outflow as a function of time, and the number of pressure iterations required. The outflow does not match the inflow exactly because of inaccuracies at the solid and mesh boundaries. The `VELOCITY_TOLERANCE` has been set to 0.01 m/s and the grid cell size is 0.1 m.

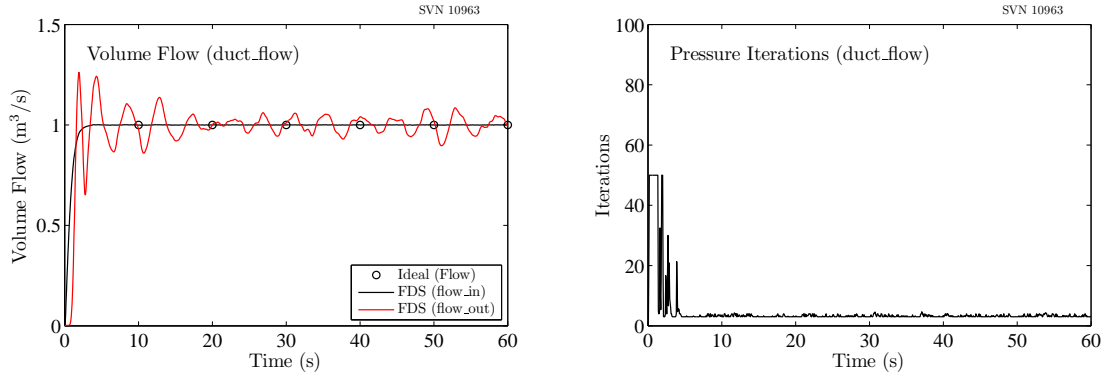


Figure 6.6: (Left) Volume flow into and out of a square duct. (Right) The number of pressure iterations as a function of time.

Example Case: Pressure_Solver/dancing_eddies

In this example, air is pushed through a 300 m long, two-dimensional channel at 5 m/s. A plate obstruction normal to the flow creates a Karman vortex street. The computational domain is divided into 4 meshes, each 75 m long. Two simulations are performed, one in which the `VELOCITY_TOLERANCE` is set to 0.1 m/s (`dancing_eddies_tol=pl.fds`), and one in which it is set to 0.01 m/s (`dancing_eddies.fds`). Figure 6.7 shows the downstream pressure histories for these two 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.

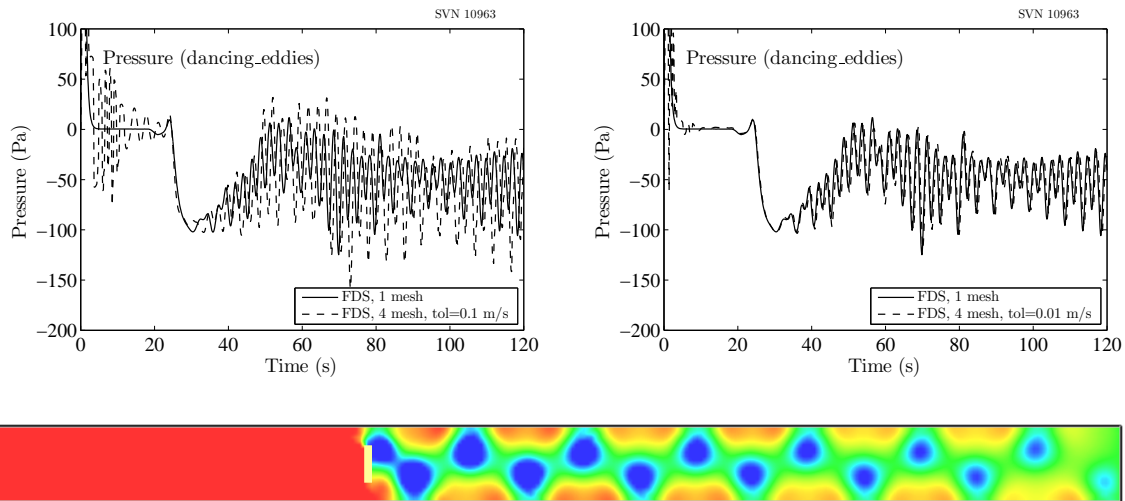


Figure 6.7: (Top) Comparison of pressure traces 200 m from the left opening of the channel for two different settings of `VELOCITY_TOLERANCE`. (Bottom) A contour plot of the pressure after 120 s for the case with the tolerance set to 0.01 m/s.

6.7 Special Topic: Setting Limits: The CLIP Namelist Group (Table 16.2)

The algorithms in FDS are designed to work within a certain range of values for the thermodynamic quantities 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 user 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, temperature, or species mass fractions. Temperature and density bounds are input under the namelist group called `CLIP`. The parameters are listed in Table 16.2. 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.

To force the species mass fractions to remain between 0 and 1, set `CLIP_MASS_FRACTION=.TRUE.` on the `MISC` line.

Chapter 7

Building the Model

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, *etc.*; 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 Bounding Surfaces: The SURF Namelist Group (Table 16.27)

Before describing how to build up the geometry, it is first necessary to explain how to describe what these bounding surfaces consist of. SURF is the namelist group that 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 temperature fixed at `TMPA`, and is referred to as 'INERT'. 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=.TRUE.` on the SURF line.

7.2 Creating Obstructions: The OBST Namelist Group (Table 16.16)

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 sextuplet

`XB = X1, X2, Y1, Y2, Z1, Z2`

In addition to the coordinates, the boundary conditions for the obstruction can be specified with the parameter SURF_ID, which designates which SURF group (Section 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(S)` 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.

Some additional features of obstructions are as follows:

- 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 can have zero thickness. Often, thin sheets, like a window, form a barrier, but if the numerical mesh is coarse relative to the thickness of the barrier, the obstruction might be unnecessarily large if it is assumed to be one layer of mesh cells thick. All faces of an obstruction are shifted to the closest mesh cell. If the obstruction is very thin, the two faces may be approximated on the same cell face. FDS and Smokeview render this obstruction as a thin sheet, but it is allowed to have thermally thick boundary conditions. This feature is fragile, especially in terms of burning and blowing gas. A thin sheet obstruction can only have one velocity vector on its face, thus a gas cannot be injected reliably from a thin obstruction because whatever is pushed from one side is necessarily pulled from the other. For full functionality, the obstruction should be specified to be at least one mesh cell thick. Thin sheet obstructions work fine as flow barriers, but other features are fragile and should be used with caution. To prevent FDS from allowing thin sheet obstructions, set `THICKEN_OBSTRUCTIONS=.TRUE.` on the `MISC` line, or `THICKEN=.TRUE.` on each `OBST` line for which the thin sheet assumption is not allowed.
- Obstructions that are too small relative to the underlying numerical mesh are rejected. Be careful when testing cases on coarse meshes.
- Obstructions may be created or removed during a simulation. See Section [15.4.1](#) for details.
- If two obstructions overlap at one or more faces, the one listed last in the input file takes precedence over the one listed first, in the sense that the latter's surface properties will be applied to the overlapping face. 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 first obstruction 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.
- Obstructions can be protected from the `HOLE` punching feature. Sometimes it is convenient to create a door or window using a `HOLE`. 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=.FALSE.` is included on the `OBST`

line. In general, any OBSTRUCTION can be made impenetrable to a HOLE using this phrase. By default, PERMIT_HOLE=.TRUE., meaning that an OBSTRUCTION is assumed to be penetrable unless otherwise directed. Note that if an penetrable OBSTRUCTION and an impenetrable OBSTRUCTION overlap, the OBSTRUCTION with PERMIT_HOLE=.FALSE. should be listed first.

- If the obstruction is not to be removed or rejected for any reason, set REMOVABLE=.FALSE. 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=.FALSE.
- 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 drawn solid in Smokeview. To draw an outline representation, set OUTLINE=.TRUE.

7.2.2 Repeated Obstructions: The MULT Namelist Group (Table 16.15)

Sometimes obstructions 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 obstruction or set of obstructions repeats itself in a regular pattern, you can use a utility known as a multiplier. If you want to repeat an obstruction, 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=..., MULT_ID='m1' /
```

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

$$\begin{aligned} x_i &= x_0 + \delta x_0 + i \delta x & ; & \quad I_LOWER \leq i \leq I_UPPER \\ y_j &= y_0 + \delta y_0 + j \delta y & ; & \quad J_LOWER \leq j \leq J_UPPER \end{aligned}$$

Note that the same rules apply for the z direction as well. In situations where the position of the obstruction needs shifting prior to the multiplication, use the parameters DX0, DY0, and DZ0.

A generalization 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, define the lower y bound of the original obstruction by XB(3, 0). The n th obstruction would be:

$$XB(3, N) = XB(3, 0) + N * DXB(3)$$

Notice that we use N_LOWER and N_UPPER 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, SVN $Revision: 11024 $' /
&MESH IJK=100,100,100, XB=0.0,1.0,0.0,1.0,0.0,1.0 /
&TIME T_END=0. /
```

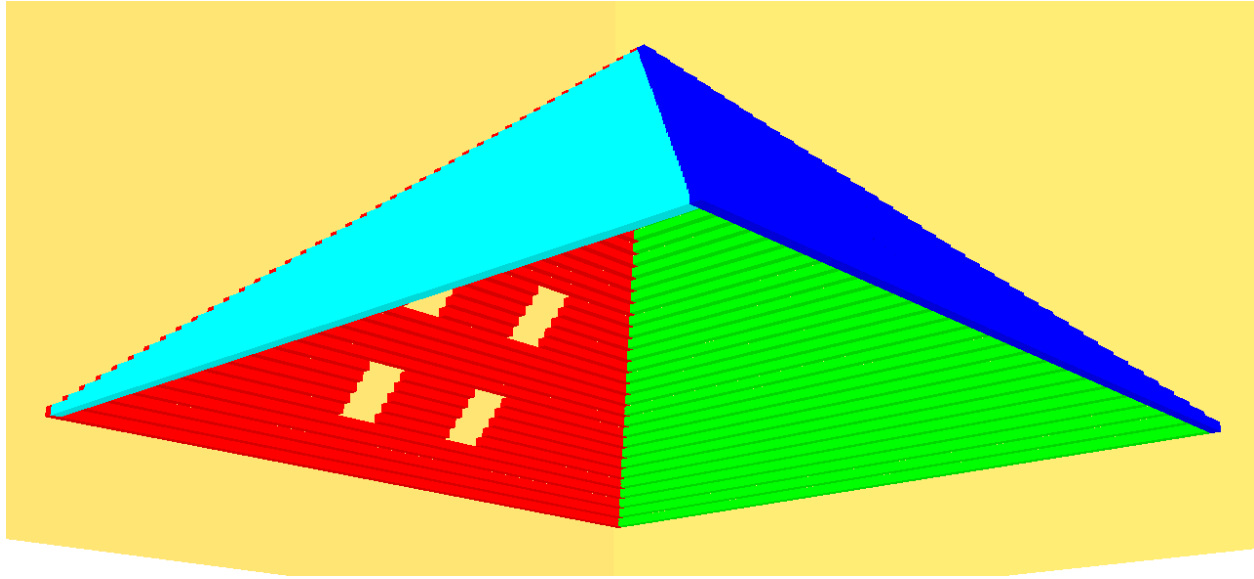


Figure 7.1: An example of the multiplier function.

```
&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.

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

7.2.3 Non-rectangular Geometry and Sloped Ceilings

The efficiency of FDS is due to the simplicity of its numerical mesh. However, there are situations in which certain geometric features do not conform to the rectangular mesh, such as a sloped ceiling or roof. In these cases, construct the curved geometry using rectangular obstructions, a process sometimes called “stair-stepping”. A concern is that the stair-stepping changes the flow pattern near the wall. To lessen the impact of stair-stepping on the flow field near the wall, prescribe the parameter `SAWTOOTH=.FALSE.` on each `OBST` line that makes up the stair-stepped obstruction. The effect of this parameter is to prevent vorticity from being generated at sharp corners, in effect smoothing out the jagged steps that make up the obstruction. This is not a complete solution of the problem, but it does provide a simple way of ensuring that the flow field around a non-rectangular obstruction is not inhibited by extra drag created at sharp corners.

Do not apply `SAWTOOTH=.FALSE.` to obstructions that have any `SURF_IDS` with the attribute `BURN_AWAY=.TRUE.`

Example Case: sawtooth

In this example, we look at the flow field past a diagonally oriented obstruction. If `SAWTOOTH=.FALSE.`, then the velocity boundary conditions will be applied in such a way as to minimize the impact of the boundaries due to vortices at sharp corners, as shown in the following example:

```
&OBST XB= 0.00, 0.05,-0.01, 0.01, 0.00, 0.05, SAWTOOTH=.FALSE., COLOR='EMERALD GREEN' /  
&OBST XB= 0.05, 0.10,-0.01, 0.01, 0.00, 0.10, SAWTOOTH=.FALSE., COLOR='EMERALD GREEN' /  
&OBST XB= 0.10, 0.15,-0.01, 0.01, 0.05, 0.15, SAWTOOTH=.FALSE., COLOR='EMERALD GREEN' /  
&OBST XB= 0.15, 0.20,-0.01, 0.01, 0.10, 0.20, SAWTOOTH=.FALSE., COLOR='EMERALD GREEN' /
```

In Figure 7.2, the top set of obstructions are using the default `SAWTOOTH=.TRUE.` and the bottom set of obstructions are using `SAWTOOTH=.FALSE.` The adjacent obstructions that have `SAWTOOTH=.FALSE.` are displayed in Smokeview as one smooth obstruction, shown in green. Notice that as the air moves across the different sets of obstructions, the air velocity on the bottom set of obstructions is not affected as much by the vortices.

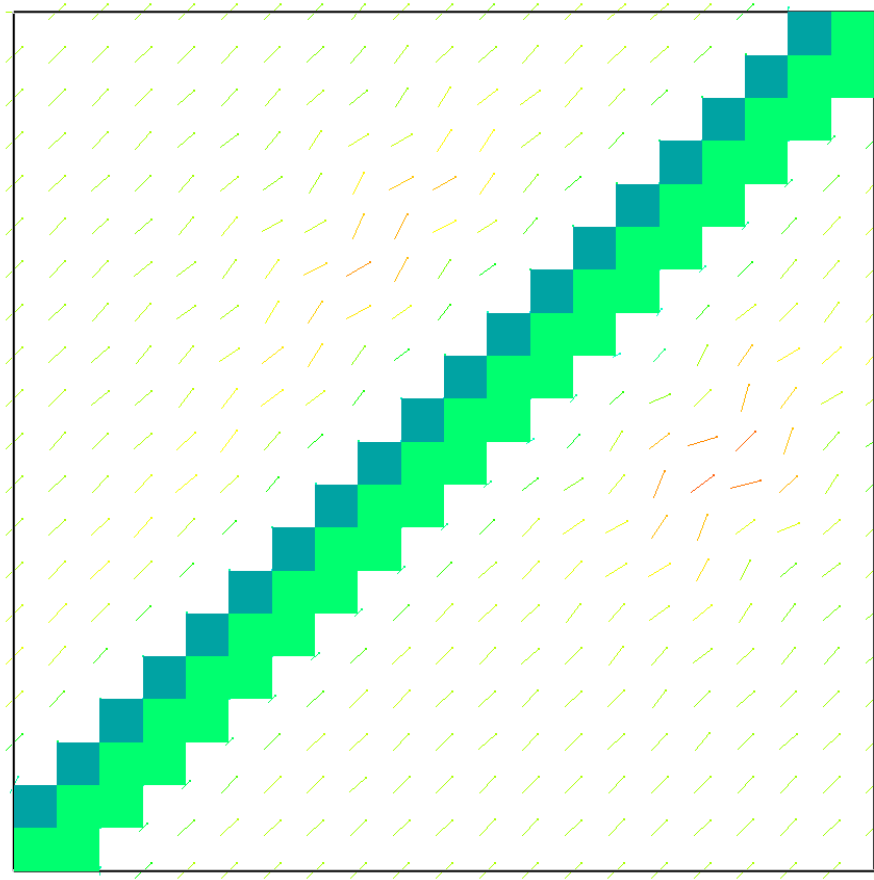


Figure 7.2: Simple example of `SAWTOOTH=.FALSE.`

7.3 Creating Voids: The HOLE Namelist Group (Table 16.8)

The HOLE namelist group is used to define parameters (Table 16.8) to 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 Section 15.4.1 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 line.

If you want the obstruction that is to be cut out to have a different color than the original obstruction, set the COLOR or integer triplet RGB on the HOLE line (see Section 7.5). If you want the obstruction to be invisible, set COLOR='INVISIBLE'.

When a HOLE is in a .FALSE. state, an obstruction is placed in the hole. To make this obstruction transparent, the TRANSPARENCY parameter should be specified by a real number from 0 to 1. Note that if TRANSPARENCY is specified, then either a COLOR or RGB triplet ought to be specified as well. A TRANSPARENCY value near, but not equal to, zero can be used to simulate a window when the HOLE's INITIAL_STATE=.FALSE.. When the DEVC or CTRL is activated and changes the state of the hole to .TRUE., the HOLE is then open and completely transparent. See Section 15.4.1 for an example.

If an obstruction is not to be punctured by a HOLE, add PERMIT_HOLE=.FALSE. to the OBST line.

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.

Note that a HOLE has no effect on a VENT or a mesh boundary. It only applies to OBSTstructions.

7.4 Applying Surface Properties: The VENT Namelist Group (Table 16.31)

Whereas the OBST group is used to specify obstructions within the computational domain, the VENT group (Table 16.31) 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.

7.4.1 Basics

The vents are chosen in a similar manner to the 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 wall cell. If additional VENT lines are specified for a given wall cell, FDS will output a warning message and ignore the subsequent lines (i.e. only the first vent will be applied)

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. See Section 9.1 for instructions on specifying different types of fans that allow gases to flow through.

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 keyword MB is applied to all of the meshes, 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 prescribed. Use color as much as possible to double-check the set-up. More detail on color in Section 7.5 and Table 7.1. Also, the parameter OUTLINE=.TRUE. causes the VENT to be drawn as an outline in Smokeview.

7.4.2 Special VENTS

There are three reserved `SURF_ID`'s that may be applied to a `VENT` – '`OPEN`', '`MIRROR`', and '`PERIODIC`'. 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 change this, use an `OPEN` vent as if it were an open door or window. To create a totally or partially open domain, use `OPEN` vents on the exterior mesh boundaries (`MBs`).

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`. 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 Section 9.3.3 for more details. If you want to specify a non-ambient pressure at the `OPEN` boundary, see Section 9.4.

The user should also be aware that 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 $\mathcal{H} = k = 50 \text{ m}^2/\text{s}^2$. But when the vertical velocity is negative (-0.001 m/s) then $\mathcal{H} = 0$ (see [15]). For this reason, `OPEN` vents are not recommended for the top or side boundaries in an outdoor flow. See Section 6.4.2 for a better alternative.

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 Section 15.4.2 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 reversed. From a numerical point of 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.

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

A word of warning about `MIRROR` boundaries in FDS. 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 boundary of the domain in any of the three coordinate directions. As an example, consider the following:

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

In this example, the entire *xmin* boundary is periodic with the *xmax* boundary.

Note that periodic vents may not be used to connect offset vents or vents in different coordinate directions. For such cases, the user must employ HVAC capabilities (see Section 9.2).

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 Section 15.4.2 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.

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.

7.4.5 Special Topic: 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). Refer to Jarrin [16] 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. 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. If the fluctuations are not isotropic, then the Reynolds stresses must be specified componentwise. For example,

```
&SURF ID='inlet', VEL=-10. /
&VENT MB='XMIN', SURF_ID='inlet', N_EDDY=100, L_EDDY=0.5, VEL_RMS=1. /
```

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.

In Chapter 7 of Jarrin’s thesis [16], he introduces the Modified Synthetic Eddy Method in which the eddy length scales are anisotropic. This allows more realistic characterization of streamwise 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)`. For example,

```
&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
      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 /
```

7.5 Coloring Obstructions, Vents, Surfaces and Meshes

Colors for many items within FDS can be prescribed in two ways; a triplet of integers after keyword RGB or one of many COLOR name character strings.

The three RGB integer numbers 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.

Table 7.1 provides a small sampling of RGB values and COLOR names for a variety of colors. A complete listing of all 500+ colors that can be specified by name after the COLOR keyword is available on the FDS website. If the COLOR name is not listed in the table on the website, then that name does not exist to FDS.

It is highly recommended that colors be assigned to surfaces via the SURF line because as the geometries of FDS simulations become more complex, it is very useful to use color as a spot check to determine if the desired surface properties have been assigned throughout the room or building under study.

For example, if you desire that all surfaces associated with a given SURF line be colored the same way, prescribe a triplet of integers called RGB on the SURF line. The following SURF line

```
&SURF ID='UPHOLSTERY', ..., RGB=0,255,0 /
```

will cause the furnishings with a SURF_ID of 'UPHOLSTERY' to be colored green in Smokeview. It is best to avoid using the primary colors because these same colors are used by Smokeview to draw color contours.

Obstructions and vents may be colored individually (over-riding the SURF line's RGB) by specifying COLOR value to any of the listed names in Table 7.1 or 'INVISIBLE' on the respective OBST or VENT line. Using 'INVISIBLE' causes the vent or obstruction to not be drawn.

Colors may also be specified using the integer triplet RGB on an OBST or VENT line to gain a wider color palette. See Table 7.1 for a list of color names and RGB values.

7.5.1 Texture Mapping

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 text





























































```
&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 should read it and display the image wherever the paneling is used. 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,

```
&OBST XB=1.0,2.0,3.0,4.0,5.0,7.0, SURF_ID='wood paneling', TEXTURE_ORIGIN=1.0,3.0,5.0 /
```

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 (1.0,3.0,5.0). 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.

Table 7.1: Sample of Color Definitions (A complete list is included on the website)

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
DIMGRAY		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

Chapter 8

Fire and Thermal Boundary Conditions

This chapter describes how to specify the thermal properties of solid objects. **This is the most challenging part of setting up the simulation.** Why? First, for both real and simulated fires, the growth of the fire is very sensitive to the thermal properties of the surrounding materials. Second, even if all the material properties are known to some degree, the physical phenomena of interest may not be simulated properly due to limitations in the model algorithms or resolution of the numerical mesh. It is your responsibility to supply the thermal properties of the materials, and then assess the performance of the model to ensure that the phenomena of interest are being captured.

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` (Section 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 (Section 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.

The thickness of the wall indicated by the `OBST` line need not match that indicated by the `SURF` line. The thickness of the material on the surface of the wall is dictated by the parameter `THICKNESS`. These two parameters are independent for each other, the `OBST` line describes the overall geometric structure, the `SURF` line describes the characteristics of the surfaces of the geometry which includes the thickness of the layers of materials applied to that surface.

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

Usually, the thermal properties of a solid boundary are specified via the `MATL` namelist group, which is in turn invoked by the `SURF` entry via the character string `MATL_ID`. However, sometimes it is convenient to specify a fixed temperature boundary condition, in which case set `TMP_FRONT` to be the surface temperature in units of °C:

```
&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 Special Topic: Convective Heat Transfer Options

This section is labeled as a special topic because normally you do not need to modify the convective heat transfer model in FDS. 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

By default in an LES calculation, the convective heat flux to the surface is obtained from a combination of natural and forced convection correlations

$$\dot{q}_c'' = h \Delta T \quad \text{W/m}^2 \quad ; \quad h = \max \left[C |\Delta T|^{\frac{1}{3}}, \frac{k}{L} \text{Nu} \right] \quad \text{W/m}^2/\text{K} \quad (8.1)$$

where ΔT is the difference between the wall and the gas temperature, C is the coefficient for natural convection (1.52 for a horizontal surface and 1.31 for a vertical surface, by default). The Nusselt number, Nu , depends on the shape of the obstruction [17]:

$$\text{Nu} = \begin{cases} 0.037 \text{Re}^{\frac{4}{3}} \text{Pr}^{\frac{1}{3}} & \text{Cartesian} \\ 0.664 \text{Re}^{\frac{1}{2}} \text{Pr}^{\frac{1}{3}} & \text{Cylindrical} \\ 2 + 0.6 \text{Re}^{\frac{1}{2}} \text{Pr}^{\frac{1}{3}} & \text{Spherical} \end{cases} \quad (8.2)$$

L is a characteristic length related to the size of the physical obstruction; k is the thermal conductivity of the gas, and the Reynolds Re and Prandtl Pr numbers are based on the gas flowing past the obstruction. Since the Reynolds number is proportional to the characteristic length, L , the heat transfer coefficient is weakly related to L (for high Re , $h \sim L^{-1/5}$). For this reason, L is taken to be 1 m for most calculations. You can change the empirical coefficients using `C_HORIZONTAL` or `C_VERTICAL` for C and `C_FORCED`, `C_FORCED_CYLINDER`, or `C_FORCED_SPHERE` for the constant in the Nusselt number correlation, all of which are input on the `MISC` line. The length scale can be changed for a specific surface by using `CONVECTION_LENGTH_SCALE` on the `SURF` line.

Changing the Convective Heat Transfer Coefficient

If you want to change the default convective heat transfer coefficient, you can set it to a constant using `H_FIXED` on the `SURF` line in units of $\text{W/m}^2/\text{K}$.

Specifying the Heat Flux at a Solid Surface

Instead of altering the convective heat transfer coefficient, you may specify a fixed heat flux 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 separately the `CONVECTIVE_HEAT_FLUX`, in units of kW/m^2 , and the radiative heat flux. The radiative heat flux is specified by setting both `TMP_FRONT` and `EMISSIVITY` appropriately on the `SURF` line. Note that if you wish there to be only a convective heat flux from a surface, then the `EMISSIVITY` should be set to zero. 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.

8.2.3 Special Topic: Adiabatic Surfaces

For some special applications, it is often desired that a solid surface be adiabatic, that is, there is no net heat transfer (radiative and convective) from the gas to the solid. For this case, all that must be prescribed on the `SURF` line is `ADIABATIC=.TRUE.`, and nothing else. FDS will compute a wall temperature so that the sum of the net convective and radiative heat flux is zero. Specifying a surface as `ADIABATIC` will result in FDS defining `NET_HEAT_FLUX=0`.

No solid surface is truly adiabatic; thus, the specification of an adiabatic boundary condition should be used for diagnostic purposes only.

8.3 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. The `MATL` namelist group is used to define the properties of the materials that make up boundary solid surfaces. 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 `SURF` is applied to the face of an `OBST`, the first `MATL_ID` is at the face of the `OBST`, with the other `MATL_ID`s being applied in succession with the final `MATL_ID` being applied on the opposite face of the `OBST`. 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 %.

It is important to notice that the components of the solid mixtures are treated as pure substances with no voids. The density of the mixture is

$$\rho = \left(\sum_i \frac{Y_i}{\rho_i} \right)^{-1} \quad (8.3)$$

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/m³. 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/m³ of brick left. To have a density of 1600 kg/m³ 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` (W/m·K), `DENSITY` (kg/m³), `SPECIFIC_HEAT` (kJ/kg/K), and `EMISSIVITY` (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.

8.3.3 Back Side Boundary Conditions

The layers of a solid boundary are listed in order from the surface. By default, this innermost layer is assumed to back up to an air gap at ambient temperature. This is true even if the obstruction forms a wall in the model that backs up to another compartment. A good example of the default back side boundary condition is a sheet of gypsum board attached to wood studs. It is assumed that the back side of the gypsum board is an ambient temperature void space within the wall. It does not matter if the obstruction on which the boundary condition is applied is thick or thin.

There are other back side boundary conditions that can be applied. One is to assume that the wall backs up to an insulated material in which case no heat is lost to the backing material. The expression `BACKING='INSULATED'` on the `SURF` line prevents any heat loss from the back side of the material. Use of this condition means that you do not have to specify properties of the inner insulating material because it is assumed to be perfectly insulated.

If the wall is assumed to back up to the room on the other side of the wall and you want FDS to calculate the heat transfer through the wall into the space behind the wall, the attribute `BACKING='EXPOSED'` should be listed on the `SURF` line. This feature only works if the wall is less than or equal to one mesh cell thick, and if there is a non-zero volume of computational domain on the other side of the wall. Obviously, if the wall is an external boundary of the domain, the heat is lost to an ambient temperature void. The same happens if the back side gas cell cannot be found.

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 and Back Side 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. The layers of the surface can have different initial temperatures. Also, the back side temperature boundary condition of a solid can be set using the parameter `TMP_BACK` on the `SURF` line. `TMP_BACK` is not the actual back side surface temperature, but rather the gas temperature to which the back side surface is exposed. This parameter has no meaning for surfaces with `BACKING='EXPOSED'` or `BACKING='INSULATED'`.

Note that the parameters `TMP_INNER` and `TMP_BACK` are only meaningful for solids with specified `THICKNESS` and material properties (via the `MATL_ID` keyword).

8.3.5 Walls with Different Materials Front and Back

If you apply the attribute `BACKING='EXPOSED'` on a `SURF` line that is applied to a zero or one-cell thick obstruction, FDS calculates the heat conduction through the entire `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, so be careful how you specify multiple layers. If the layering is symmetric, the same `SURF` line can be applied to both sides. However, if the layering is not symmetric, you must create two separate `SURF` lines and apply 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'
  COLOR           = 'ANTIQUE WHITE'
  BACKING         = 'EXPOSED'
```

```

MATL_ID(1:3,1)      = 'PLASTIC','INSULATION','STEEL'
THICKNESS(1:3)      = 0.002,0.036,0.0063 /

&SURF ID            = 'COLUMN INTERIOR'
COLOR               = 'BLACK'
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 you need to indicate which side of the column would generate the fuel vapor. In this case, the steel is impermeable; thus you should add the parameter `LAYER_DIVIDE=2.0` to the SURF line labeled 'COLUMN EXTERIOR' to indicate that fuel vapors formed by the heating of the two first layers ('PLASTIC' and 'INSULATION') are to be driven out of that surface. You need to also specify `LAYER_DIVIDE=0.0` on the SURF line labeled 'COLUMN INTERIOR' to indicate that no fuel vapors are to be 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.

8.3.6 Special Topic: 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. Use `GEOMETRY='SPHERICAL'` to describe a spherical object.

8.3.7 Special Topic: Solid Phase Numerical Gridding Issues

To compute the temperature and reactions inside the solids, FDS solves the one-dimensional heat transfer equation numerically. The size of the mesh cells on the surface of the solid is automatically chosen using a rule that makes the cell size smaller than the square root of the material diffusivity ($k/\rho c$). By default, the solid mesh cells increase towards the middle of the material layer and are smallest on the layer boundaries.

The default parameters are usually appropriate for simple heat transfer calculations but sometimes the use of pyrolysis reactions makes the temperatures and burning rate fluctuate. Adjustments may also be needed in case of extremely transient heat transfer situations. The numerical accuracy and stability of the solid phase solution may be improved by one of the following methods:

Make the mesh density more uniform inside the material by setting `STRETCH_FACTOR(NL)=1.` on the `SURF` line. This will generate a perfectly uniform mesh for layer number `NL`. (This happens automatically if the layer contains one or more reacting materials.) Values between 1 and 2 give different levels of stretching. Note that `STRETCH_FACTOR` needs to be specified for all the layers.

Make the mesh cells smaller by setting `CELL_SIZE_FACTOR` less than 1.0. For example, `CELL_SIZE_FACTOR=0.5` makes the mesh cells half the size. The scaling always applies to all layers.

Improve the time resolutions by setting `WALL_INCREMENT=1` on the `TIME` line. This forces the solid phase temperatures to be solved on every time step.

Enforce the re-gridding of shrinking wall cells by setting `REGRID_FACTOR` on the `SURF` line to a value that is larger than 0.9 (default) and smaller than 1.0. For example, `REGRID_FACTOR=0.99` will cause re-gridding when the any of the cells has changed one percent in size, thus reducing the amount of interpolation errors. Setting `REGRID_FACTOR` less than 0.9 can save computing time in some cases.

If all the material components of the surface are reacting, and the pyrolysis reactions have no solid residue, the thickness of the surface is going to shrink when the surface reacts. 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 via `MINIMUM_LAYER_THICKNESS` keyword, defaulting to 1.E-6 m. When all the material of a shrinking surface is consumed but `BURN_AWAY` is not prescribed, the surface temperature is set to `TMP_BACK`, convective heat flux to zero and burning rate to zero.

See Section 8.5 for ways to check and improve the accuracy of the solid phase calculation.

8.4 Pyrolysis Models

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 the user to explicitly define the gas phase reaction. See Chapter 12 for details.

8.4.1 A Gas Burner with a Specified Heat Release Rate

Solids and liquid fuels can be modeled by specifying their relevant properties via the `MATL` namelist group. However, if you simply want to specify a fire of a given heat release rate (HRR), you need not specify any material properties. A specified fire is basically 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 kW/m^2 to any surface with the attribute `SURF_ID='FIRE'`. See the discussion of **Time Dependent Conditions** in Chapter 10 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 $\text{kg/m}^2/\text{s}$. Do not specify both `HRRPUA` and `MLRPUA` on the same `SURF` line. With either, the stoichiometry of the gas phase reaction is set by the simple chemistry input parameters on the `REAC` line. If an Arrhenius-rate combustion model is desired instead of the default mixing controlled model, see Section 12.3.

Specifying `HRRPUA` or `MLRPUA` requires the use of the simple chemistry inputs for `REAC`.

8.4.2 Special Topic: A Radially-Spreading Fire

Sometimes it is desired that a fire spread radially at some specified rate. Rather than trying to design material properties to achieve this, you can alternatively use a `VENT` or a `SURF` line in a special way. First, you need to add a `SURF` line with a specified heat release rate, `HRRPUA`, and an optional time history parameter, `RAMP_Q` or `TAU_Q`. Then, you can also specify `XYZ` and `SPREAD_RATE` on either the `VENT` or `SURF` line. The fire is directed to start at the point `XYZ` 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 patch at $z = 0$ on which the fire starts at the point (1.5,4.0,0.0) and spreads outwards 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 in this construct 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 the “ t -squared” curve – 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.

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.

8.4.3 Solid Fuels 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 net 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 /
```

The parameters `HRRPUA`, `IGNITION_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. It should have no reaction parameters, product yields, and so on, like those described in the previous sections. By specifying `HRRPUA`, you are controlling the burning rate rather than letting the material pyrolyze based on the conditions of the surrounding environment.

8.4.4 Solid Fuels that do NOT Burn at a Specified Rate

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 Polymers,” by Hirschler and Morgan, or “Flaming Ignition of Solid Fuels,” by Torero, both of which are in the 4th edition of the *SFPE Handbook of Fire Protection Engineering*.

The 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`. It is very important that you designate `N_REACTIONS` 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 `MATL`, the j th reaction can produce other solid materials whose names are designated with `MATL_ID(i, j)`, and gas species whose names are designated with `SPEC_ID(i, j)`. Note that the index, i , runs from 1 to the number of material or gaseous species. 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 or gaseous products are input to FDS via the *yields*: `NU_MATL(i, j)` and `NU_SPEC(i, j)`, respectively. The yields are all zero by default. If `NU_MATL(i, j)` or `NU_SPEC(i, j)` is non-zero, then you *must* indicate what the solid residue is via `MATL_ID(j)`, the ID of another `MATL` that is also listed in the input file. Ideally, the sum of the yields should add to 1, meaning that the mass of the reactant is conserved. However, there are times when it is convenient to have the yields sum to something 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.

For consistency, the `HEAT_OF_COMBUSTION(j)` can also be specified for each reaction, j . These values are used only if the corresponding heats of combustion for the gaseous species are greater than zero.

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)                    = 'PYROLYZATE','OXYGEN','WATER VAPOR','CARBON DIOXIDE'
    NU_SPEC(1:4,1)                    = 0.82,0,0,0
    HEAT_OF_REACTION(1)                = 430.
    HEAT_OF_COMBUSTION(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.

The Reaction Rates

For each reaction that each material component undergoes you must specify kinetic parameters of the reaction rate. The general evolution equation for a material undergoing one or more reactions is:

$$\frac{\partial Y_{s,i}}{\partial t} = - \sum_{j=1}^{N_{ri}} r_{ij} + \sum_{i'=1}^{N_m} \sum_{j=1}^{N_{ri'}} v_{s,i'j} r_{i'j} \quad (i' \neq i) \quad ; \quad r_{ij} = A_{ij} Y_{s,i}^{n_{s,ij}} \exp\left(-\frac{E_{ij}}{RT_s}\right) \quad ; \quad Y_{s,i} = \left(\frac{\rho_{s,i}}{\rho_{s0}}\right) \quad (8.4)$$

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 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. ρ_{s0} is the initial density of the layer. Thus, $Y_{s,i} = \rho_{s,i}/\rho_{s0}$ 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. If the layer is composed of only one material, then $\rho_{s,i}/\rho_{s0}$ is initially 1. $n_{s,ij}$ 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 $n_s = 1$.

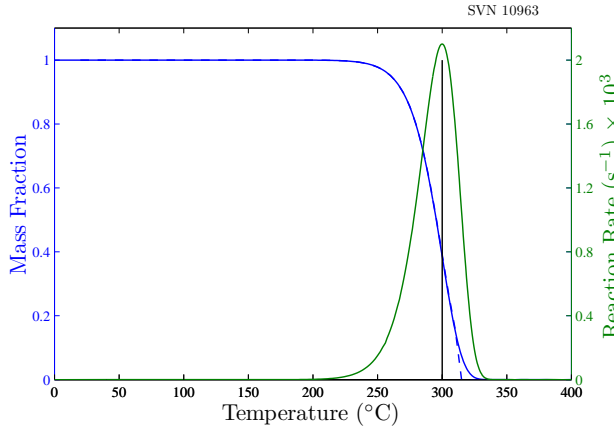
The pre-exponential factor, A_{ij} , is prescribed under the name `A(j)` on the `MATL` line of the i th material, with units of s^{-1} . E_{ij} , the activation energy, is prescribed via `E(j)` in units of kJ/kmol. Remember that 1 kcal is 4.184 kJ, and be careful with factors of 1000. For a given reaction, 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 (Thermo-Gravimetric Analysis).

It is very important to keep in mind that A and E are not available for most real materials. If A and E are not known, there are several parameters that can be used by FDS to derive effective values. The most important parameter to specify in place of A and E is the `REFERENCE_TEMPERATURE` ($^{\circ}\text{C}$). To understand this parameter, consider the plot shown in Fig. 8.1. These curves represent the results of a hypothetical TGA experiment. The Mass Fraction (blue curve) is the normalized density of the material (Y) which decreases as the sample is slowly heated, in this case at a rate of 5 K/min. The Reaction Rate (green curve) is the

rate of change of the mass fraction as a function of time ($-dY/dt$). Where this curve peaks is referred to in FDS as the `REFERENCE_TEMPERATURE`.² Note that the `REFERENCE_TEMPERATURE` is *not* the same as 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 the reaction are found from the formulae³:

$$E = \frac{e r_p}{Y_0} \frac{R T_p^2}{\dot{T}} \quad ; \quad A = \frac{e r_p}{Y_0} e^{E/RT_p} \quad (8.5)$$

where T_p and r_p/Y_0 are the reference temperature and rate, respectively. The `REFERENCE_RATE` is the reaction rate, in units of s^{-1} , at the given `REFERENCE_TEMPERATURE` divided by the mass fraction, Y_0 , of material in the original sample undergoing the reaction. For a single component, single reaction material, $Y_0 = 1$. 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. 8.1, the area under the green curve (Reaction Rate) is equal to the heating rate (in units of K/s).



$$\frac{dY}{dt} = -AY \exp(-E/RT) \quad Y(0) = 1$$

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

Figure 8.1: The blue curve represents the normalized mass, $Y = \rho_s/\rho_{s0}$, of a solid material undergoing heating at a rate of 5 K/min. The green curve represents the reaction rate, $-dY/dt$. The system of ordinary differential equations that describe 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 (8.5).

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 along with `REFERENCE_TEMPERATURE` (T_p) 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 green curve, assuming its shape to be roughly triangular. Its default value is 80 $^\circ\text{C}$. Using these input parameters, an estimate is made of the peak reaction rate, r_p , with

²The term “reference temperature” is used simply to maintain backward compatibility with earlier versions of FDS.

³These formulas have been derived from an analysis that considers a first-order reaction. When using the proposed method, do not specify non-unity value for the reaction order `N_S` on the `MATL` line.

which E , then A , are calculated.

$$\frac{r_p}{Y_0} = \frac{2\dot{T}}{\Delta T} (1 - v_s) \quad (8.6)$$

The parameter, v_r , is the yield of solid residue.

When in doubt about the values of these 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 `REFERENCE_TEMPERATURE`, and do not specify `PYROLYSIS_RANGE` if you specify `REFERENCE_RATE`. For the material decomposition shown in Fig. 8.1, the `MATL` would have the form:

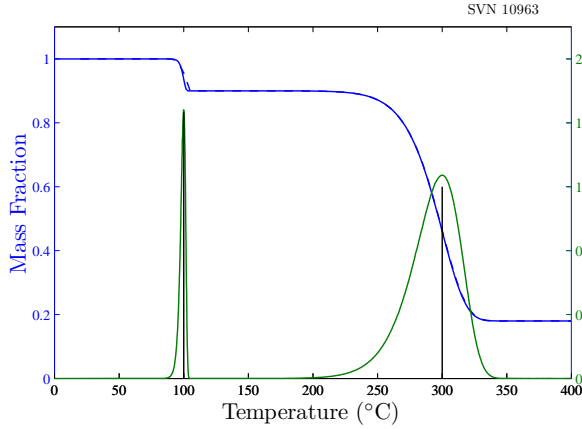
```
&MATL ID                      = 'My Fuel'
...
N_REACTIONS                   = 1
SPEC_ID (1,1)                 = '...'
NU_SPEC (1,1)                 = 1.
REFERENCE_TEMPERATURE (1)     = 300.
REFERENCE_RATE (1)            = 0.002
HEATING_RATE (1)              = 5.
HEAT_OF_COMBUSTION (1)        = ...
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, you need not include the `(1)`, but it is a good habit to get into. Note also that if the default combustion model is used, you can denote that the reaction produces fuel gas using the appropriate `SPEC_ID`. Note also that 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, so why is it specified here? The answer is that there can be only one *gas phase* reaction of fuel and oxygen in FDS, but there can be dozens of different materials and dozens of *solid phase* reactions. To ensure that the fuel vapors from different materials combust to produce the proper amount of energy, it is very important to specify a `HEAT_OF_COMBUSTION` for each material. That way, the mass loss rate of fuel gases is automatically adjusted so that the effective mass loss rate multiplied by the single, global, gas phase heat of combustion produces the expected heat release rate. If, for example, the `HEAT_OF_COMBUSTION` 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 reaction, j , via the parameter `HEAT_OF_COMBUSTION (j)`.

Multiple Solid Phase Reactions

The solid phase reaction represented by Fig. 8.1 is fairly simple – a single, homogenous material is heated and gasified completely. In general, real materials are not so simple. First, they consist of more than one material component, each of which can react over a particular temperature interval, and some of which leave behind a solid residue. Some material components may even undergo multiple reactions that form different residues, like woods that form various amounts of tar, char, and ash, depending on the rate of heating. Figure 8.2 demonstrates a more complicated material than the one previously described. It is a hypothetical material that contains 10 % (by mass) water and 90 % solid material. The water evaporates in the neighborhood of 100 °C and the solid pyrolyzes in the neighborhood of 300 °C, leaving 20 % of its mass behind in the form of a solid residue.

The key input lines for this reaction are shown in Fig. 8.3. Note that the only parameters shown are those that describe the reaction mechanism, and that each of these parameters can be found either from



$$\begin{aligned}
 \frac{dY_1}{dt} &= -A_{1,1} Y_1 \exp(-E_{1,1}/RT) & Y_1(0) &= 0.1 \\
 \frac{dY_2}{dt} &= -A_{2,1} Y_2 \exp(-E_{2,1}/RT) & Y_2(0) &= 0.9 \\
 \frac{dY_3}{dt} &= -v_{s,2,1} \frac{dY_2}{dt} & Y_3(0) &= 0.0 \\
 T_{p,1,1} &= 100 + 273 \text{ K} & T_{p,2,1} &= 300 + 273 \text{ K} \\
 r_{p,1,1} &= 0.0016 \text{ s}^{-1} & r_{p,2,1} &= 0.0012 \text{ s}^{-1} \\
 v_{s,1,1} &= 0 & v_{s,2,1} &= 0.2 \\
 \dot{T} &= 5 \text{ K/min}
 \end{aligned}$$

Figure 8.2: The blue curve represents the combined mass fraction, $\sum Y_i$, and the green curve the net reaction rate, $-d/dt(\sum Y_i)$, 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. In this case, there are three material components, and the first two each undergo a single reaction. The third material component is formed as a residue of the reaction of the second material. The system of ordinary differential equations that governs the transformation of the materials is shown at right.

visual inspection of the the mass loss (blue) curve or the reaction rate (green) curve. Even if TGA or similar data were unavailable in this case, you can still model the solid as a combination of water that evaporates at 100 °C and some other material that pyrolyzes in the vicinity of 300 °C, leaving 20 % of its mass as a residue. The full set of parameters for these cases are listed in **pyrolysis_1.fds** and **pyrolysis_2.fds**. Those interested in testing potential solid phase reaction mechanisms ought to use these test cases as templates.

The Heat of Reaction

Equation (8.4) 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 something else, 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.

Special Topic: The “Threshold” Temperature

In FDS, the reaction rate expression in Eq. (8.4) includes an optional term:

$$r_{ij} = A_{ij} Y_{s,i}^{n_{s,ij}} \exp\left(-\frac{E_{ij}}{RT_s}\right) \max[0, S_{thr,ij}(T_s - T_{thr,ij})]^{n_{t,ij}} \quad (8.7)$$

$T_{thr,ij}$ is an optional “threshold” temperature that allows the definition of non-Arrhenius pyrolysis functions and ignition criteria, and is prescribed by `THRESHOLD_TEMPERATURE(j)`. $S_{thr,ij}$ is the “threshold direction”

```

&SURF ID                                = 'SAMPLE'
...
MATL_ID(1,1:2)                          = 'stuff','water'
MATL_MASS_FRACTION(1,1:2) = 0.9,0.1 /

&MATL ID                                = 'water'
EMISSIONIVITY                           = 1.0
DENSITY                                 = 1000.
CONDUCTIVITY                            = 0.20
SPECIFIC_HEAT                           = 4.184
N_REACTIONS                             = 1
REFERENCE_TEMPERATURE                   = 100.
PYROLYSIS_RANGE                         = 10.
HEATING_RATE                            = 5.
SPEC_ID                                 = 'WATER VAPOR'
NU_SPEC                                 = 1.
HEAT_OF_REACTION                        = 2500. /

&MATL ID                                = 'stuff'
EMISSIONIVITY                           = 1.0
DENSITY                                 = 500.
CONDUCTIVITY                            = 0.20
SPECIFIC_HEAT                           = 1.0
N_REACTIONS                             = 1
REFERENCE_TEMPERATURE                   = 300.
PYROLYSIS_RANGE                         = 80.
HEATING_RATE                            = 5.
NU_SPEC                                 = 0.8
SPEC_ID                                 = 'FUEL'
NU_MATL                                 = 0.2
MATL_ID                                 = 'ash'
HEAT_OF_REACTION                        = 1000. /

&MATL ID                                = 'ash'
EMISSIONIVITY                           = 1.0
DENSITY                                 = 500.
CONDUCTIVITY                            = 0.20
SPECIFIC_HEAT                           = 1.0 /

```

Figure 8.3: Input parameters for sample case **pyrolysis_2**.

that allows the triggering of reaction when temperature gets "above" $T_{thr,ij}$ ($S_{thr,ij} = +1$) or "below" $T_{thr,ij}$ ($S_{thr,ij} = -1$). $n_{t,j}$ is prescribed under the name `N_T(j)` and $S_{thr,ij}$ under `THRESHOLD_SIGN`.

By default, $T_{thr,ij}$ is -273.15 degrees Celsius, $n_{t,j}$ is zero and $S_{thr,ij} = +1$; thus, the last term of Equation 8.7 does not affect the pyrolysis rate. The term can be used to describe a threshold temperature for the pyrolysis reaction by setting $T_{thr,ij}$ and $n_{t,j} = 0$. Then the term is equal to 0 at temperatures below $T_{thr,ij}$ and 1 at temperatures above.

The threshold temperature can be used to simulate simple phase change reactions, such as melting and freezing. To make the reaction rate controlled by available energy, i.e. *not* kinetics, another optional term should be included in the reaction rate formula

$$r_{ij} = A_{ij} \frac{1}{H_{r,\alpha\beta}\Delta t} \max [0, S_{thr,ij}(T_s - T_{thr,ij})]^{n_{t,ij}} \quad (8.8)$$

This form of reaction rate can be implemented by setting a logical parameter `PCR(j) = .TRUE.`. The pre-exponential factor A_{ij} should then be given a value that is close or slightly smaller than the specific heat

(kJ/kgK) of the material mixture at phase change temperature.

As an example, consider a small amount of liquid water at +10°C, cooled down to -10°C during a 10 min period, and then heated up again to +10°C. The input lines for implementing the freezing and melting reactions are given in Fig. 8.4, and the complete input file as **water_ice_water.fds**. The mass fraction of the liquid water as a function of temperature is plotted in Fig. 8.5. The cooling phase is indicated by the blue line and heating phase by the red line.

```
&SURF ID = 'SAMPLE'
      H_FIXED = 50.
      BACKING = 'INSULATED'
      THICKNESS = 0.00001
      MATL_ID = 'LIQUID_WATER' /

&MATL ID = 'LIQUID_WATER'
      DENSITY = 1000.
      CONDUCTIVITY = 0.60
      SPECIFIC_HEAT = 4.19
      N_REACTIONS = 1
      PCR(1) = .TRUE.
      MATL_ID(1,1) = 'ICE_WATER'
      NU_MATL(1,1) = 1.
      A(1) = 2.0
      E(1) = 0.
      N_S(1) = 0.0
      N_T(1) = 1.0
      THRESHOLD_SIGN = -1
      THRESHOLD_TEMPERATURE(1) = 2.
      HEAT_OF_REACTION(1) = -333. /

&MATL ID = 'ICE_WATER'
      DENSITY = 1000.
      CONDUCTIVITY_RAMP = 'KS_ICE'
      SPECIFIC_HEAT_RAMP = 'CS_ICE'
      N_REACTIONS = 1
      PCR(1) = .TRUE.
      MATL_ID(1,1) = 'LIQUID_WATER'
      NU_MATL(1,1) = 1.
      A(1) = 2.0
      E(1) = 0.
      N_S(1) = 0.0
      N_T(1) = 1.0
      THRESHOLD_TEMPERATURE(1) = -2.
      HEAT_OF_REACTION(1) = 333. /

&RAMP ID = 'KS_ICE' T = -50. F = 2.76 /
&RAMP ID = 'KS_ICE' T = 0. F = 2.22 /
&RAMP ID = 'CS_ICE' T = -50 F = 1.751 /
&RAMP ID = 'CS_ICE' T = 0 F = 2.050 /
```

Figure 8.4: Input parameters for sample case **water_ice_water**.

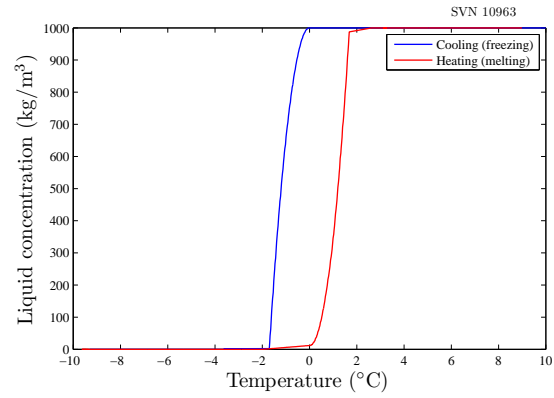


Figure 8.5: Freezing and melting of water.

8.4.5 Liquid Fuels

For a liquid fuel, the thermal properties are similar to those of a solid material, with a few exceptions. The evaporation rate of the fuel is governed by the Clausius-Clapeyron equation (see FDS Technical Reference Guide for details). The drawback of this approach is that the fuel mass flux is not an explicit function of temperature, but rather an iterative result depending on the temperature and flow conditions. To initiate the evaporation, an initial value of the fuel vapor volume flux is needed. If the initial value is (relatively) high, the evaporation starts regardless of any ignition source, and the fuel begins burning at once.

Figure 8.6 contains the key input parameters to describe a steel pan filled with a thin layer of ethanol. Note that the material properties are not all traceable to a measurement.

```
&MATL ID          = 'ETHANOL LIQUID'
  EMISSIVITY       = 1.0
  SPEC_ID          = 'FUEL'
  NU_SPEC          = 0.97
  HEAT_OF_REACTION = 880.
  CONDUCTIVITY     = 0.17
  SPECIFIC_HEAT    = 2.45
  DENSITY          = 787.
  ABSORPTION_COEFFICIENT = 40.
  BOILING_TEMPERATURE = 76. /

&MATL ID          = 'STEEL'
  EMISSIVITY       = 1.0
  DENSITY          = 7850.
  CONDUCTIVITY     = 45.8
  SPECIFIC_HEAT    = 0.46 /

&MATL ID          = 'CONCRETE'
  DENSITY          = 2200.
  CONDUCTIVITY     = 1.2
  SPECIFIC_HEAT    = 0.88 /

&SURF ID          = 'ETHANOL POOL'
  FYI              = '4 kg of ethanol in a 0.7 m x 0.8 m pan'
  COLOR            = 'YELLOW'
  MATL_ID          = 'ETHANOL LIQUID', 'STEEL', 'CONCRETE'
  THICKNESS        = 0.0091, 0.001, 0.05
  TMP_INNER        = 18., 18., 15. /
```

Figure 8.6: Input parameters for sample case **ethanol_pan**.

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 gaseous fuel yield, `NU_SPEC`, is 0.97 instead of 1 to account for impurities in the liquid that do not take part in the combustion process.

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 initial value of the fuel vapor volume flux can be specified using the parameter `INITIAL_VAPOR_FLUX`. Its default value is 5×10^{-4} m/s.

Note also the `ABSORPTION_COEFFICIENT` for the liquid. This denotes the absorption in depth of

thermal radiation. Liquids do not just absorb radiation at the surface, but rather over a thin layer near the surface. Its effect on the burning rate is significant.

In the current implementation of the liquid fuel model, the evaporation rate is strongly grid dependent. Thus, it should be used with caution.

8.4.6 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 :

$$t_b = \frac{\rho_s \delta_s}{\dot{m}''} \quad (8.9)$$

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

If a heat release rate `RAMP` function is not included for a solid fuel that burns at a specified rate, the surface will continue to burn at the specified rate indefinitely with no fuel burnout. If detailed heat release rate versus time data is not available, you can estimate the burnout time for a surface using the heat of combustion, ΔH , material density, ρ_s , material thickness, δ_s , and `HRRPUA`, \dot{q}_f'' :

$$t_b = \frac{\rho_s \delta_s \Delta H}{\dot{q}_f''} \quad (8.10)$$

Use the `RAMP` function to stop the burning once the calculated burnout time is reached.

The burnout time of a reacting solid fuel is calculated automatically by FDS based on the layer `THICKNESS`, component `DENSITY`, and the calculated burning rate.

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.

Special topic: Making Fuels Disappear (`BURN_AWAY`)

If a burning object is to disappear from the calculation once it is consumed, set `BURN_AWAY = .TRUE.` on the corresponding `SURF` line. The solid object disappears from the calculation cell by cell, as the mass contained by each mesh cells is consumed either by the pyrolysis reactions or by the prescribed `HRR`. The mass of each mesh cell is the cell face area multiplied by the surface density of the `SURF` type. The following issues should be kept in mind when using `BURN_AWAY`:

- For reacting surfaces, the surface density is computed as a sum of the layer densities multiplied by the layer thicknesses. This value can be over-ridden by setting `SURFACE_DENSITY` on the `SURF` line. For surfaces with prescribed `HRR` (`HRRPUA`), `SURFACE_DENSITY` parameter is the only way of defining the mass of the object.
- For surfaces with prescribed `HRR` (`HRRPUA`) or prescribed mass loss rate (`MLRPUA`) AND thermally thick heat conduction, the mass flux is affected by the heats of combustion defined for the gas phase reaction and the first listed material (`MATL`) component.

- Use `BURN_AWAY` parameter cautiously. If an object has the potential of burning away, a significant amount of extra memory has to be set aside to store additional surface information as the rectangular block is eaten away.
- If `BURN_AWAY` is prescribed, the `SURF` should be applied to the entire object, not just a face of the object because it is unclear how to handle edges of solid obstructions that have different `SURF_IDS` on different faces.
- 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/m^3) can be applied to the `OBST` rather than the `SURF` line. This parameter is used to determine 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 from which a combustible mass would be calculated.
- The mass of the object is based on the densities of the 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.

Example Case: Fires/box_burn_away

These are examples of a solid block of “foam” that is pyrolyzed until it is completely consumed. The heat flux is generated by placing hot surfaces around the box. There is no combustion. In the first example, `box_burn_away1`, the released gas is (`'METHANE'`), and in the second example, `box_burn_away2`, it is an additional species called `'GAS'`. In the third and fourth examples `box_burn_away3` and `box_burn_away4`, the released gas is fuel but the pyrolysis rate is specified. In the fourth case, the heat of combustion for the foam material is set different from that of the gas, with ratio 0.75. The properties of the block of foam were chosen simply to assure a quick calculation. The objective of the test is to check that the released mass and the integrated burning rate is consistent with the material properties of the block. The block is 0.4 m on a side, with a density of 20 kg/m^3 . The integrated densities of the pyrolysis product gases (written to `box_burn_away#_devc.csv`), as well as the integrated burning rate (written to `box_burn_away#_hrr.csv`) at the end of the 30 s calculation ought to be:

$$(0.4)^3 \text{ m}^3 \times 20 \text{ kg/m}^3 = 1.28 \text{ kg} \quad (8.11)$$

except for the fourth case, where the amount of released gas is affected by the ratio of heats of combustion

$$0.75 \times 1.28 \text{ kg} = 0.96 \text{ kg} \quad (8.12)$$

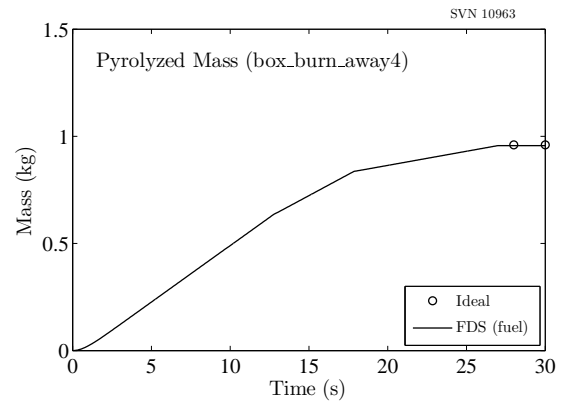
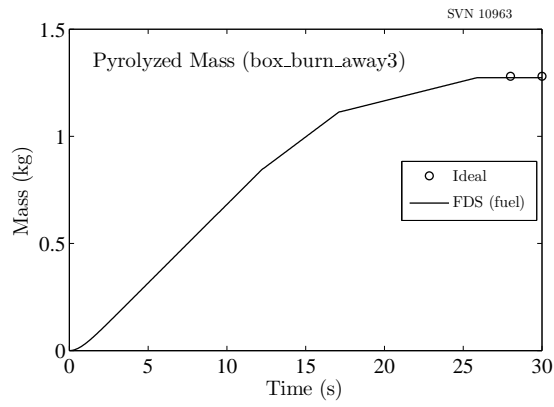
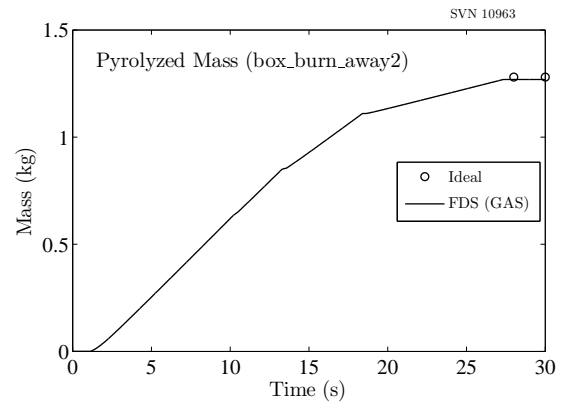
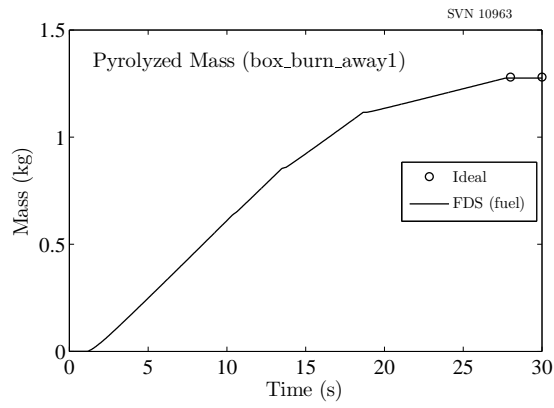


Figure 8.7: Output of box_burn_away test cases.

8.5 Testing Your Pyrolysis Model

Real materials that can burn can be very complicated. Undoubtedly, the `SURF` and `MATL` lines in the input file will 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, it is easy to perform a simple check of any set of surface properties by essentially turning off the gas phase – no combustion and no convective heat transfer. There are several parameters that allow you to do this, spread out over the various namelist groups.

1. Create a trivially small mesh, just to let FDS run. Since the gas phase calculation is essentially being shut off, you just need 4 cells in each direction (`IJK=4, 4, 4`) for the pressure solver to function properly.
2. 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 whatever value 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.
3. Put `H_FIXED=0.` on the `SURF` line. This turns off the convective heat flux from gas to surface and vice versa. The heat flux to the solid is specified via `EXTERNAL_FLUX`⁴ (kW/m²) on the `SURF` line that is assigned to the solid surface. If you want to specify a particular convective heat flux to the solid surface, you can set `ASSUMED_GAS_TEMPERATURE` on the `MISC` line, along with a non-zero value of `H_FIXED` on `SURF` in units of W/m²/K.
4. Turn off all the gas phase computations by setting `SOLID_PHASE_ONLY=.TRUE.` on the `MISC` line. This will also speed up the computations significantly. If the gas phase computations are needed, you may turn off combustion by creating a `REAC` line with only `Y_O2_INFTY=0.01`. This sets the background oxygen mass fraction to 0.01, too low to support any burning.
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 cone calorimeter.
6. Assign the `SURF_ID` to a `VENT` that spans the bottom of the computational domain. Create `OPEN` vents on all other faces.
7. Finally, add solid phase output devices to the solid surface, like `'WALL TEMPERATURE'`, `'NET HEAT FLUX'`, `'BURNING RATE'`, `'GAUGE HEAT FLUX'`, and `'WALL THICKNESS'` (assuming the solid is to burn away). Use these to track the condition of the solid as a function of time. In particular, make sure that the `'BURNING RATE'` is appropriate for the particular external heat flux applied. Make sure that the `'WALL TEMPERATURE'` is appropriate. 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 a short 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 52 kW/m² “external” heat flux. The solid in this example

⁴You can control `EXTERNAL_FLUX` using either `TAU_EF` or `RAMP_EF`. See Section 10 for more details.

is a 8.5 mm thick slab of PMMA. For more details, see the FDS Validation Guide under the heading “FAA Polymers.”

```
&HEAD CHID='FAA_Polymers_PMMA', TITLE='Black PMMA at 50 kW/m2, No Gas Phase Reaction' /

&MESH IJK=3,3,4, XB=-0.15,0.15,-0.15,0.15,0.0,0.4 /

&TIME T_END=600., WALL_INCREMENT=1, DT=0.01 /

&REAC FUEL='METHANE', Y_O2_INFTY=0.01 /

&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.
      HEAT_OF_COMBUSTION=25200.
      CONDUCTIVITY = 0.20
      SPECIFIC_HEAT = 2.2

&SURF ID='PMMA SLAB'
      COLOR='BLACK'
      BACKING='INSULATED'
      MATL_ID='BLACKPMMA'
      THICKNESS=0.0085
      H_FIXED=0.
      EXTERNAL_FLUX=52 / External Flux is ONLY for this simple demo exercise

&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='BURNING RATE', ID='MLR' /
&DEVC XYZ=0.0,0.0,0.0, IOR=3, QUANTITY='WALL THICKNESS', ID='thick' /

&TAIL /
```

8.6 Full-Scale Examples: Furniture

The example input files called `Fires/couch.fds` and `Fires/room_fire.fds` demonstrate how to model furniture. In residential fires, upholstered furniture makes up a significant fraction of the combustible load. A single couch can generate several megawatts of energy and sometimes lead to compartment flashover. Modeling a couch fire requires a simplification of its structure and materials. At the very least, we want the upholstery to be described as fabric covering foam:

```
&MATL ID          = 'FABRIC'
FYI               = 'Properties completely fabricated'
SPECIFIC_HEAT     = 1.0
CONDUCTIVITY      = 0.1
DENSITY           = 100.0
N_REACTIONS       = 1
SPEC_ID           = 'FUEL'
NU_SPEC           = 1.
REFERENCE_TEMPERATURE = 350.
HEAT_OF_REACTION  = 3000.
HEAT_OF_COMBUSTION = 15000. /

&MATL ID          = 'FOAM'
FYI               = 'Properties completely fabricated'
SPECIFIC_HEAT     = 1.0
CONDUCTIVITY      = 0.05
DENSITY           = 40.0
N_REACTIONS       = 1
SPEC_ID           = 'FUEL'
NU_SPEC           = 1.
REFERENCE_TEMPERATURE = 350.
HEAT_OF_REACTION  = 1500.
HEAT_OF_COMBUSTION = 30000. /

&SURF ID          = 'UPHOLSTERY'
FYI               = 'Properties completely fabricated'
COLOR             = 'PURPLE'
BURN_AWAY         = .TRUE.
MATL_ID(1:2,1)    = 'FABRIC','FOAM'
THICKNESS(1:2)    = 0.002,0.1
PART_ID           = 'smoke' /
```

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 specified by the user on the `REAC` line. The attribute `BURN_AWAY` forces FDS to break up the couch into individual cell-sized blocks that will disappear from the calculation as soon as the fuel is exhausted. The surface is specified as consisting of two layers, with a thickness of 2 mm for the `FABRIC` and 10 cm for the `FOAM`. The 10 cm is chosen to be the same as the mesh cell size.

The same couch model is included in a room-scale fire simulation, known as the **room_fire** test case. Figure 8.8 shows the fire after 5 and 10 minutes, respectively. Note that after 5 minutes, the couch is fully-involved, and after 10 minutes the room has flashed over. Only the reaction zone of the fire is shown; the smoke is hidden so that you can see the fire progressing from the couch to the doorway at the right of the scene. This door is the only opening to the compartment, and after 10 minutes, the flames can be seen flowing out.

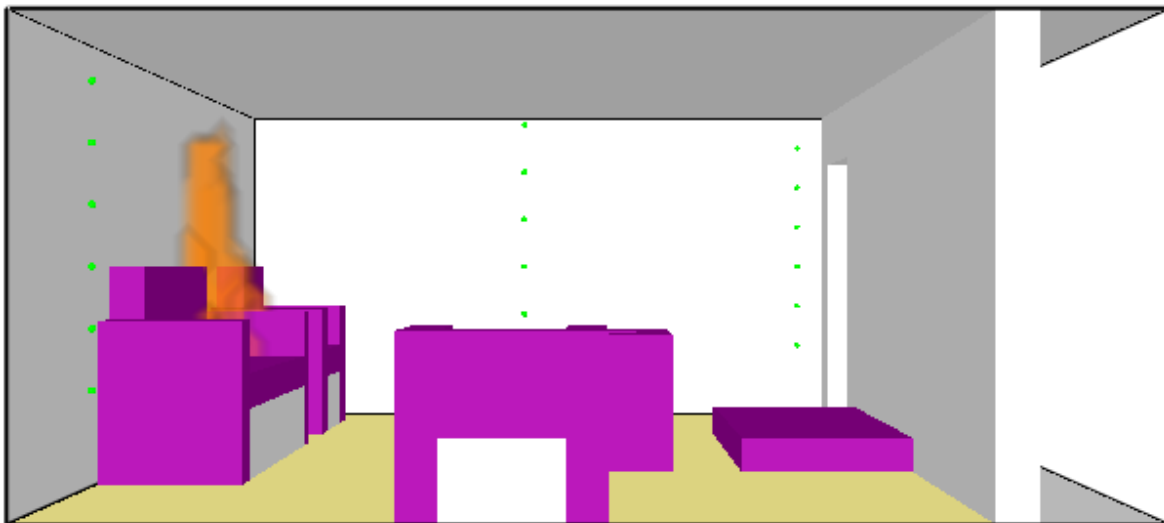
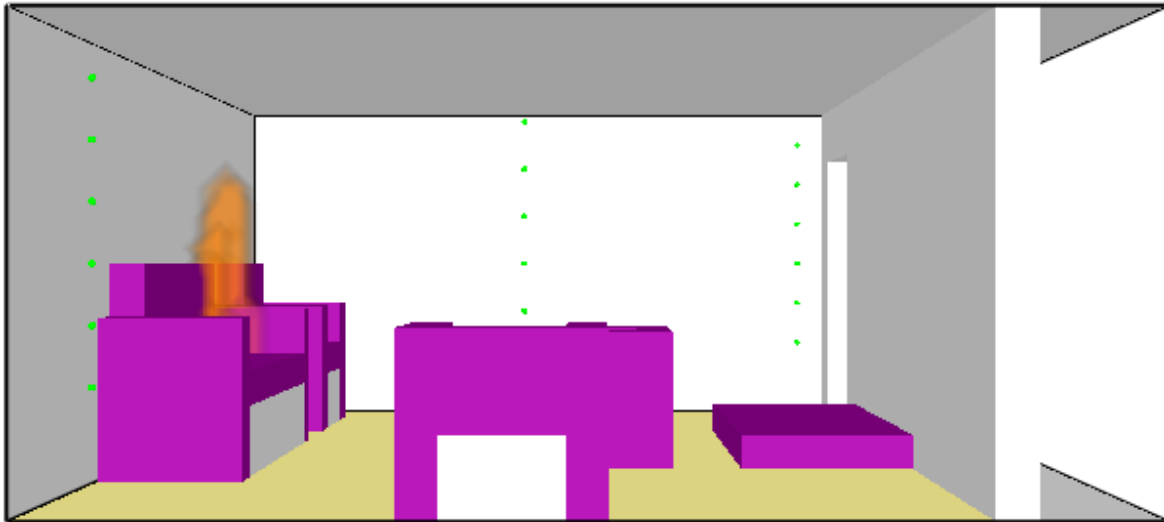


Figure 8.8: Output of **room_fire** test case showing fire after 5 and 10 minutes, respectively.

Chapter 9

Ventilation

This chapter explains how to model a ventilation system. There are two ways to do this. First, if you only want to specify air flow rates into and out of compartments, read Section 9.1 for a description of simple velocity boundary conditions. However, if you want to model the entire HVAC system, read Section 9.2.

9.1 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.

9.1.1 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_FLUX can be prescribed instead of VEL. The units are m³/s. If the flow is entering the computational domain, VOLUME_FLUX should be a negative number, the same convention as for VEL. Note that a SURF with a VOLUME_FLUX 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_FLUX divided by the area of that particular face. For example:

```
&SURF ID='SUPPLY', VOLUME_FLUX=-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 either `VEL` or `VOLUME_FLUX` should be prescribed, not both. The choice depends on whether an exact velocity is desired at a given vent, or whether the given volume flux is desired.

9.1.2 Total Mass Flux

Most often, you specify a simple supply or exhaust vent by setting either a normal velocity or volume flux at a solid surface. However, you may wish to control the mass flow rate (kg/s), as opposed to the volume flow rate (m^3/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.

9.1.3 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.

9.1.4 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

```
&SURF ID='LOUVER', VEL=-1.2, VEL_T=0.5, -0.3 /
```

is a boundary condition for a louvered vent that pushes air into the space with a normal velocity of 1.2 m/s and a tangential velocity of 0.5 m/s in either the x or y direction and -0.3 m/s in either the y or z direction, depending on what the normal direction is.

In cases of limited mesh resolution, it may not be possible to describe a louvered vent or slot diffuser using `VEL_T` because there may not be enough mesh cells spanning the opening. 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 9.2.6 for details on how to specify the louver.

9.1.5 Special Topic: 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=.TRUE.` by default). For an LES (Large Eddy Simulation), the Werner-Wengle wall model is applied for smooth walls and a log law is applied for rough walls. The surface roughness (in meters) is set by `ROUGHNESS` on `SURF`. See the FDS Technical Reference Guide [1] for wall model details. To force a solid boundary to have a free-slip condition, set `FREE_SLIP=.TRUE.` on the `SURF` line. In LES, to override the wall model and force a no-slip boundary condition, set `NO_SLIP=.TRUE.` on the `SURF` line.

9.1.6 Species and Species Mass Flux Boundary Conditions

There are two species boundary conditions that can be specified (see Section 11.1 for details on inputting and using species). These boundary conditions are `MASS_FLUX(:)` and `MASS_FRACTION(:)`. If a simple no-flux condition is desired at a solid wall, do not set anything. If the mass fraction of the species is to be some value at a forced flow boundary (`VEL` or `MASS_FLUX_TOTAL`) set `MASS_FRACTION(:)` equal to the desired mass fraction 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, no `VEL` should be set. It is automatically calculated based on the 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 $\text{kg/m}^2/\text{s}$.

Note that specifying `MASS_FRACTION(:)`, sets the “ghost” cell values for the species mass fractions. Since the mass conservation equation is an advection-diffusion equation, if the specified velocity is small, then the diffusion term can dominate resulting in an unintended mass flux of species. To obtain a guaranteed mass flux of a species, you should use `MASS_FLUX(:)`

9.2 HVAC Systems: The HVAC Namelist Group (Table 16.9)

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, 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. The current 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). 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. You can select from three fan models.

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', ... /
```

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', or 'AIRCOIL'.

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.

Note: It is recommended that for anything but the most simple HVAC systems (where a constant flow is explicitly specified), that you keep STRATIFICATION=.TRUE..

9.2.1 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., ROUGHNESS=0.001, FAN_ID='fan 1', DEVC_ID='device 1' /
```

All possible duct inputs are given below:

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 m². DIAMETER (m) can be used instead.

CTRL_ID is the ID for a CTRL for a damper, fan, or aircoil in the duct.

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.

DEVC_ID is the ID of a DEVC for a damper, fan, or aircoil in the duct.

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 (m³/s) through the duct. If you specify VOLUME_FLOW, you can change its value in time either using the characteristic time, TAU_VF, to define a tanh (TAU_VF > 0) or t² ramp (TAU_VF < 0); or you can specify a RAMP_ID. The operation of the fan can be controlled by either a device or a control function.

LOSS is a pair of real numbers giving the forward and reverse flow loss in the duct. Forward is flow from the first node listed in NODE_ID to the second node listed in NODE_ID.

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. REVERSE is a logical parameter that when .TRUE. indicates that the specified FAN_ID or VOLUME_FLOW blows from the second node to the first.

ROUGHNESS is the absolute roughness in m of the duct that is used to compute the friction factor for the duct.

Note that only one of AIRCOIL_ID, DAMPER_ID, or FAN_ID should be specified for a duct.

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. For example the top of Figure 9.1 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. 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 and, 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, a representative area or diameter. The pressure losses associated with all the segments of the duct can be collapsed in to in single loss by summing all of the fitting, K , losses through the duct as follows:

$$K_{eff} = \sum_i K_i \frac{A_{eff}}{A_i} \quad (9.1)$$

where i is a fitting, j is a duct segment, and A is the area of the duct or fitting.

9.2.2 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_ARRAY=lossarray, XYZ=x,y,z /
&HVAC TYPE_ID='NODE', ID='FDS connection', DUCT_ID='duct 1', VENT_ID='vent',
      LOSS_ARRAY=enter,exit /
&HVAC TYPE_ID='NODE', ID='ambient', DUCT_ID='duct 1', LOSS_ARRAY=enter,exit,
      XYZ=x,y,z /
```

All possible duct node inputs are given below.

AMBIENT is a logical value. If .TRUE., 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. 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 can only have two connected ducts.

LOSS is an n by n array of real numbers giving the flow losses for the node. $LOSS(I, J)$ is the loss for flow from duct I to duct J expressed in terms of the downstream duct area (see discussion in 9.2.1 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 for flow entering the HVAC system and for flow exiting the HVAC system.

VENT_ID is the name of the VENT where the node connects to the FDS computational domain.

XYZ is a triplet of real numbers giving the coordinates of the node. This location is used to compute buoyancy heads. If the node is connected to the FDS domain, then do not specify XYZ. 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 = .TRUE. or a VENT_ID specified. When defining a VENT as a component of an HVAC system you must set SURF_ID = 'HVAC' and you must set the ID for the VENT. Note that you cannot give more than one VENT the same ID.

9.2.3 HVAC Fan Parameters

Below are given sample inputs for the three types of fans supported by FDS.

```
&HVAC TYPE='FAN', ID='constant volume', DEVC_ID='device 1', VOLUME_FLOW=1.0, LOSS=2./
&HVAC TYPE='FAN', ID='quadratic', DEVC_ID='device 1',
      MAX_FLOW=1., MAX_PRESSURE=1000., LOSS=2. /
&HVAC TYPE='FAN', ID='user fan curve', RAMP_ID='fan curve', DEVC_ID='device 1', LOSS=2. /
```

All fan inputs are described below.

CTRL_ID Name of a control function controlling the operation of the fan.

DEVC_ID Name of a device controlling the operation of the fan.

LOSS is the flow loss through the fan when it is not operational.

MAX_FLOW is the maximum volumetric flow of the fan in m^3/s . This input activates a quadratic fan model.

MAX_PRESSURE is the stall pressure of the fan in Pa. This input activates a quadratic fan model.

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 or is an optional table of time versus volumetric flow rates (m^3/s) for a constant volume fan.

TAU_FAN defines a tanh ($TAU_FAN > 0$) or t^2 ramp ($TAU_FAN < 0$) for the fan. This is applied to the flowrate 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).

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 Section 9.1 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:

$$\dot{V}_{\text{fan}} = \dot{V}_{\text{max}} \text{sign}(\Delta p_{\text{max}} - \Delta p) \sqrt{\frac{|\Delta p - \Delta p_{\text{max}}|}{\Delta p_{\text{max}}}} \quad (9.2)$$

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 \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 m^3/s at Δp_{max} . If the pressure difference increases beyond this, air will be forced backwards through the fan. Figure 9.2 displays a typical fan curve.

More complicated fan curves can be specified by defining a RAMP.

Example Case: fan_test

Here is an example how fans can be specified. In it, two simple compartments share a common wall. Both compartments are considered as separate “pressure zones.” Two HVAC ducts are defined. One is a quadratic fan with a maximum volumetric flow of 0.16 m^3/s and a stall pressure of 10 Pa. The second is open duct with a flow loss of 10. The relevant input lines are:

```
&OBST XB= -0.15, 0.15,-1.0, 1.0, 0.0, 2.0 / Partition Wall

&VENT XB=-0.15,-0.15,-0.2, 0.2, 0.4, 0.6, COLOR='RED', ID='BLOW LEFT 1',
SURF_ID='HVAC',IOR=-1/
&VENT XB= 0.15, 0.15,-0.2, 0.2, 0.4, 0.6, COLOR='GREEN', ID='BLOW LEFT 2',
SURF_ID='HVAC',IOR=1/
&VENT XB=-0.15,-0.15,-0.2, 0.2, 1.4, 1.6, COLOR='BLUE', ID='BLOW RIGHT 1',
SURF_ID='HVAC',IOR=-1/
&VENT XB= 0.15, 0.15,-0.2, 0.2, 1.4, 1.6, COLOR='YELLOW', ID='BLOW RIGHT 2',
SURF_ID='HVAC',IOR=1/

&HVAC ID='LEFT 1', TYPE_ID='NODE', DUCT_ID='LEFT', VENT_ID='BLOW LEFT 1' /
&HVAC ID='LEFT 2', TYPE_ID='NODE', DUCT_ID='LEFT', VENT_ID='BLOW LEFT 2' /
&HVAC ID='RIGHT 1', TYPE_ID='NODE', DUCT_ID='RIGHT', VENT_ID='BLOW RIGHT 1' /
&HVAC ID='RIGHT 2', TYPE_ID='NODE', DUCT_ID='RIGHT', VENT_ID='BLOW RIGHT 2' /

&HVAC ID='LEFT', TYPE_ID='DUCT', NODE_ID='LEFT 1','LEFT 2',
LENGTH=1, AREA=0.04, FAN_ID='LEFT', REVERSE=.TRUE., LOSS=0,0 /
&HVAC ID='RIGHT', TYPE_ID='DUCT', NODE_ID='RIGHT 1','RIGHT 2',
```

```

LENGTH=1, AREA=0.04, LOSS=10,10 /
&HVAC ID='LEFT',TYPE_ID='FAN',MAX_FLOW=0.16,MAX_PRESSURE=10./

```

The volume flow through the fans is given by the expression:

$$\dot{V}_{\text{fan}} = \dot{V}_{\text{max}} \text{sign}(\Delta p_{\text{max}} - \Delta p) \sqrt{\frac{|\Delta p - \Delta p_{\text{max}}|}{\Delta p_{\text{max}}}} \quad (9.3)$$

where \dot{V}_{max} is the maximum output of the fan (0.16 m³/s from Zone 1 to Zone 2 and 0.2 m/s from Zone 2 to Zone 1), and Δp_{max} is the maximum pressure difference the fan can operate upon (in this case 10 Pa).

In steady state, the volume flow from compartment to compartment (or Zone to Zone) should be equal and opposite in sign. This occurs when the positive pressure added by the fan equals the pressure drop due to the flow loss through the return duct.

$$\sqrt{\frac{2|p_2 - p_1|}{(1.2 \text{ kg/m}^3)10}} (0.4 \text{ m/s}) = (0.16 \text{ m}^3/\text{s}) \sqrt{\frac{|p_1 - p_2 - 10 \text{ Pa}|}{10 \text{ Pa}}} \quad (9.4)$$

The solution is $p_2 = 4.5 \text{ Pa}$ and $p_1 = -4.5 \text{ Pa}$ (see Fig. 9.3). Note that the sign of the Volume Flow of a duct in FDS has to do with whether the flow is moving from the first node to the second (positive) or the second node to the first (negative). This convention can make these types of calculations a bit tricky.

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 `OBSTstruction` 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. 9.4).

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 can be positioned about halfway along the passage (if a louvered fan is being used, place the fan at the end of the passage).

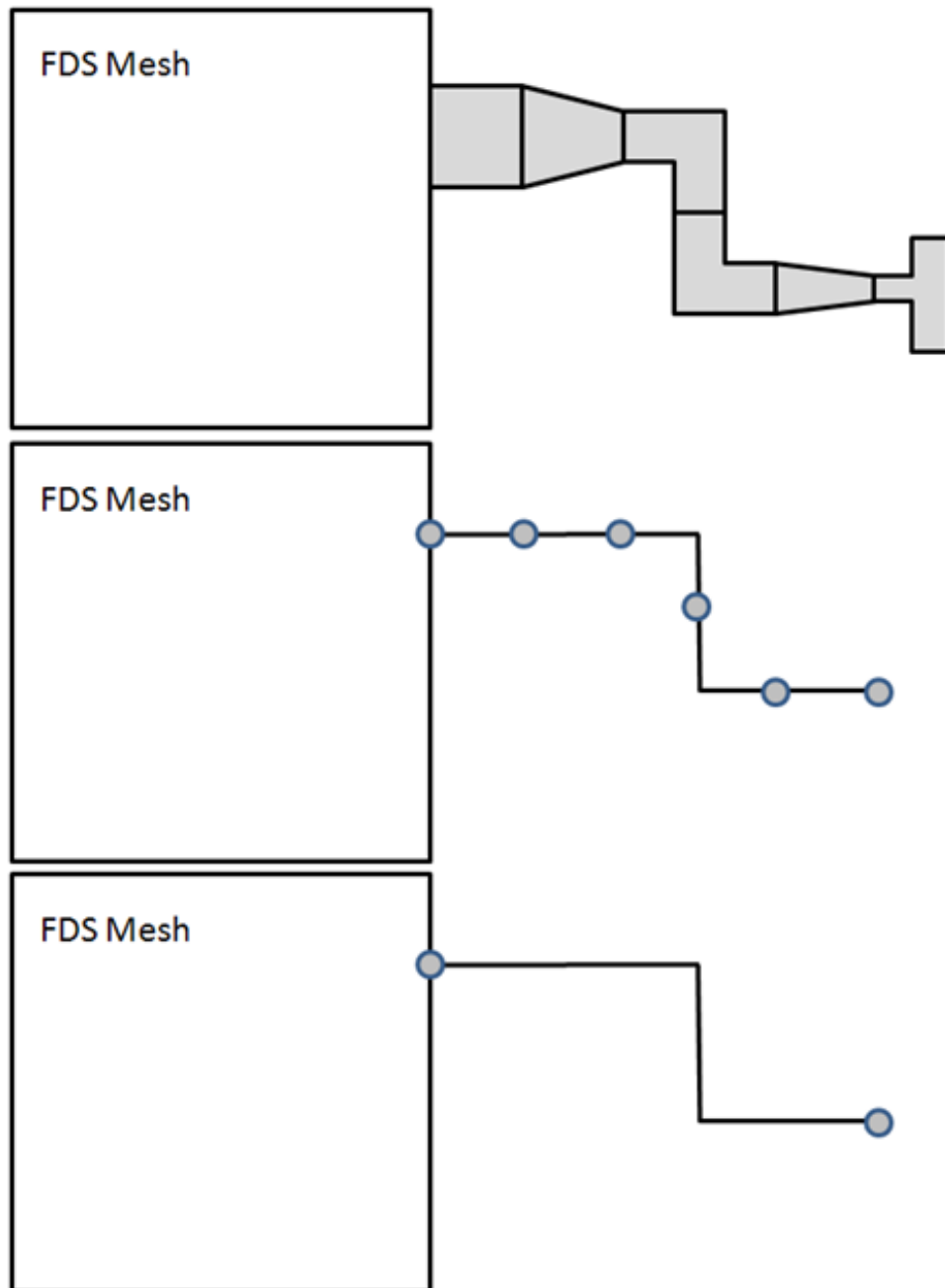


Figure 9.1: An example simplifying a complex duct.

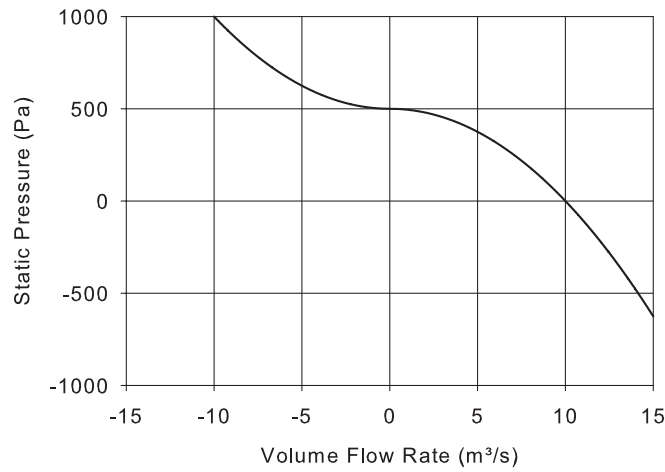


Figure 9.2: Fan curve corresponding to `VOLUME_FLUX=10` and `MAX_PRESSURE=500`. Note that a volume flux greater than 10 is brought about by a negative pressure difference; that is, when the downstream pressure is less than the upstream. Note also that when the pressure difference is greater than 500 Pa, the volume flow becomes negative; that is, the flow reverses.

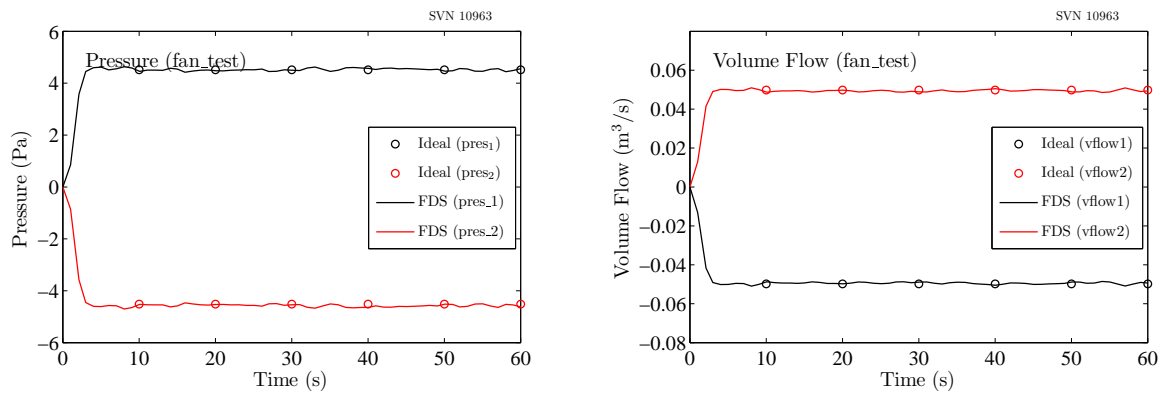


Figure 9.3: Pressure and volume flow for the **fan_test**.

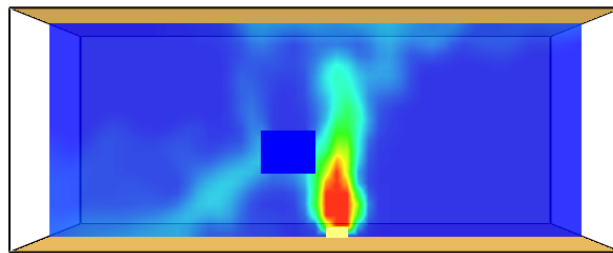


Figure 9.4: Jet fan with a louvered output $UVW=-1, 0, -1$

9.2.4 HVAC Filter Parameters: HVAC_filter

A sample input for a filter is given below.

```
&HVAC TYPE='FILTER', ID='filter 1', LOADING=0., SPEC_ID='SOOT',  
      EFFICIENCY=0.99, LOADING_MULTIPLIER=1, CLEAN_LOSS=2., LOSS=100./
```

All possible filter inputs are discussed below.

CLEAN_LOSS is the flow loss through the filter when it is new (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 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 loss.

LOSS Invokes a linear flow loss model where the flow loss is given as a linear function of the total loading, $K_{filter} = K_{CLEAN_LOSS} + K_{LOSS} \sum (L_i M_i)$, 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 (see summation in **LOSS**. Only one of **LOSS** or **RAMP_ID** should be specified.

SPEC_ID identifies the tracked species for the inputs of **LOADING_MULTIPLIER** and **LOADING**.

Note that a filter input refers to a class of filters and that multiple ducts can reference the same filter definition.

The sample case **HVAC_filter** demonstrates the use of the filter input. A 1 m³ compartment is initialized with a particulate species with a mass fraction of 0.001. A 100 % efficient filter with a clean loss of 1 and a loading loss of 7732.446 /kg (gives a total loss of 10 when all the soot in the compartment is filtered). A quadratic fan in a 0.01 m² duct with a maximum pressure of 20 Pa and a maximum flow of 0.2 m³/s takes suction from one side of the compartment and discharges into the other. Over time, the filter removes the particulate from the compartment. However, since the filter loss increases with the mass of particulate filtered, the rate of removal will decrease over time. Applying the conservation of energy to the compartment and the steady state duct momentum equation to the duct, we can solve for the compartment temperature and pressure and the duct velocity. These results along with the loading of the filter and the mass of particulate in the compartment are shown in Figure 9.5.

9.2.5 HVAC Aircoil Parameters: HVAC_aircoil

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 is provided below.

```
&HVAC TYPE='AIRCOIL', ID='aircoil 1', DEVC_ID='device 1', EFFICIENCY=0.5,  
      COOLANT_CP=4.186, COOLANT_TEMPERATURE= 10., COOLANT_MDOT= 1./
```

All possible aircoil inputs are shown below.

COOLANT_CP The specific heat (kJ/kg/K) of the working fluid.

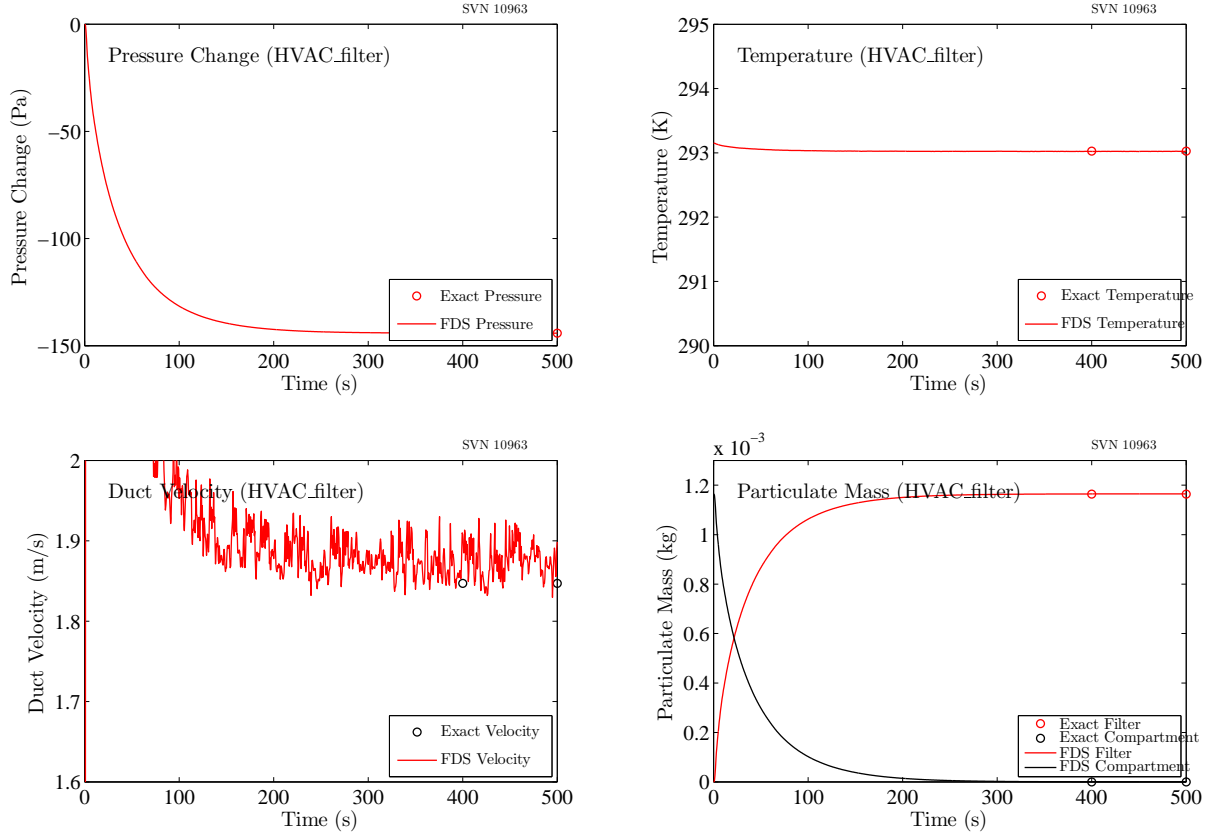


Figure 9.5: Compartment pressure (upper left), compartment temperature (upper right), duct velocity (lower left), and particulate mass in the compartment and on the filter (lower right)

COOLANT_MDOT The flow rate of the working fluid (kg/s).

COOLANT_TEMPERATURE The inlet temperature of the working fluid (°C).

CTRL_ID Name of a control function controlling the operation of the aircoil.

DEVC_ID Name of a device controlling the operation of the aircoil.

EFFICIENCY The heat exchanger efficiency, η , from 0 to 1. 1 indicates the exit temperatures on both sides of the heat exchanger will be equal.

FIXED_Q Constant heat exchange rate. Negative is heat removal from the duct.

Note that either `FIXED_Q` or the set `COOLANT_CP`, `COOLANT_MDOT`, `COOLANT_TEMPERATURE`, and `EFFICIENCY` should be specified. In the later case the heat exchange is computed as:

$$T_{fluid,out} = \frac{c_{p,gas}u_{duct}A_{duct}\rho_{duct}T_{duct,in} + c_{p,fluid}\dot{m}_{fluid}T_{fluid,in}}{c_{p,gas}u_{duct}A_{duct}\rho_{duct} + c_{p,fluid}\dot{m}_{fluid}} \quad (9.5)$$

$$\dot{q}_{coil} = \eta c_{p,fluid}\dot{m}_{fluid} (T_{fluid,in} - T_{fluid,out}) \quad (9.6)$$

Note that an aircoil input refers to a class of aircoils and that multiple ducts can reference the same aircoil definition.

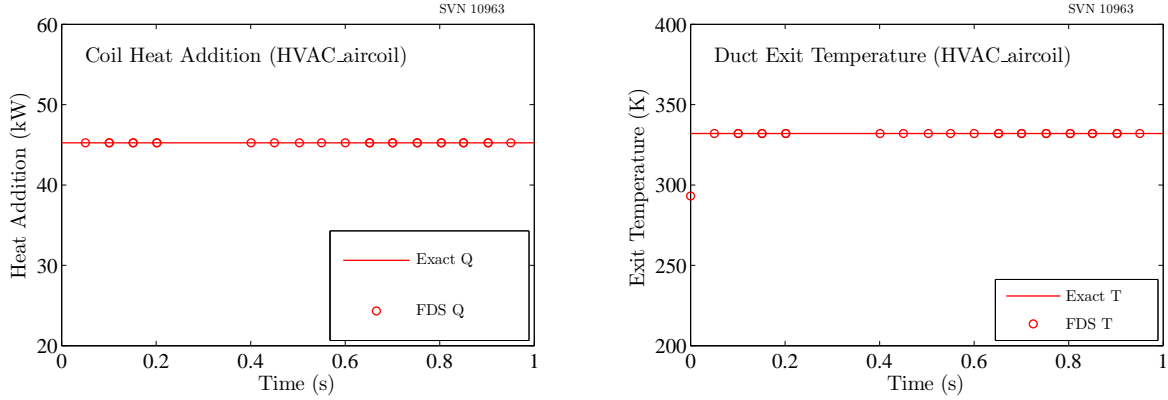


Figure 9.6: (Left) Heat addition and (Right) duct exit temperature for the HVAC_aircoil case.

The sample case **HVAC_aircoil** 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 $1 \text{ m}^3/\text{s}$. 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 Figure 9.6.

9.2.6 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.

9.3 Pressure-Related Effects: The ZONE Namelist Group (Table 16.31)

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. You can specify any number of sealed compartments within the computational domain that can have their own “background” pressures, and these compartments, or “pressure zones,” can be connected via leakage and duct paths whose flow rates are tied to the pressure of the adjacent zones.

9.3.1 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 currently no algorithm within FDS to identify these zones based solely on your specified obstructions. Consequently, it is necessary that you identify these zones explicitly in the input file. The basic syntax for a pressure `ZONE` is:

```
&ZONE XB=0.3,1.2,0.4,2.9,0.3,4.5 /
```

This means that the rectangular region, $0.3 < x < 1.2$, $0.4 < y < 2.9$, $0.3 < z < 4.5$, is assumed to be within a sealed compartment. 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. If the sealed region is not rectangular, FDS will extend the specified `ZONE` boundaries to conform to the non-rectangular region. It is possible to “break” pressure zones by removing obstructions between them. An example of how to break pressure zones is given below. Second, pressure zones **can** span multiple meshes, but it is recommended that you check the pressure in each mesh to ensure consistency. Also, if the `ZONE` does span multiple meshes, make sure that the specified rectangular coordinates do so as well. This allows FDS to determine the actual extent of the `ZONE` independently for each mesh.

Note that if you plan to have one zone open up to another via the removal of an obstruction, make sure that the coordinates of the two zones abut (i.e. touch) even if one of the zones includes the solid obstruction that separates them. FDS recognizes that a zone boundary has been removed when two adjacent cells belonging to two different zones have no solid obstruction between them. It is recommended that you extend at least one of the zone boundaries *into* the solid obstruction separating the two zones. That way, when the obstruction is removed, the newly created gas phase cells will be assigned to one or the other zone and it will become obvious that two adjacent gas phase cells are of two different zones, at which point the zones will merge and no longer have distinct background pressures.

Example Case: pressure_rise

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 $0.1 \text{ kg/m}^2/\text{s}$ 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 `.FALSE.` 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. We expect the pressure, temperature and density to rise during the 60 s injection period. Afterwards, the temperature, density, and pressure should remain constant, according to the equation of state. The figures below show 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.

Example Case: zone_break

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

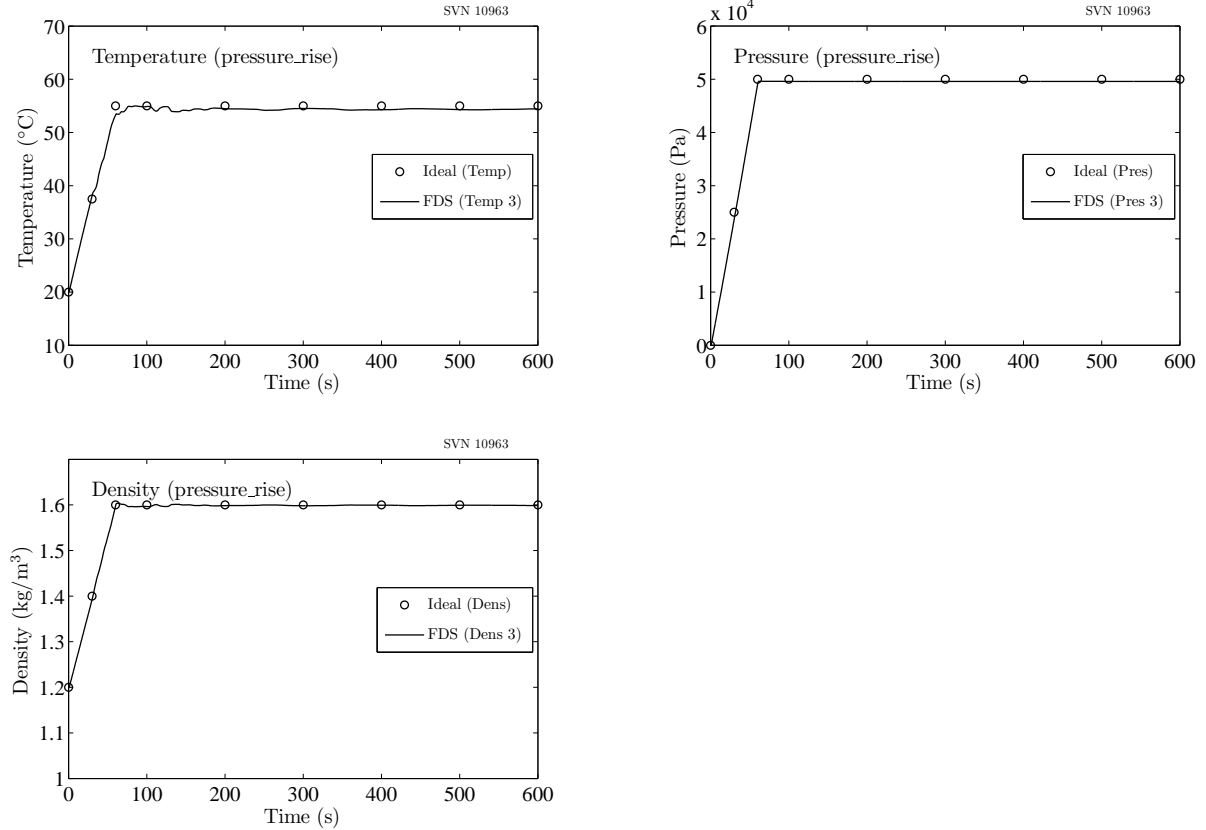


Figure 9.7: Output of **pressure_rise** test case.

rate of 0.1 kg/s, increasing its pressure approximately 1970 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 9.8 displays the pressure within each compartment.

Notice that the pressure within each compartment does 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 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:

Since pressure zones are defined using `XB`, it might not be possible to define a complexly shaped set of pressure zones using just the `ZONE` inputs. A work around for this is to define the complex zone as a series of zones in the same manner you would divide a domain into multiple meshes. The check for merged pressure zones only works if two zones were initially isolated at the start of the calculation (there must have been a wall that was removed). Therefore, define an obstruction at the boundary of the zones that is removed after the first timestep.

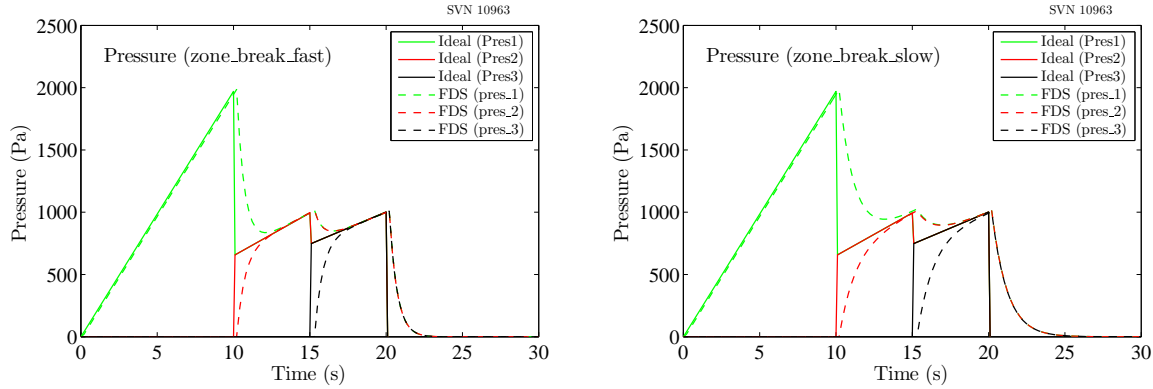


Figure 9.8: Output of **zone_break** test cases. The figure on the left results from using a pressure relaxation time of 0.5 s. The figure on the right uses 1 s, the default.

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.

9.3.2 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 sub-grid 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 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. Leakage from one compartment to another is then designated on the input lines defining the individual pressure **ZONES**:

```
&ZONE XB=0.3,1.2,0.4,2.9,0.3,4.5, LEAK_AREA(0)=0.0001 /
&ZONE XB=2.3,5.8,1.4,2.9,6.8,9.7, LEAK_AREA(1)=0.0002, LEAK_AREA(0)=0.0003 /
```

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 vis 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.

The volume flow, \dot{V} , through a leak of area A_L is given by

$$\dot{V}_{\text{leak}} = A_L \text{sign}(\Delta p) \sqrt{2 \frac{|\Delta p|}{\rho_\infty}} \quad (9.7)$$

where Δp is the pressure difference between the adjacent compartments (in units of Pa) and ρ_∞ is the ambient density (in units of kg/m^3). The discharge coefficient normally seen in this type of formula is assumed to be 1.

Within FDS, leakage flows are handled by the HVAC solver. A leak between two zones is considered to be a pair of HVAC VENTS connected by a duct with a loss coefficient of 1. To specify a leak between two zones, a solid surface in each of the zones must be given the attribute LEAK_PATH. Leakage is uniformly distributed over all of the solid surfaces assigned the LEAK_PATH. In the above example, the leakage from Zone 1 to Zone 0 requires the attribute LEAK_PATH=1, 0, meaning that the leak between Zones 1 and 0 is uniformly distributed over solids defined with:

```
&SURF ID='whatever', ..., LEAK_PATH=1,0 /
```

Likewise, the boundaries of Zone 1 and Zone 2 must include solids whose SURF properties include the parameter LEAK_PATH=1, 2, but these solids need not form a boundary between the two zones. The SURFACES with the LEAK_PATH attribute lump all of the leakage over these areas. The order of the two pressure zones designated by LEAK_PATH is unimportant.

The HVAC output quantities can be used to determine the leakage flows. FDS give the duct connecting Zone A with Zone B the name 'LEAK A B' and the duct nodes will be named 'LEAK A B' for the Zone A side of the leak and 'LEAK B A' for the Zone B side of the leak. Note that for the duct names, FDS will use the lower numbered zone as Zone A.

9.3.3 Special Topic: Stack Effect

Tall buildings often experience buoyancy-induced air movement due to temperature differences between inside and outside, known as *stack effect*. The different temperatures result in different densities and hence different pressure gradients inside the building compared to outside the building. This can result in inducing flow in vertical shafts (stairwells, atriums, elevator shafts, etc.) through leakage paths or openings at the top and bottom of the building. 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 `LAPSE_RATE` (°C/m) that is entered on the `MISC` line.

For the case where the stack flow is through small leakage paths, it is recommended to define the building using a pressure `ZONE`. The leakage paths can then be defined as individual HVAC systems. Note that the leakage model would aggregate all leaky surfaces over the entire height of the building and as a result average out the pressure gradients. For doing stack effect calculations individual leakage paths should be defined. In the section that follows is a simple example.

Example Case: Atmospheric_Effects/stack_effect

This subsection contains a simple example demonstrating the stack effect. A detailed discussion of the stack effect can be found in [18].

The **stack_effect** test case is a two-dimensional simulation of a building with an interior hotter than the exterior and two leakage paths at the top and the bottom of the building. Since the inside is hotter, it will have a smaller pressure gradient than the outside resulting in higher pressure inside the top of the building than the outside. This will cause flow out the upper leakage path and in the bottom leakage path.

The building 100 m tall building with an interior temperature, T_b , initially set to 20 °C, and an exterior temperature, T_∞ , set to 10 °C. 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 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 m² and a flow loss 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:

$$\frac{\rho_0(z)}{\rho_\infty} = \exp\left(-\frac{g\bar{W}}{\mathcal{R}T_0}z\right) \quad (9.8)$$

where \mathcal{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/m³, respectively.

Since the openings in the building are equally spaced over its height, the neutral plane should be close to its midpoint. By making some simple assumptions including using only the ground level ambient pressure for the equation of state and that the densities are constant with height, a hand calculation can be done to determine the flows through the leakage paths. Note that in actuality both pressure and density will change with height; however, as seen in the above densities, these changes are small.

The pressure difference across the building's wall can be computed as

$$\Delta p = \frac{\bar{W} p_0(z) g h}{\mathcal{R}} \left(\frac{1}{T_\infty} - \frac{1}{T_b} \right) \quad (9.9)$$

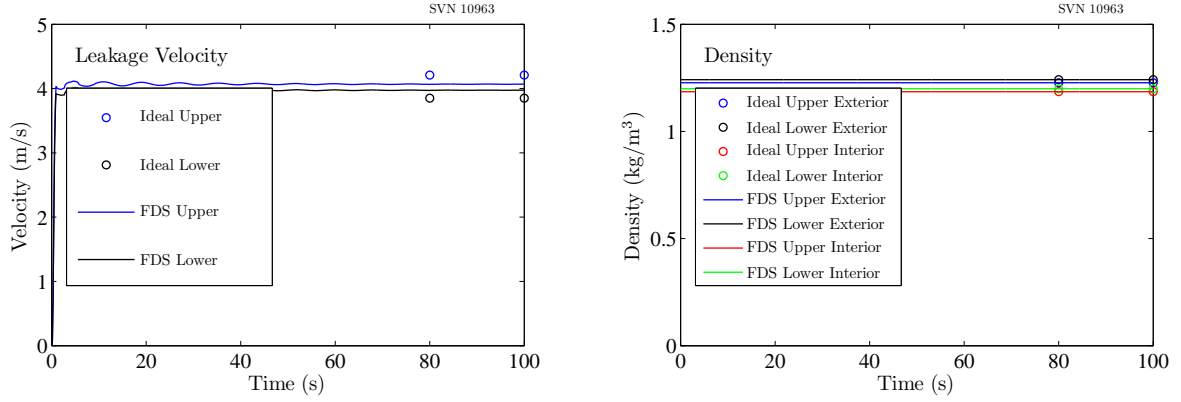


Figure 9.9: (Left) Velocity at the upper and lower vents for the stack_effect case. (Right) Upper and lower exterior and interior densities for the stack_effect case.

where h is the distance from the neutral plane. The pressure gradient will cause flow in and out of the leaks and the neutral plane is located where the mass flows in and out are balanced. This can be computed using:

$$\frac{H}{H_n} = 1 + \frac{T_\infty}{T_b} \quad (9.10)$$

where H_n is the neutral plane height above the bottom of the shaft and H is the height between the vents. This gives a neutral plane of 46.79 m and pressure differences of 44.3 Pa and 50.7 Pa at the top and bottom of the vent. Using the loss of 2 and the pressure difference in the HVAC momentum equation results in a steady-state velocity of inflow velocity at the bottom is 3.85 m/s and an outflow velocity at the top is 4.21 m/s.

Results for velocity and density are shown in Figure 9.9.

9.4 Special Topic: 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) \mathcal{H} is specified as

$$\begin{aligned} \mathcal{H} &= \text{DYNAMIC_PRESSURE} / \rho_\infty + |\mathbf{u}|^2 / 2 \quad (\text{outgoing}) \\ \mathcal{H} &= \text{DYNAMIC_PRESSURE} / \rho_\infty \quad (\text{incoming}) \end{aligned} \quad (9.11)$$

where ρ_∞ is the ambient density and \mathbf{u} is the most recent value of the velocity on the boundary. The `PRESSURE_RAMP` allows you to alter 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.

Example Case: pressure_boundary

The following lines, taken from the sample case, **pressure_boundary**, demonstrates how to do 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 9.10 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.

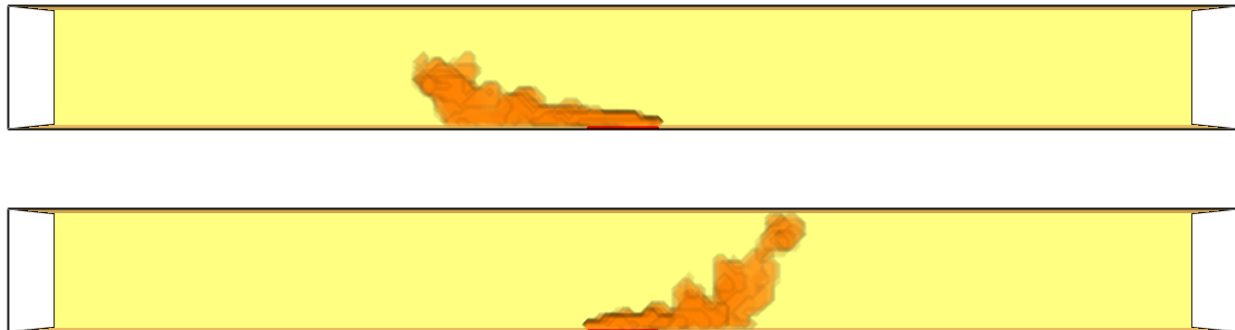


Figure 9.10: Snapshots from the sample case **pressure_boundary** showing a fire in a tunnel leaning left, then right, due to a positive, then negative, pressure imposed at the right portal.

9.5 Special Topic: Fires and Flows in the Outdoors

Simulating a fire in the outdoors is not much different than a fire indoors, but there are a few issues that need to be addressed. First, the velocity of the wind profile at any exterior boundary will be a top hat (constant) by default, but the parameter `PROFILE` on the `SURF` line can yield other profiles. For example, `PROFILE='PARABOLIC'` produces a parabolic profile with `VEL` being the maximum velocity, and `'ATMOSPHERIC'` produces a typical atmospheric wind profile of the form $u = u_0(z/z_0)^p$. 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 some height where the wind speed is measured, like an elevated weather station. It is assumed that the ground is located at $z = 0$. To change this assumption, set `GROUND_LEVEL` on the `MISC` line to be the appropriate value of z . Be careful not to apply an atmospheric velocity profile below `GROUND_LEVEL` or FDS will stop with an error.

Another useful parameter for outdoor simulations is the temperature lapse rate of the atmosphere. Typically, in the first few hundred meters of the atmosphere, the temperature decreases several degrees Celsius per kilometer. These few degrees are 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 `MISC` line in units of $^{\circ}\text{C}/\text{m}$. 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 $0^{\circ}\text{C}/\text{m}$.

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=.FALSE.` on the `MISC` line.

Chapter 10

User-Specified Functions, Ramps and Tables

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. These functions can be complex, thus you have to have some way to convey them. The namelist groups, `RAMP` and `TABL`, allow you to control the behavior of selected 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).

10.1 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. When the calculation starts temperatures, velocities, burning rates, *etc.*, are ramped-up from their starting values because nothing can happen instantaneously. By default, everything is ramped-up to their prescribed values in roughly 1 s. However, control the rate at which things turn on, or turn off, by specifying time histories for the boundary conditions that are listed on a given `SURF` line. The above boundary conditions can be made time-dependent using either prescribed functions or 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_FLUX`), or `MASS_FLUX_TOTAL` are to ramp up to their prescribed values in `TAU` seconds and remain there. If `TAU_Q` is positive, then the heat release rate ramps up like $\tanh(t/\tau)$. If negative, then the HRR ramps up like $(t/\tau)^2$. If the fire ramps up following a t^2 curve, it remains constant after `TAU_Q` seconds. These rules apply to `TAU_T` and `TAU_V` as well. The default value for all `TAUs` is 1 s. If something other than a \tanh or t^2 ramp up is desired, then a user-defined function must 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 /
.
.
.
&RAMP ID='rampname2', T= 0.0, F=0.0 /
&RAMP ID='rampname2', T=10.0, F=0.3 /
&RAMP ID='rampname2', T=20.0, F=0.8 /
.
```

·
·

Here, T is the time, and F indicates the fraction of the heat release rate, wall temperature, velocity, mass fraction, *etc.*, to apply. Linear interpolation¹ 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, the simple blowing vent from above 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 /
```

Note that the temperature and velocity can be independently controlled by assigning different RAMPs via RAMP_T and RAMP_V, respectively.

Use TAU_T or RAMP_T to control the ramp-ups for surface temperature. The surface temperature will be computed as

$$T(t) = T_0 + f(t)(T_f - T_0) \quad (10.1)$$

where $T(t)$ is the desired surface temperature, $f(t)$ is the result of evaluating the RAMP_T at time t , T_0 is the ambient temperature, and T_f is TMP_FRONT, 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 . Here, the N refers to the N th species listed on the SURF line. 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/m²/s and 0.3 kg/m²/s over time periods of approximately 5 s and 10 s, respectively. It does not matter in what order the SPEC lines are listed in the input file.

¹By 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 10.1: Parameters for controlling the time-dependence of given quantities.

Quantity	Group	Input Parameter(s)	TAU	RAMP ID
Heat Release Rate	SURF	HRRPUA	TAU_Q	RAMP_Q
Temperature	SURF	TMP_FRONT	TAU_T	RAMP_T
Velocity	SURF	VEL, VOLUME_FLUX, MASS_FLUX_TOTAL	TAU_V	RAMP_V
Mass Fraction/Flux	SURF	MASS_FRACTION (N) , 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

10.2 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 here (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.

10.3 Tabular Functions

Some input quantities, such as a sprinkler spray pattern, vary multi-dimensionally. In such cases, use the TABL namelist group. The format of the TABL lines is application-specific, but in general look like this:

```
&TABL ID='TABLE1', TABLE_DATA=40,50, 85, 95,10,0.5 /
&TABL ID='TABLE1', TABLE_DATA=40,50,185,195,10,0.5 /
```

A detailed description of the various table entries is given in the sections that describe quantities that use such tables. Currently, only sprinklers and nozzles use this group of parameters to define a complex spray pattern.

Note that each set of `TABL` lines must have a unique `ID`. Specific requirements on ordering the lines will depend upon the type of `TABL` and those requirements are provided in the appropriate section in this guide.

Chapter 11

Chemistry

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.

There are two namelist groups that describe gas species. The first is `SPEC`, which describes each gas species that is included in the simulation. 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. Or a gas species might just serve as the “background” species; that is, it is the gas that is present at the start of the simulation. Or, a gas species might be one component of a mixture of gases that are transported together. 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.” The default combustion model in FDS assumes that the reaction is mixing-controlled, and transport equations for only the lumped species – air, fuel, and products – are solved. 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 lumped species. More detail on combustion is given in Chapter 12. 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.

11.1 Specifying Gas Species: The `SPEC` Namelist Group

There are three ways in which a species can be used in FDS: it can be tracked as a separate species, it can be the background species, or it can only exist as part of a lumped species defined with `SMIX`. If a species only exists as part of one or more lumped species, the parameter `SMIX_COMPONENT_ONLY=.TRUE.` should be set. 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=.TRUE.` should be set. This also tells FDS not to allocate space for the species in the array of tracked gases. A species with `SMIX_COMPONENT_ONLY=.TRUE.` cannot be used as an individual species and it cannot be used as the background species; however, the background species and an individual species can also be used as part of one or more lumped species.

Suppose that gases are introduced into the simulation that are neither reactants nor products of combustion. These gases can be tracked separately via additional scalar transport equation¹ In fact, there does not

¹ 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.

need to be any fire at all – FDS can be used to transport a mixture of non-reacting ideal gases.

11.1.1 Basics

The namelist group `SPEC` is used to specify an individual gas species. Each `SPEC` line should include at the very least the name of the species via a character string, `ID`. Next, if the ambient (initial) mass fraction² of the gas is something other than 0, then the parameter `MASS_FRACTION_0` is used to specify it. Several gases that can be included in a calculation are listed in Table 11.1. Here is an example:

```
&SPEC ID='ARGON', MASS_FRACTION_0=0.1, MW=40. /
```

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(:)`. Following is a very simple example of how a gas can be introduced into the simulation.

Sample Case: Flowfields/gas_filling

Consider the short input file:

```
&HEAD CHID='gas_filling', TITLE='Fill an Empty Room with Hydrogen' /
&MESH IJK=32,32,15, XB=-3.2,3.2,-3.2,3.2,0.0,3.0 /
&TIME T_END=300.0 /
&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=.TRUE. /
&SLCF PBY=0.0, QUANTITY='HYDROGEN' /
&TAIL /
```

The case is nothing more than hydrogen gas filling a box. The gas is injected through a 1 m by 1 m vent at a rate of 0.01667 kg/m²/s and shut off after 3 min. The total mass of hydrogen at that point ought to be 3 kg (see Fig. 11.1). Notice that no properties were needed for the `HYDROGEN` because it is a species whose properties are included in FDS. 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)`, and the index 1 refers to the first, and only, gas species that is specified in the input file. The parameter `MASS_FILE=.TRUE.` instructs FDS to produce an output file that contains a time history of the hydrogen mass.

11.1.2 Special Topic: Gas and Liquid Properties

There are several option for specifying the properties of gas and liquid species.

Option 1: JANAF Tables

Gases and liquids whose properties are tabulated within FDS are listed in Table 11.1. The physical properties of these species are known and do not need to be specified.

²HUMIDITY can be specified for `WATER VAPOR` in place of mass fraction. `HUMIDITY` is the relative humidity of water vapor in units of %. It is 40 % by default.

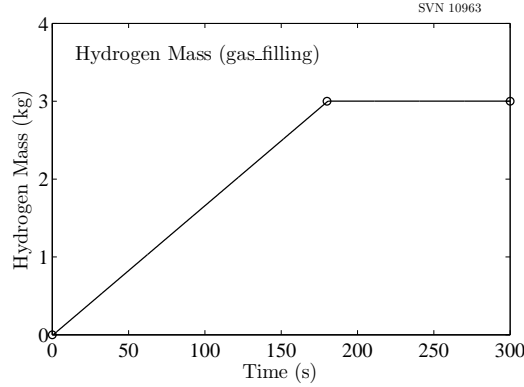


Figure 11.1: Hydrogen mass vs. time for **gas_filling** test case.

Option 2: Lennard-Jones Potential Parameters

If the gas species is not included in Table 11.1, its molecular weight, MW , must be specified on the `SPEC` line in units of g/mol. In addition, if known, the Lennard-Jones potential parameters σ (`SIGMALJ`) and ϵ/k (`EPSILONKLJ`) should be specified.

Option 3: User-Specified Properties

If the gas species is not included in Table 11.1, and the Lennard-Jones potential parameters are not known, you must specify its molecular weight, MW (g/mol), its `VISCOSITY` (kg/m/s), its (thermal) `CONDUCTIVITY` (W/m/K), and its `DIFFUSIVITY` (m²/s). The diffusivity is assumed to be the binary diffusion coefficient between the given species and the background species. If any of these properties are omitted, FDS will assume the properties of air. The specific heat of the gas will be calculated from its molecular weight using the relation:

$$c_{p,\alpha} = \frac{\gamma}{\gamma - 1} \frac{\mathcal{R}}{W_\alpha} \quad (11.1)$$

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=.TRUE.` on the `MISC` line.

Additional Parameters

It is assumed that the enthalpy of the gas species is given by the following formula:

$$h(T) = h(T_{ref}) + \int_{T_{ref}}^T c_p(T) dT \quad (11.2)$$

The (optional) `REFERENCE_TEMPERATURE`, T_{ref} , is the temperature that corresponds to the `REFERENCE_ENTHALPY`. If `SPECIFIC_HEAT` is specified and the `REFERENCE_ENTHALPY` is not, the `REFERENCE_ENTHALPY` will be set to $h(T_{ref}) = c_p T_{ref}$.

If a liquid droplet is to be evaporating into a gas species and properties are being provided, then the following relationship should hold:

$$h_{gas}(T_{boil}) = h_{liquid}(T_{boil}) + h_v \quad (11.3)$$

Recognized species that are emissive will be defined as `ABSORBING` and radiative absorption for those species will be computed. The keyword `ABSORBING` can be specified on the `SPEC` line as well. If `.TRUE.` and the species is not in the recognized list, then it will be assumed to be a fuel when invoking RadCal to compute its absorptivity.

Liquids

If the species listed in Table 11.1 includes liquid properties, it can be applied to liquid droplets. More detail is included in Chapter 14.

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, IUPAC³ 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' /
```

³International Union of Pure and Applied Chemistry

Table 11.1: **Optional Gas Species** [19]

Species	Mol. Wgt. (g/mol)	Formula	σ (Å)	ε/k (K)	Liquid
ACETYLENE	26.037280	C ₂ H ₂	4.033	231.8	
ACROLEIN	56.063260	C ₃ H ₄ O	4.549	576.7	Y
AIR	28.848523		3.711	78.6	Y
ARGON	39.948000	Ar	3.42	124.0	Y
BUTANE	58.122200	C ₄ H ₁₀	4.687	531.4	Y
CARBON DIOXIDE	44.009500	CO ₂	3.941	195.2	
CARBON MONOXIDE	28.010100	CO	3.690	91.7	Y
ETHANE	30.069040	C ₂ H ₆	4.443	215.7	Y
ETHANOL	46.068440	C ₂ H ₅ OH	4.530	362.6	Y
ETHYLENE	28.053160	C ₂ H ₄	4.163	224.7	Y
FORMALDEHYDE	30.025980	CH ₂ O	3.626	481.8	Y
HELIUM	4.002602	He	2.551	10.22	Y
HYDROGEN	2.015880	H ₂	2.827	59.7	Y
HYDROGEN BROMIDE	80.911940	HBr	3.353	449.0	Y
HYDROGEN CHLORIDE	36.460940	HCl	3.339	344.7	Y
HYDROGEN CYANIDE	27.025340	HCN	3.63	569.1	Y
HYDROGEN FLUORIDE	20.006343	HF	3.148	330.0	Y
ISOPROPANOL	60.095020	C ₃ H ₇ OH	4.549	576.7	Y
METHANE	16.042460	CH ₄	3.758	148.6	Y
METHANOL	32.041860	CH ₂ OH	3.626	481.8	Y
N-DECANE	142.281680	C ₁₀ H ₂₂	5.233	226.46	
N-HEPTANE	100.201940	C ₇ H ₁₆	4.701	205.75	Y
N-HEXANE	86.175360	C ₆ H ₁₂	5.949	399.3	Y
N-OCTANE	114.228520	C ₈ H ₁₈	4.892	231.16	Y
NITRIC OXIDE	30.006100	NO	3.492	116.7	Y
NITROGEN	28.013400	N ₂	3.798	71.4	Y
NITROGEN DIOXIDE	46.05500	NO ₂	3.992	204.88	Y
NITROUS OXIDE	44.012800	N ₂ O	3.828	232.4	Y
OXYGEN	31.998800	O ₂	3.467	106.7	Y
PROPANE	44.095620	C ₃ H ₈	5.118	237.1	Y
PROPYLENE	42.079740	C ₃ H ₆	4.678	298.9	Y
SOOT	12.010700	C	3.798	71.4	
SULFUR DIOXIDE	64.063800	SO ₂	4.112	335.4	Y
SULFUR HEXAFLUORIDE	146.055419	SF ₆	5.128	146.0	
TOLUENE	92.138420	C ₆ H ₅ CH ₃	5.698	480.0	Y
WATER VAPOR	18.015280	H ₂ O	2.641	809.1	Y

11.2 Specifying Gas Mixtures: The SMIX Namelist Group

In typical fire simulations, there are mixtures of gas species that are created and transported together. Solving transport equations for each species wastes computational resources. For example, you can define air as a combination of nitrogen, oxygen, water vapor, and carbon dioxide. However, this would require solving four separate transport equations. Alternatively, you can define a “lumped species” that represents this mixture and save on CPU time. The SMIX namelist group allows you to define species mixtures.

The following inputs:

```
&SPEC ID='NITROGEN', BACKGROUND=.TRUE./  
&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 /
```

are equivalent to these inputs using SMIX

```
&SPEC ID='NITROGEN', SMIX_COMPONENT_ONLY=.TRUE. /  
&SPEC ID='OXYGEN', SMIX_COMPONENT_ONLY=.TRUE. /  
&SPEC ID='WATER VAPOR', SMIX_COMPONENT_ONLY=.TRUE. /  
&SPEC ID='CARBON DIOXIDE', SMIX_COMPONENT_ONLY=.TRUE. /  
  
&SMIX ID='AIR', BACKGROUND=.TRUE.,  
    SPEC_ID='NITROGEN', 'OXYGEN', 'WATER VAPOR', 'CARBON DIOXIDE'  
    MASS_FRACTION=0.76274,0.23054,0.00626,0.00046 /
```

The keyword `SMIX_COMPONENT_ONLY` denotes that the species is only present as part of a lumped species. When this flag is set, FDS will not allocate space to track that species individually.

The available inputs for SMIX are:

BACKGROUND Denotes that this lumped species is to be used as the background species. The `BACKGROUND` lumped species must be defined first.

ID Character string identifying the name of the species. You must provide this.

SPEC_ID Character array containing the names of the species that make up the lumped species.

MASS_FRACTION The mass fractions of the species making up the lumped species in the order listed by `SPEC_ID`. FDS will normalize the values to 1. Specify only one of `MASS_FRACTION` or `VOLUME_FRACTION`.

VOLUME_FRACTION The volume fractions of the species making up the lumped species in the order listed by `SPEC_ID`. FDS will normalize the values to 1. Specify only one of `MASS_FRACTION` or `VOLUME_FRACTION`.

Any FDS input that requires a `SPEC_ID` can also use the `ID` of an SMIX.

11.2.1 Background Species

If the simulation does not involve the simple chemistry model – either because no combustion is desired or if a finite rate reaction(s) is being specified (see Section 12.3) – you can specify that the background gas species be something other than air. For a gas mixture comprised of n species, FDS only solves transport equations for $n - 1$ because it also solves an equation for total mass conservation. To set the properties of the background species, use the `SPEC` or `SMIX` line with `BACKGROUND=.TRUE.` specified.

Chapter 12

Combustion

A common source of confusion in FDS is the distinction between gas phase *combustion* and solid phase *pyrolysis*. The former refers to the reaction of fuel vapor and oxygen; the latter the generation of fuel vapor at a solid or liquid surface. Whereas there can be many types of combustibles in an FDS fire simulation, in the simple chemistry, mixing-controlled combustion model there can only be one gaseous fuel. The reason is cost. It is expensive to solve transport equations for multiple gaseous fuels. Consequently, the burning rates of solids and liquids are automatically adjusted by FDS to account for the difference in the heats of combustion of the various combustibles. In effect, you specify a single gas phase reaction as a surrogate for all the potential fuel sources.

Combustion can be modeled in two ways. By default, the reaction of fuel and oxygen is infinitely fast and controlled only by mixing, hence the label *mixing-controlled*. The alternative is that the reaction is *finite-rate*. The latter approach usually requires very fine grid resolution is not practical for large-scale fire applications. This chapter describes both methods, with an emphasis on the more commonly used mixing-controlled model. The REAC namelist group contains the parameters for both modes of combustion.

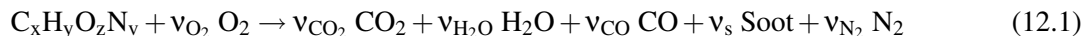
12.1 Single-Step, Mixing-Controlled Combustion

This approach to combustion considers a single fuel species that is composed primarily of C,H,O, and N that reacts with oxygen in one or more mixing-controlled steps to form H₂O, CO₂, soot, and CO. For the default simple chemistry model, only a single REAC line is needed.

Note that starting with FDS version 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.

12.1.1 Simple Chemistry Inputs

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 internally to determine the amount of

combustion products that are formed:

$$\begin{aligned}
 v_{O_2} &= v_{CO_2} + \frac{v_{CO}}{2} + \frac{v_{H_2O}}{2} - \frac{z}{2} \\
 v_{CO_2} &= x - v_{CO} - (1 - X_H) v_s \\
 v_{H_2O} &= \frac{y}{2} - \frac{X_H}{2} v_s \\
 v_{CO} &= \frac{W_f}{W_{CO}} y_{CO} \\
 v_s &= \frac{W_f}{W_s} y_s \\
 v_{N_2} &= \frac{v}{2} \\
 W_s &= X_H W_H + (1 - X_H) W_C
 \end{aligned}$$

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. When using simple chemistry, specifying FUEL will cause FDS to use the built-in thermophysical properties for that species when computing quantities such as specific heat or viscosity. This parameter is independent of the inputs for the fuel chemistry, i.e. C, H, O, N. Table 11.1 provides a listing of the available species. If the FUEL is in the table, then FDS will use the built-in formula to obtain the values of C,H,O, and N. By default, FDS uses the gas thermophysical properties of ETHYLENE for the fuel.

FORMULA A character string that identifies the chemical formula of the fuel species for the reaction. This input only has meaning when simple chemistry is being used and the formula can only contain C, H, O, or N. Specifying a formula means the individual inputs of C, H, O, and N do not need to be specified. See 11.1.2 for a description on how to input a FORMULA.

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.

C, H, O, N The fuel chemical formula. All numbers are positive. One of either C or H must be specified. This input is not needed if FORMULA is specified or if the FUEL is in Table 11.1.

Y_O2_INFTY Ambient mass fraction of oxygen (Default 0.232428)

Y_CO2_INFTY Ambient mass fraction of carbon dioxide (Default 0.0058)

HUMIDITY Relative humidity of the background air species, in units of %. (Default 40 %).

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.)

SOOT_H_FRACTION The fraction of the atoms in the soot that are hydrogen. For all other types of REAC inputs the species SOOT is assumed to be pure carbon unless a FORMULA is provided. Note that this parameter is only appropriate when the simple chemistry model is applied. (Default 0.1 - equivalent to the input FORMULA='C0.9H0.1' for other REAC types)

A few sample REAC lines are given here.

```
&REAC FUEL = 'METHANE' /
```

Here, there is no need for a FORMULA or atom count because the FUEL is listed in Table 11.1. It is also assumed that the soot and CO yields are zero.

```
&REAC FUEL          = 'PROPANE'  
      SOOT_YIELD     = 0.01  
      CO_YIELD       = 0.02  
      HEAT_OF_COMBUSTION = 46460. /
```

Here, the FORMULA is known, but the product yields are not. The heat of combustion is specified explicitly rather than calculated. See Section 12.1.2 for more details on the heat of combustion.

```
&REAC FUEL          = 'MY FUEL'  
      SOOT_YIELD     = 0.05  
      FORMULA        = 'C3H8O3N4'  
      HEAT_OF_COMBUSTION = 46124. /
```

Here, nothing is known or assumed.

12.1.2 Heat of Combustion

The HEAT_OF_COMBUSTION (kJ/kg) is the amount of energy released per unit mass of fuel consumed. Note that if the heat of combustion is not specified, it is assumed to be

$$\Delta H \approx \frac{v_{O_2} W_{O_2}}{W_f} \text{ EPUMO2} \quad \text{kJ/kg} \quad (12.2)$$

The quantity EPUMO2 (kJ/kg) is the amount of energy released per unit mass of oxygen consumed. Its default is 13,100 kJ/kg. Note that if both EPUMO2 and HEAT_OF_COMBUSTION are specified that FDS will ignore the value for EPUMO2.

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. With the simple chemistry combustion model, it is assumed that there is only one fuel. However, in a realistic fire scenario, 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.

The heat of combustion can be determined in a couple of ways. One approach is to take the difference in the heats of formation for the products (assuming complete combustion) and the reactants. This is typically how values are tabulated for pure fuels (e.g. one species) in handbooks. This ideal heat of combustion does not account for the SOOT_YIELD or CO_YIELD that occurs in a real fire. Carbon and hydrogen that go to soot and CO rather than CO₂ and H₂O result in a lower effective heat of combustion. Setting IDEAL=.TRUE. will reduce the HEAT_OF_COMBUSTION based upon the inputs for SOOT_YIELD and CO_YIELD. If EPUMO2 is specified instead of HEAT_OF_COMBUSTION, then the EPUMO2 will not be changed. The second 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=.FALSE., the measured value of the heat release rate includes the effects of any CO or soot that is produced and no adjustment is needed. The default value is IDEAL=.FALSE.

12.1.3 Special Topic: 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. This section provides a brief explanation of these rules and the parameters that control them.

In an LES simulation, the consumption rate of gaseous fuel is given by the expression:

$$\dot{m}_f''' = -\rho \min \left(Y_F, \frac{Y_{O_2}}{s}, \beta \frac{Y_P}{1+s} \right) \left(1 - e^{-\delta t / \tau} \right) \quad ; \quad s = \frac{W_F}{v_{O_2} W_{O_2}} \quad (12.3)$$

Here, τ is a mixing time scale and β is an empirical parameter. You can change the value of β by specifying `BETA_EDC`. Its default value is 2. There is a minimum value of Y_P required to allow for a reaction to occur before any products have accumulated. It is specified with `Y_P_MIN_EDC`, and its default value is 0.01.

12.1.4 Special Topic: 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 compartment is challenging because the relevant physical mechanisms occur at length scales smaller than a single mesh cell. Flames are extinguished due to lowered temperatures and dilution of the oxygen supply. A simple suppression algorithm has been implemented in FDS that attempts to gauge whether or not combustion is viable. The Technical Reference Guide [1] contains more details about how the mechanism works. The only parameter you can control is the `CRITICAL_FLAME_TEMPERATURE` set on the `REAC` line. The default value is 1427 °C. To eliminate any gas phase suppression, set `CRITICAL_FLAME_TEMPERATURE` to -273.15 °C, or turn off suppression completely by setting `SUPPRESSION=.FALSE.` on the `REAC` line. This latter approach saves on computing time because it prevents FDS from entering the suppression algorithm altogether.

If the mixing-controlled combustion model is used and flame extinction occurs, the unburned fuel gas will re-ignite when it mixes with oxygen somewhere else in the domain. To prevent this from happening, you can set the `AUTO_IGNITION_TEMPERATURE` (°C) below which combustion will not occur. Note that if this parameter is used, then some form of heat/ignition source must be present in order for combustion to begin.

Example Case: door_crack

This example uses a compartment surrounded by the ambient 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 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. As heat is lost to the walls, and as pressure is relieved, the pressure rise slows, and eventually the pressure decreases which allows the fan to push air back into the compartment. Eventually, the fire at the burner is self-extinguished due to lack of oxygen; however, the fuel-rich gases in the compartment will continue to burn where the fan is blowing fresh air into the compartment and at the leak where fuel rich gases exit the compartment.

While this case has a number of interesting physical effects, and it *verifies* several features of FDS, it is important to note that although there is smoke seen flowing backwards out the fan duct, in reality there would have been much more. Most conventionally built structures will not withstand over-pressures of

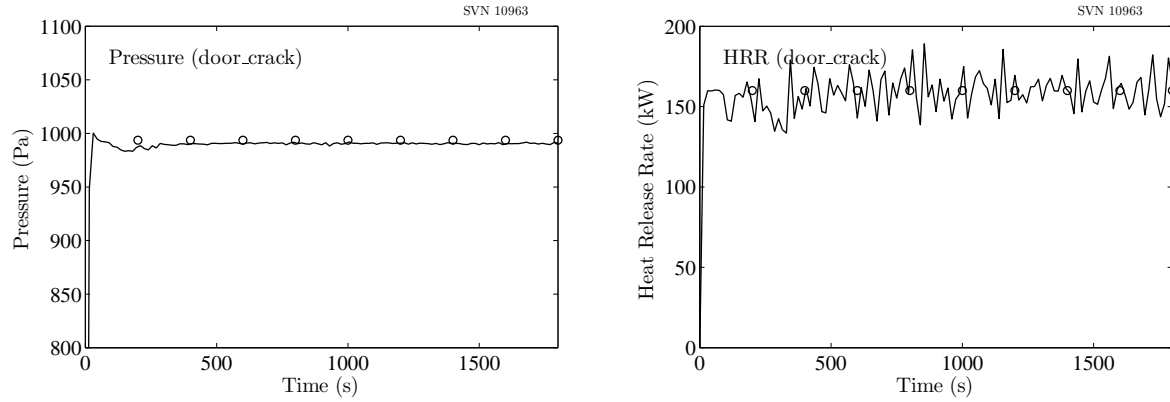


Figure 12.1: Output of **door_crack** test case.

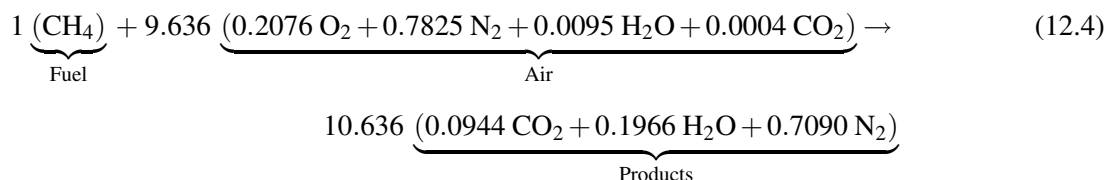
0.25 atm without some sort of relief. The fan and the crack under the door obey simple formulae based on pressure differences, but these assumptions have limits. Also, it is likely that the fire in this scenario would indeed extinguish itself as the oxygen volume fraction decreased below about 15 %. **But**, its re-ignition at the door crack and fan grill would depend on the presence of a spark or hot spot of some sort. FDS continues to flow fuel into the compartment past the point of local extinction, but the compartment cools. The default combustion algorithm in FDS assumes that in every grid cell there is a “virtual spark plug” that initiates combustion if fuel and oxygen are present.

12.2 Complex Stoichiometry

The “simple chemistry” inputs described above can only be used when there is a single mixing-controlled reaction and 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. The easiest way to explain this is by way of example. Consider a single reaction involving methane. When you specify the REAC line to be:

```
&REAC FUEL = 'METHANE' /
```

FDS actually 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',          SMIX_COMPONENT_ONLY=.TRUE. /
&SPEC ID='OXYGEN',            SMIX_COMPONENT_ONLY=.TRUE. /
&SPEC ID='WATER VAPOR',       SMIX_COMPONENT_ONLY=.TRUE. /
&SPEC ID='CARBON DIOXIDE',    SMIX_COMPONENT_ONLY=.TRUE. /
&SPEC ID='METHANE' /
&SMIX ID='AIR', SPEC_ID='OXYGEN','NITROGEN','WATER VAPOR','CARBON DIOXIDE',
VOLUME_FRACTION=0.2076,0.7825,0.0095,0.0004, BACKGROUND=.TRUE. /
&SMIX ID='PRODUCTS', SPEC_ID='CARBON DIOXIDE','WATER VAPOR','NITROGEN'
VOLUME_FRACTION=0.0944,0.1966,0.7090 /
&REAC FUEL='METHANE', SPEC_ID='METHANE','AIR','PRODUCTS',
NU=-1,-9.636,10.636, HEAT_OF_COMBUSTION=50000. /
```

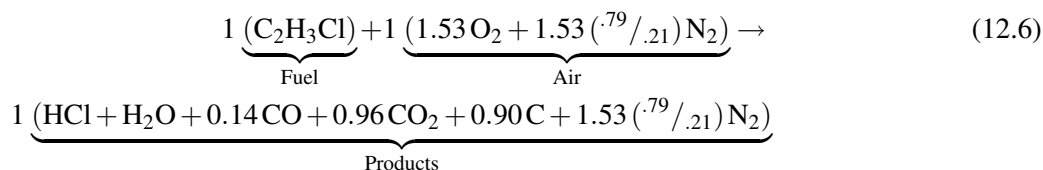
Note that the stoichiometry pertains to the species mixtures, and not the primitive species. Obviously, for this example, the first REAC line is preferable to the second. However, there are many situations where you cannot use the simple chemistry model and must resort to the second formulation.

12.2.1 Complex Fuel Molecules

Fires 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, we 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 $\text{C}_2\text{H}_3\text{Cl}$ for the purpose of modeling. To use PVC as the fuel molecule on the REAC line, and assuming that all of the Cl in the fuel is converted into HCl, do the following: Define the governing chemical reaction in terms of the primitive (or individual) species in the reaction. As a user, you need to determine the appropriate soot and CO yields for the specified fuel. In this example, the SFPE Handbook [20] was used to find soot and CO yields for PVC which are 0.172 and 0.063 respectively. For simple chemistry, the stoichiometric coefficient of species i can be found from its yield following:

$$\nu_i = \frac{W_f}{W_i} y_i \quad (12.5)$$

Since it is assumed that all of the Cl goes 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 Table 11.1, 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 transport by itself will have an SMIX_COMPONENT_ONLY = .TRUE. designation.

```
&SPEC ID = 'PVC', FORMULA = 'C2H3Cl' /
&SPEC ID = 'OXYGEN', SMIX_COMPONENT_ONLY = .TRUE. /
&SPEC ID = 'NITROGEN', SMIX_COMPONENT_ONLY = .TRUE. /
&SPEC ID = 'HYDROGEN CHLORIDE', SMIX_COMPONENT_ONLY = .TRUE. /
&SPEC ID = 'WATER VAPOR', SMIX_COMPONENT_ONLY = .TRUE. /
&SPEC ID = 'CARBON MONOXIDE', SMIX_COMPONENT_ONLY = .TRUE. /
&SPEC ID = 'CARBON DIOXIDE', SMIX_COMPONENT_ONLY = .TRUE. /
&SPEC ID = 'SOOT', SMIX_COMPONENT_ONLY = .TRUE. /
```

For the oxidizer and products, which are both composed of multiple primitive species, SMIX lines are needed to define the composition of the lumped species. As the user, you can define the SMIX by either the MASS_FRACTIONS of the component gases or the VOLUME_FRACTIONS. For this case, volume fractions will be used. If Equation 12.6 is properly balanced, you can directly use the stoichiometric coefficients of the primitive species to define the lumped species.

```
&SMIX ID = 'AIR', SPEC_ID = 'OXYGEN', 'NITROGEN', VOLUME_FRACTION = 1.53, 5.76,
BACKGROUND=.TRUE. /
&SMIX ID = 'PRODUCTS', SPEC_ID = 'HYDROGEN CHLORIDE', 'WATER VAPOR', 'CARBON MONOXIDE',
'CARBON DIOXIDE', 'SOOT', 'NITROGEN',
VOLUME_FRACTION = 1.0, 1.0, 0.14, 0.96, 0.90, 5.76 /
```

Note: the order of which the SPEC_IDs are defined must be the same order as which you defined your volume fractions or mass fractions. To set the initial concentration of fuel, an INIT line is used:

```
&INIT MASS_FRACTION(1)=0.229, SPEC_ID(1)='PVC' /
```

In this mixing-controlled example, you must define the fuel, heat of combustion, species participating the reaction, and the stoichiometric coefficients that describe the reaction. The heat of combustion for PVC was found from the SFPE Handbook [20]. In the lumped species approach, the stoichiometric coefficients for the SMIXs are the coefficients of the lumped species themselves.

```
&REAC FUEL = 'PVC', HEAT_OF_COMBUSTION=16400,
SMIX_ID = 'PVC', 'AIR', 'PRODUCTS', NU=-1, -1, 1
FIXED_MIX_TIME=0.1,
ODE_SOLVER= 'SINGLE EXACT' /
```

Note: The sign on NU corresponds to whether that species is consumed (-) or produced (+). The results, Figure 12.2, from this example problem are compared to expected results with both cases using a fixed turbulent mixing time of 0.1s.

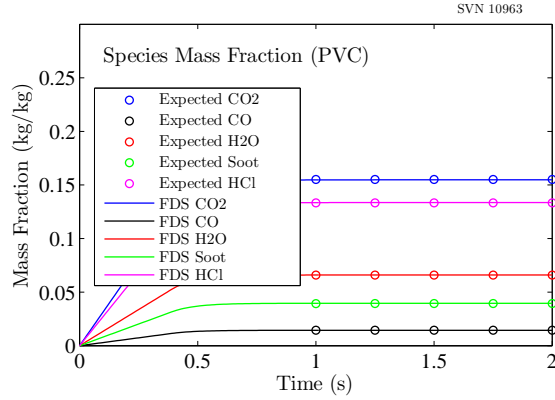
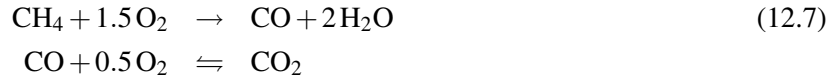


Figure 12.2: Product species mass fractions for model PVC example.

12.2.2 Two-Step Reaction Involving the Formation of CO

Consider the simplified two-step methane oxidation reaction developed by Westbrook and Dryer [21]:



In this case, if we use all primitive species and select nitrogen as the background species, Eq. (12.7) can be used directly. Each of the species in the reactions, plus the background species, must be defined by SPEC lines:

```
&SPEC ID='NITROGEN',      BACKGROUND=.TRUE. /
&SPEC ID='METHANE',       MASS_FRACTION_0 = 0.0 /
&SPEC ID='OXYGEN',        MASS_FRACTION_0 = 0.23 /
&SPEC ID='CARBON MONOXIDE', MASS_FRACTION_0 = 0.0 /
&SPEC ID='WATER VAPOR',   MASS_FRACTION_0 = 0.0 /
&SPEC ID='CARBON DIOXIDE', MASS_FRACTION_0 = 0.0 /
```

The initial mass fraction of oxygen is 0.23 with the remainder being the background, nitrogen. To define the reactions from Eq. (12.7), you must specify a REAC line for each, including the reversible reaction:

```
&REAC ID='FWD_METHANE'
  FUEL='METHANE'
  SMIX_ID='METHANE','OXYGEN','CARBON MONOXIDE','WATER VAPOR'
  NU=-1,-1.5,1,2
  HEAT_OF_COMBUSTION=32371.
  ODE_SOLVER='RK2 RICHARDSON' /

&REAC ID='FWD_CO'
  FUEL='CARBON MONOXIDE'
  SMIX_ID='CARBON MONOXIDE','OXYGEN','CARBON DIOXIDE'
  NU=-1,-0.5,1
  HEAT_OF_COMBUSTION=10102.6
  ODE_SOLVER='RK2 RICHARDSON' /

&REAC ID='REV_CO2'
  FUEL='CARBON DIOXIDE'
```



```

SMIX_ID='CARBON DIOXIDE','OXYGEN','CARBON MONOXIDE'
NU=-1,0.5,1
HEAT_OF_COMBUSTION=0
FWD_ID='FWD_CO'
REVERSIBLE=.TRUE.
ODE_SOLVER='RK2 RICHARDSON' /

```

In this example, the `SMIX_IDS` refer directly to the `SPEC` lines. The stoichiometric coefficients can be taken directly from Eq. (12.7). The heats of combustion are found by using the enthalpy of formation of each species¹:

$$\overline{\Delta h_c} = \overline{h}_{products} - \overline{h}_{reactants} = (\overline{h}_f^\circ + \overline{\Delta h})_{products} - (\overline{h}_f^\circ + \overline{\Delta h})_{reactants} \quad (12.8)$$

If the reference temperature is set as the ambient temperature, $\overline{\Delta h}$ drops out of the equation. As an example, the heat of combustion for the `FWD_METHANE` reaction can be calculated by:

$$\begin{aligned}
\overline{\Delta h_c} &= \nu_{CO} \overline{h}_{f_{CO}}^\circ + \nu_{H_2O} \overline{h}_{f_{H_2O}}^\circ - \nu_{CH_4} \overline{h}_{f_{CH_4}}^\circ - \nu_{O_2} \overline{h}_{f_{O_2}}^\circ \\
&= 1(-110.527) + 2(-241.826) - 1(-74.873) - 1(0.0) \\
&= -519.3 \text{ kJ/mol}
\end{aligned} \quad (12.9)$$

Change the units to kJ/kg by dividing by the molecular weight of the fuel. The heat of combustion becomes 32,371 kJ/kg. Note that in the third (reversible) reaction, the `HEAT_OF_COMBUSTION` is set to 0. The actual value is not 0, but it is set to 0 for purposes of this example. Below `HEAT_OF_COMBUSTION`, you will notice two inputs unique to reversible reactions: `FWD_ID` and `REVERSIBLE=.TRUE.`. These parameters indicate that the reaction is reversible and what its forward counterpart is. FDS then calculates the heat of combustion for the reverse reaction (determined from the product of the ratio of the molecular weights of the fuels and the heat of combustion of the forward reaction), applies the appropriate sign, and overwrites the value from the input file. Alternatively, you can specify directly the heat of combustion, in which case the third `REAC` line will look like this:

```

&REAC ID='REV_CO2'
FUEL='CARBON DIOXIDE'
SMIX_ID='CARBON DIOXIDE','OXYGEN','CARBON MONOXIDE'
NU=-1,0.5,1
HEAT_OF_COMBUSTION=-6430.
ODE_SOLVER='RK2 RICHARDSON' /

```

¹<http://kinetics.nist.gov/janaf/>

12.3 Finite Rate Combustion

Usually, FDS uses a mixing-controlled combustion model. However, FDS can also employ finite-rate reactions using an Arrhenius model. This section describes how to do this.

1. It is strongly recommended that finite-rate reactions be invoked only when FDS is running in DNS mode. Set `DNS=.TRUE.` on the `MISC` line. You may use the finite-rate reaction model in an LES calculation, but because the temperature in a large scale calculation is smeared out over a mesh cell, some of the reaction parameters may need to be modified to account for the lower temperatures.
2. The background species, if unspecified on a `SPEC` or `SMIX` line, is normally set to be 'AIR'.
3. Read Sections 11.1 and 11.2 for a description of the boundary conditions for the gas species.
4. The `REAC` namelist group is used to designate the fuel and the reaction rate parameters. For a finite-rate reaction you can specify multiple `REAC` lines. Note that FDS will evaluate the reactions simultaneously.

The inputs to use the finite-rate model are given below:

`FUEL` Character string indicating which of the listed gas species is the fuel.

`A` Pre-exponential factor in units of $\text{cm}^3/\text{mole}/\text{s}$.

`E` Activation energy in units of kJ/kmol .

`EQUATION` (Optional) Character string defining the reaction stoichiometry. Can be used instead of specifying `SMIX_ID` and `NU`. See 12.5.1.

`NU` Array containing the stoichiometric coefficients where negative values indicate reactants and positive values indicate products.

`N_S` Array containing the exponents for the finite rate equation for each species.

`N_T` The temperature exponent for the rate (Default is 0 or no temperature dependence).

`SPEC_ID` Array containing the names of the species in the rate equation for the reaction. Note that it is possible that a given `SPEC` can contribute to the rate but not be a reactant or product. In other words, the rate equation can be dependent on a species that does not participate directly in the reaction. These entries correspond to the inputs on `N_S`.

`HEAT_OF_COMBUSTION` The effective heat of combustion of the `FUEL` in chemical reaction in units of kJ/kg . (No default)

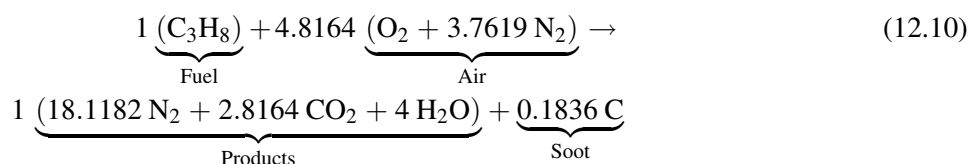
12.3.1 Special Topic: Mixed-Mode Combustion

In a typical simulation, flame temperatures are not likely to be resolved well enough to allow for a finite rate reaction to be computed. However, the temperatures in a layer environment are likely to support a finite rate reaction for slow reactions (*e.g.* carbon monoxide oxidation in a hot layer). A mixed-mode combustion model can be used for this situation. To enable this model, specify the finite-rate reaction inputs described in the previous section as well as the input parameter `THRESHOLD_TEMPERATURE`. During a simulation, if there is heat release from other reactions, then a mixed-mode reaction will be evaluated as a mixing-controlled reaction. If there is no heat release from other reactions and the local temperature exceeds the `THRESHOLD_TEMPERATURE`, then the mixed-mode reaction will be evaluated as a finite-rate reaction.

12.4 Aerosol Deposition

By default FDS will compute all of the soot deposition phenomena discussed in the Technical Reference Guide, but they can be selectively disabled with the logical flags `GRAVITATIONAL_DEPOSITION`, `THERMOPHORETIC_DEPOSITION`, and `TURBULENT_DEPOSITION`. The deposition model is invoked by defining a species with the parameter `AEROSOL=.TRUE.` (Note; this model should be considered an experimental feature.). The properties of the aerosol can be defined on the species input using `DENSITY_SOLID`, `CONDUCTIVITY_SOLID`, and `MEAN_DIAMETER`.

This example shows how to define a reaction that invokes an aerosol deposition model in FDS (e.g. soot deposition). For this example propane will be the fuel with a 5 % soot yield. The stoichiometric equation is shown below.



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 transport by itself will have an `SMIX_COMPONENT_ONLY = .TRUE.` designation. Note that for soot to deposit it must be tracked separately and it must be given `AEROSOL = .TRUE.`.

```

&SPEC ID = 'PROPANE' /
&SPEC ID = 'OXYGEN',          SMIX_COMPONENT_ONLY = .TRUE. /
&SPEC ID = 'NITROGEN',        SMIX_COMPONENT_ONLY = .TRUE. /
&SPEC ID = 'WATER VAPOR',     SMIX_COMPONENT_ONLY = .TRUE. /
&SPEC ID = 'CARBON DIOXIDE',   SMIX_COMPONENT_ONLY = .TRUE. /
&SPEC ID = 'SOOT',            AEROSOL = .TRUE. /

```

For the oxidizer and products, which are both composed of multiple primitive species, `SMIX` lines are needed to define the composition of the lumped species. As the user, you can define the `SMIX` by either the `MASS_FRACTIONS` of the component gases or the `VOLUME_FRACTIONS`. For this case, volume fractions will be used. If Equation 12.10 is properly balanced, you can directly use the stoichiometric coefficients of the primitive species to define the lumped species.

```

&SMIX ID = 'AIR', SPEC_ID = 'NITROGEN', 'OXYGEN',
    VOLUME_FRACTION = 3.7619, 1., BACKGROUND = .TRUE. /
&SMIX ID = 'PRODUCTS', SPEC_ID = 'NITROGEN', 'CARBON DIOXIDE', 'WATER VAPOR',
    VOLUME_FRACTION = 18.1182, 2.8164, 4. /

```

Note: the order of which the `SPEC_IDS` are defined must be the same order as which you defined your volume fractions or mass fractions.

In this mixing-controlled example, the user is responsible for defining the fuel, heat of combustion of that fuel, species participating the reaction, and the stoichiometric coefficients that describe the reaction. The heat of combustion for `PROPANE` was found from the SFPE Handbook [20]. In the lumped species approach, the stoichiometric coefficients for the `SMIX` lines are the coefficients of the lumped species themselves.

```

&REAC FUEL = 'PROPANE', HEAT_OF_COMBUSTION=44715.,
    SMIX_ID = 'PROPANE', 'AIR', 'PRODUCTS', 'SOOT',
    NU=-1., -4.8164, 1, 0.1836/

```

Note: The sign of `NU` corresponds to whether that species is consumed (-) or produced (+). Figure 12.3 shows boundary file output from Smokeview of soot surface deposition on the wall. This boundary quantity is given by the input below.

```
&BNDF QUANTITY='SURFACE DEPOSITION', SPEC_ID='SOOT' /
```

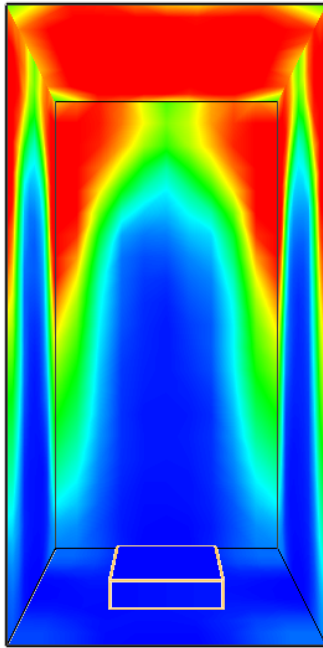


Figure 12.3: Wall soot deposition for propane flame with deposition example.

12.5 Other Reaction Parameters

12.5.1 Special Topic: Using the EQUATION input parameter

The input parameter EQUATION allow the specification of the chemical reaction by specifying a text equation. The rules for the use of this input are:

- Species can be given as either their chemical formula (which must match the formula defined on SPEC or by the ID of the species. Only tracked species can be listed.
- The stoichiometry is given before each species and is separated by a *. Real numbers are allowed but not exponential notation (i.e. 201.1 but not 2.011E2).
- The reactants and products are separated by an =.

For example if the reaction defines the complete combustion of methane using primitive species the following would be equivalent:

```
&REAC FUEL='METHANE', EQUATION = 'METHANE+2*OXYGEN=CARBON DIOXIDE+2*WATER VAPOR',  
      HEAT_OF_COMBUSTION=50000. /  
  
&REAC FUEL='METHANE', EQUATION = 'CH4+2*O2=CO2+2*H2O',  
      HEAT_OF_COMBUSTION=50000. /  
  
&REAC FUEL='METHANE', EQUATION = 'METHANE+2*O2=CO2+2*H2O',  
      HEAT_OF_COMBUSTION=50000. /
```

12.5.2 Special Topic: Reaction Rate Limiters

At certain critical points in a calculation, like the moment of ignition, the local reaction rate can be very large due to limitations in the models, long time steps, or both. To prevent fictitiously high values of the reaction rate that can lead to numerical instabilities, a limiting value of the heat release rate per unit volume is used:

$$\dot{q}_{\max}''' = \frac{\text{HRRPUA_SHEET}}{\delta x} + \text{HRRPUV_AVERAGE} \quad (12.11)$$

where HRRPUA_SHEET is given in units of kW/m² and HRRPUV_AVERAGE is given in units of kW/m³.

12.5.3 Special Topic: Diagnostic Parameters

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 .FALSE. will bypass this check. (Default .TRUE.)

REAC_ATOM_ERROR Error tolerance in units of atoms for the reaction stoichiometry check. (Default 1.E-5)

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)

Chapter 13

Radiation

For most FDS simulations, thermal radiation transport is computed by default and you need not set any parameters to make this happen. However, there are situations where it is important to be aware of issues related to the radiative transport solver.

13.1 Basic Radiation Parameters: The `RADI` Namelist Group

`RADI` is the namelist group that contains all of the parameters related to the radiation solver. There can be only one `RADI` line in the input file. It is possible to turn off the radiation transport solver (saving roughly 20 % in CPU time) by adding the statement `RADIATION=.FALSE.` 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 also input on the `RADI` line. This radiated energy completely disappears from the calculation. For fire scenarios 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.

13.1.1 Radiative Fraction

The most important radiation parameter is the fraction of energy released from the fire as thermal radiation, commonly referred to as the *radiative fraction*. It is a function of both the flame temperature and chemical composition, neither of which are reliably calculated in a large scale fire calculation because the flame sheet is not well-resolved. In calculations in which the mesh cells are on the order of a centimeter or larger, the temperature near the flame surface cannot be relied upon when computing the source term in the radiation transport equation, especially because of the T^4 dependence. As a practical alternative, the parameter `RADIATIVE_FRACTION` on the `RADI` line allows you to specify explicitly the fraction of the total combustion energy that is released in the form of thermal radiation. Some of that energy may be reabsorbed elsewhere, yielding a net radiative loss from the fire or compartment that is less than the `RADIATIVE_FRACTION`, depending mainly on the size of the fire and the soot loading. If it is desired to use the radiation transport equation as is, then `RADIATIVE_FRACTION` ought to be set to zero, and the source term in the radiative transport equation is then based solely on the gas temperature and the chemical composition. By default, the `RADIATIVE_FRACTION` is 0.35 for an LES calculation, and zero for DNS.

13.1.2 Spatial and Temporal Resolution of the Radiation Transport Equation

There are several ways to improve the spatial and temporal accuracy of the Finite Volume Method in solving the radiation transport equation (RTE), but these will increase the computation time. You can increase the number of angles from the default 100 with the integer parameter `NUMBER_RADIATION_ANGLES`. 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.

13.2 Radiative Absorption

By default FDS employs a gray gas model for the radiation absorption coefficient, a function of gas composition and temperature, which are tabulated in a look-up table using the routines found in RadCal.

13.2.1 RadCal Issues

There are several issues to keep in mind with regard to RadCal.

Path Length

Because RadCal computes effective absorption coefficients over a range of wavelengths, it requires a user-specified `PATH_LENGTH` (m). Its default value is five times the width of a single grid cell.

Fuel Species

The original version of RadCal only included absorption data for methane as a surrogate for any fuel. Thus, the default value of `RADCAL_FUEL` is 'METHANE'. Work is currently underway to extend the list of fuels.

13.2.2 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 of the water and fuel particles (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 tabulated in a `TABLE` and called using the `RADIATIVE_PROPERTY_TABLE`. More details can be found in Section 14.3.2.

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.

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.

13.2.3 Wide Band Model

The radiation solver has two modes of operation – a gray gas model (default) and a wide band model [1]. If the optional six band model is desired, set `WIDE_BAND_MODEL=.TRUE.` 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. To add three additional fuel bands, set `CH4_BANDS=.TRUE.` on the `RADI` line. Read the FDS Technical Reference Guide [15] for more details. Note also that when `WIDE_BAND_MODEL=.TRUE.`, the `ABSORPTION_COEFFICIENT` output quantity becomes practically useless, because it then corresponds to one individual band of the spectrum.

Chapter 14

Particles and Droplets

Lagrangian particles are used in FDS to represent water or liquid fuel droplets, flow tracers, and various other objects that are not defined or confined by the numerical mesh. Sometimes the particles have mass, sometimes they do not. Some evaporate, absorb radiation, *etc.* `PART` is the namelist group that is used to prescribe parameters associated with Lagrangian particles.

14.1 Basics

Properties of different types of Lagrangian particles are designated via the `PART` namelist group. Once a particular type of particle or droplet has been described using a `PART` line, then the name of that particle or droplet 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',... /
```

These Lagrangian particles can be introduced at a solid surface via the `SURF` line that defines the properties of the material, for example

```
&SURF ...,PART_ID='my smoke' /
```

or the `PART_ID` can be invoked from a `PROP` line to change the properties of the droplets ejected by a sprinkler or nozzle, for example

```
&PROP ID='Acme 123', QUANTITY='SPRINKLER LINK TEMPERATURE', PART_ID='my water', ... /
```

14.1.1 Output Quantities

The parameter `QUANTITIES` is an array of character strings indicating which scalar quantities should be used to color the particles or droplets when viewed as an animation. The choices are

```
'PARTICLE TEMPERATURE' (°C)  
'PARTICLE DIAMETER' (μm)  
'PARTICLE VELOCITY' (m/s)  
'PARTICLE MASS' (kg)  
'PARTICLE AGE' (s)
```

As a default, if no `QUANTITIES` are specified and none are selected in Smokeview, then Smokeview will display particles with a single color. To select this color specify either `RGB` or `COLOR`. By default, water droplets are colored blue and fuel droplets yellow. All others are colored black.

14.1.2 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=.TRUE., ... /
```

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=.TRUE.`, the `SAMPLING_FACTOR` is set to 1 unless you say otherwise, which would be pointless since `MASSLESS` particles are for visualization only.

14.2 Particle and Droplet Insertion

There are three ways of introducing droplets or particles into a simulation. The first way is to define a sprinkler or nozzle using a `PROP` line that includes a `PART_ID` that specifies the droplet parameters. The second way is to add a `PART_ID` to a `SURF` line, in which case particles or droplets will be ejected from that surface. Note that this only works if the surface has a normal velocity pointing into the flow domain. The third way to introduce particles or droplets is via 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.

14.2.1 Particles Introduced at a Solid Surface

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` 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 kg/m²/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. 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, but must be specified if it is not. There is a simple test case 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 m × 0.4 m × 4. The expected accumulated mass of particles on the ground after 10 s is expected to be 0.64 kg, as shown in Fig. 14.1.

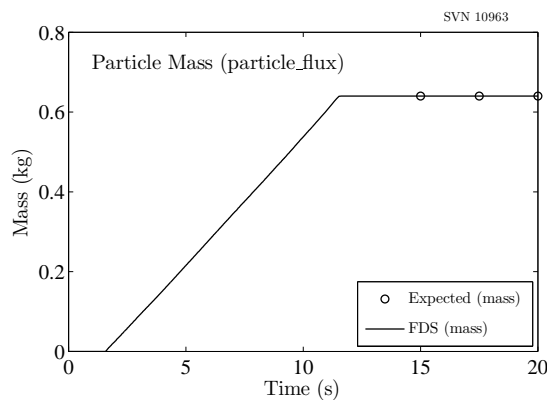


Figure 14.1: Simple test case to demonstrate mass conservation of particles ejected from an obstruction.

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 15.4 and 15.5.

14.2.2 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.

Note that `PARTICLES_PER_SECOND` can be a very important parameter. In some simulations, it is a good idea to increase this number so that the liquid mass is distributed more uniformly over the droplets. 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.

14.2.3 Particles or Droplets Introduced within a Volume

Sometimes it is convenient to introduce droplets or particles at the start of the simulation. For this purpose, use an `INIT` line with the parameter `N_PARTICLES` to indicate the number of particles within the computational domain at the start of the simulation. You may also use `N_PARTICLES_PER_CELL` to indicate the number of particles within each grid cell of the computational domain at the start of the simulation. Its default value is 0, meaning that initially there are no particles present. If non-zero, you may also specify `MASS_PER_VOLUME` (kg/m^3) if the particles have mass. 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 . Also, to limit the particles to a certain region of the domain, add the real sextuplet `XB` to designate the coordinates of a rectangular volume. The format for `XB` is the same as that used on the `OBST` line. You can only introduce a cloud of particles within a rectangular block, and the volume of the block is calculated according to the dimensions `XB`, regardless of whether there are solid obstructions within this volume. 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`. When using `N_PARTICLES_PER_CELL`, the particles will be randomly placed within the volume of each cell. If `CELL_CENTERED=.TRUE.` is set, then the particles will be placed at the center of each cell.

```
&PART ID='droplets', DIAMETER=750., SPEC_ID='WATER VAPOR' /
&PART ID='target', SURF_ID='...' /
&INIT PART_ID='droplets', XB=..., N_PARTICLES=1000, MASS_PER_VOLUME=3.5 /
&INIT PART_ID='target', XYZ=..., N_PARTICLES=1 /
```

If you use `XYZ` to position the particle at a specific point, 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 Section 15.10.5 for more details.

If you want to introduce particles within a given volume 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.

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 Section 15.4 for more information on controlling devices.

14.2.4 Controlling the Number of Particles and Droplets

Regardless of how the particles or droplets are introduced into the computational domain, the following are important parameters for controlling their number:

`DT_INSERT` Time increment in seconds between the introduction of a “batch” of particles or droplets from `SURF` or `INIT`. The number per “batch” depends on how they are introduced. The default value is 0.01 s. Note that this parameter should be specified on the `SURF` or `INIT` line, depending on whether the particles originate at a surface or a volume. Note that if you introduce particles via an `INIT` line, you must specify a value for `DT_INSERT`. Otherwise, particles or droplets will only be introduced initially.

`SAMPLING_FACTOR` Sampling factor for the output file `CHID.prt5`. This parameter 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 Section 14.1.2.

`AGE` 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.

14.3 Liquid Droplets

To define an evaporating liquid droplet, you must explicitly specify the gaseous species via the `SPEC` namelist group (see Section 11.1), and then designate the appropriate `SPEC_ID` on the `PART` line. If the droplets are defined with `SPEC_ID='WATER VAPOR'`, then the particles will be assigned the thermo-physical properties of water, the radiation absorption properties of water, and will be colored blue in Smoke-view.

14.3.1 Droplet Thermal Properties

The following parameters should be included on the `SPEC` line to control the evaporation. If the liquid properties of the `SPEC_ID` are included in Table 11.1, then no further inputs are required. Otherwise, you must provide all of the following properties of the liquid:

`DENSITY_LIQUID` The density of the liquid or solid droplet/particle (kg/m^3).

`SPECIFIC_HEAT_LIQUID` Specific heat of liquid or solid droplet/particle (kJ/kg/K).

`VAPORIZATION_TEMPERATURE` Boiling temperature of liquid droplet ($^{\circ}\text{C}$).

`MELTING_TEMPERATURE` Melting (solidification) temperature of liquid droplet ($^{\circ}\text{C}$).

`INITIAL_TEMPERATURE` Initial temperature of liquid droplet; assumed ambient, `TMPA` ($^{\circ}\text{C}$).

`HEAT_OF_VAPORIZATION` Latent heat of vaporization of liquid droplet (kJ/kg).

`H_V_REFERENCE_TEMPERATURE` The temperature corresponding to the provided `HEAT_OF_VAPORIZATION` ($^{\circ}\text{C}$).

14.3.2 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` and specifying the `ID` of that table is `RADIATIVE_PROPERTY_TABLE` property on the `PART` line. Each row of a spectral property table contains three real numbers: wavelength (μm), 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 10.0, the particles are treated as perfectly reflecting spheres. The complex part should be a non-negative number. Values less than 10^{-6} are treated as non-absorbing. Below is an example of the use of spectral property table, listing the properties at wavelengths 1, 5 and $10\mu\text{m}$.

```
&PART ID='particles',..., RADIATIVE_PROPERTY_TABLE='table' /
&TABL ID='table', TABLE_DATA= 1.0,1.33,0.0001 /
&TABL ID='table', TABLE_DATA= 5.0,1.33,0.002 /
&TABL ID='table', 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 default averaging factor, `RUN_AVG_FAC`, is set to 0.5.

14.3.3 Size Distribution

For liquid droplets, the specified `DIAMETER` (in units of μm) on the `PART` line is the median volumetric diameter of the droplets or particles, with the distribution assumed to be a combination of Rosin-Rammler and log-normal (Default 500 μm). The width of the distribution is controlled by the parameter `GAMMA_D` (default 2.4) The Rosin-Rammler/log-normal distribution is given by

$$F(d) = \begin{cases} \frac{1}{\sqrt{2\pi}} \int_0^d \frac{1}{\sigma d'} e^{-\frac{[\ln(d'/d_m)]^2}{2\sigma^2}} dd' & (d \leq d_m) \\ 1 - e^{-0.693(\frac{d}{d_m})^\gamma} & (d_m < d) \end{cases} \quad (14.1)$$

Note that the parameter σ is given the value $\sigma = 2/(\sqrt{2\pi}(\ln 2)^\gamma) = 1.15/\gamma$ which ensures that the two functions are smoothly joined at $d = d_m$. 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. Note that you can prevent droplets or particles from exceeding `MAXIMUM_DIAMETER`, which is infinitely large by default. Also note that droplets less than `MINIMUM_DIAMETER` are assumed to evaporate in a single time step. The default `MINIMUM_DIAMETER` is set to 0.005 the value of `DIAMETER`. The droplet distribution is divided into a series of bins for picking particle size. To avoid very small particle weights, the distribution is clipped at the cumulative fractions of `CDF_CUTOFF` and $(1 - \text{CDF_CUTOFF})$. To prevent FDS from generating a distribution of droplets/particles altogether, set `MONODISPERSE=.TRUE.` on the `PART` line, in which case every droplet or particle will be assigned the same `DIAMETER`.

If you set `CHECK_DISTRIBUTION=.TRUE.` 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`.

By default, the range of particle sizes is divided into seven bins, and the sampled particles are divided among these bins. This ensures that a reasonable number of particles is assigned to the entire spectrum of sizes. To change the default number of bins, set `N_STRATA` on the `PART` line.

14.3.4 Secondary breakup

If `BREAKUP=.TRUE.` 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 Sauter mean diameters, `BREAKUP_RATIO`. Its default is 3/7. Optionally, specify the distribution parameters `BREAKUP_GAMMA_D` and `BREAKUP_SIGMA_D`.

14.3.5 Fuel Droplets

If the droplets evaporate into the `FUEL` identified on the `REAC` line, they 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`, also specified on the `PART` line. Note that the burning rate will be adjusted to account for the difference between the heats of combustion of the droplets and the other fuels in the model.

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 present in the liquid properties table as a fuel, then the droplets will be given fuel radiation absorption properties.

Example Case: spray_burner

Controlled fire experiments are often conducted using a spray burner, where a liquid fuel is sprayed out of a nozzle and ignited. In this example (`spray_burner.fds`), heptane from two nozzles is sprayed

downwards 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',C=7,H=16,SOOT_YIELD=0.01,HEAT_OF_COMBUSTION=44500./

&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., HEAT_OF_COMBUSTION=44500., SAMPLING_FACTOR=1 /

&PROP ID='nozzle', CLASS='NOZZLE', PART_ID='heptane droplets',
      FLOW_RATE=1.96, FLOW_RAMP='fuel', PARTICLE_VELOCITY=10.,
      SPRAY_ANGLE=0.,30. /
&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:

$$2 \times 1.96 \frac{\text{L}}{\text{min}} \times \frac{1}{60} \frac{\text{min}}{\text{s}} \times 688 \frac{\text{kg}}{\text{m}^3} \times \frac{1}{1000} \frac{\text{m}^3}{\text{L}} \times 44500 \frac{\text{kJ}}{\text{kg}} = 2000 \text{ kW} \quad (14.2)$$

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 the user expects. 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. 14.2.

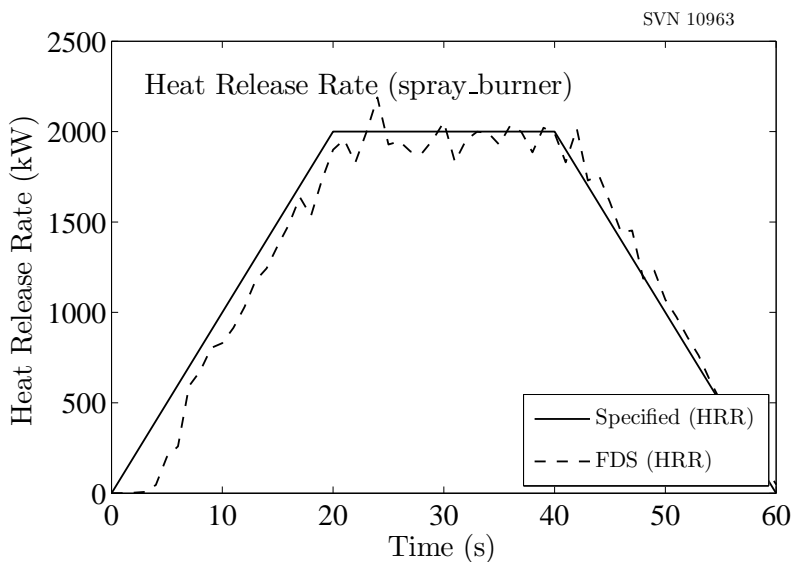


Figure 14.2: Heat Release Rate (HRR) of spray burner test.

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.

14.4 Solid Particles

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. If you simply want massless tracers, specify `MASSLESS=.TRUE.` on the `PART` line. If you specify a `SPEC_ID`, then FDS automatically assumes that you want relatively small, thermally-thin evaporating liquid droplets. However, it is possible to assign arbitrary surface properties to particles that represent subgrid-scale objects, like office clutter or vegetation. To do this, you add a `SURF_ID` to the `PART` line.

14.4.1 Basics

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=.TRUE., 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 a class of particles with the properties given by the `SURF` line 'HOT'. `STATIC` is a logical parameter indicating whether particles move or just serve as obstructions or clutter. The default value of `STATIC` is `.FALSE.` The `PROP` line is used just to tell Smokeview that the particles are to be drawn as the appropriate sized spheres. The `SURF` line establishes that the particles are hot spheres with a radius of 0.005 m and constant temperature of 500 °C. The `GEOMETRY` options are 'SPHERICAL', 'CYLINDRICAL', or 'CARTESIAN'. If 'SPHERICAL', provide the `RADIUS`. If 'CYLINDRICAL', provide the `RADIUS` and the `LENGTH`. If 'CARTESIAN', provide the `LENGTH`, `WIDTH`, and `THICKNESS` of the plate. Note that the heat conduction calculation will be done in the appropriate coordinate system. The `INIT` line randomly fills the given volume with 10 of these hot spheres.

The output quantities associated with complex particles are the same as those for solid walls. For example, to record the temperature of a given particle, use lines like the following:

```
&INIT ID='my particle', PART_ID='...', XB=..., N_PARTICLES=1 /
&DEVIC ID='...', INIT_ID='my particle', QUANTITY='WALL TEMPERATURE' /
```

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. However, you can use the parameter `ORIENTATION(1:N,1:3)` to specify `N` unique directions for the particle. For example,

```
&PART ..., ORIENTATION(1,3)=1., ORIENTATION(2,3)=-1. /
```

specifies that half the particle is facing upwards and half is facing downwards. This is useful if the source of heating does not surround the particles.

14.4.2 Drag

For massive particles the default drag law (i.e., the drag coefficient correlation as a function of the Reynolds number based on particle diameter) is that of a sphere. To invoke the cylinder drag law set `DRAW_LAW` to

'CYLINDER' on the PART line and to invoke the screen drag law (see Section 14.4.5) set to 'SCREEN' If neither of these options is applicable, the user may specify a constant value of the drag coefficient for a particle class (a specific PART_ID) by setting a USER_DRAG_COEFFICIENT on PART. The USER_DRAG_COEFFICIENT over-rides the DRAG_LAW. The USER_DRAG_COEFFICIENT is a triplet indicating the drag along each coordinate axis. If only the first value of the triplet is assigned, the drag is assumed to be isotropic.

14.4.3 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=.TRUE. /
&PART ID='tube', SAMPLING_FACTOR=1, SURF_ID='tube bc', STATIC=.TRUE. /
&PART ID='ball', SAMPLING_FACTOR=1, SURF_ID='ball bc', STATIC=.TRUE. /

&SURF ID='plate bc', 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 /
```

Notice that the SURF lines contain the necessary parameters to describe a plate, cylinder and sphere, but they do not contain the parameter THICKNESS. The reason is that the particles have no mass. Only the surface area is important. For the plate, the surface area is the length times the width. For the cylinder, the area is twice the radius times the length. For the sphere, the area is 4π times the radius squared. Figure 14.3 displays the output of the test case called surf_mass_part_specified.fds, demonstrating that the production rate of the gases is as expected.

14.4.4 Vegetation

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. Suppose we want to describe a collection of wet pine needles that occupy a certain volume. The following lines have been extracted from the sample file WUI/pine_needles.fds. Note that all of the values have been chosen simply to demonstrate the technique. These values should not be used for a real calculation.

```
&PART ID='pine needles', SAMPLING_FACTOR=1, SURF_ID='wet vegetation',
      PROP_ID='needle image', STATIC=.TRUE. /
&INIT PART_ID='pine needles', XB=0.,1.,0.,1.,0.,1., N_PARTICLES=1000, MASS_PER_VOLUME=1. /
&PROP ID='needle image', SMOKEVIEW_ID='TUBE', SMOKEVIEW_PARAMETERS='L=0.1','D=0.0005' /

&SURF ID = 'wet vegetation'
```

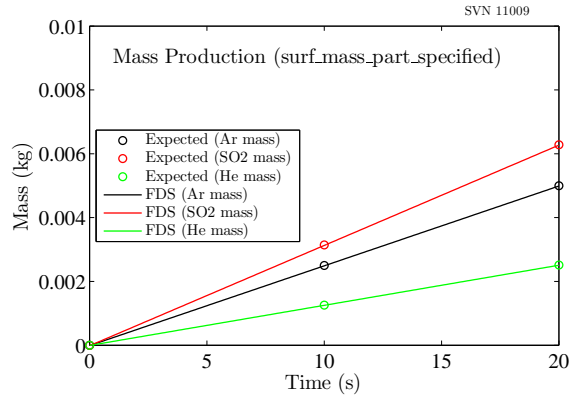


Figure 14.3: Gas production from three Lagrangian particles.

```

MATL_ID(1,1:2) = 'PINE','MOISTURE'
MATL_MASS_FRACTION(1,1:2) = 0.8,0.2
THICKNESS = 0.00025
LENGTH = 0.1
GEOMETRY = 'CYLINDRICAL' /

&MATL ID = 'PINE'
  DENSITY = 500.
  CONDUCTIVITY = 0.1
  SPECIFIC_HEAT = 1.0
  N_REACTIONS = 1
  REFERENCE_TEMPERATURE = 300.
  NU_MATL = 0.2
  NU_SPEC = 0.8
  SPEC_ID = 'GLUCOSE'
  HEAT_OF_REACTION = 1000
  MATL_ID = 'CHAR' /

&MATL ID = 'MOISTURE'
  DENSITY = 1000.
  CONDUCTIVITY = 0.1
  SPECIFIC_HEAT = 4.184
  N_REACTIONS = 1
  REFERENCE_TEMPERATURE = 100.
  NU_SPEC = 1.0
  SPEC_ID = 'WATER VAPOR'
  HEAT_OF_REACTION = 2500. /

&MATL ID = 'CHAR'
  DENSITY = 200.
  CONDUCTIVITY = 1.0
  SPECIFIC_HEAT = 1.6 /

```

In the example, 1 kg of pine needles occupy 1 m³. 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 needles are modeled as cylinders that are 0.5 mm in diameter. The `THICKNESS` on the `SURF` line refers to the radius of the cylinder in units of m. The needles are all 0.1 m long. The needles contain 20 % (by mass) moisture, and 80 % cellulose. The moisture is set to evaporate at 100 °C to create water

vapor and the cellulose pyrolyses at 300 °C to form fuel gas and char. In the example case, the original 1 kg of 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 14.4 shows the evolution of the fuel, water and char mass. Agreement with the expected values means that mass is conserved.

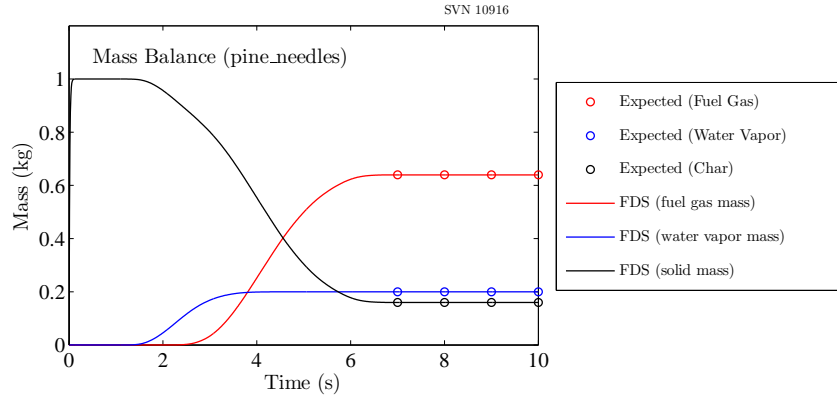


Figure 14.4: Evolution of vegetation mass in the `pine_needles` test case.

14.4.5 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=.TRUE., 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=.TRUE.,
      SURF_ID='SCREEN', ORIENTATION=1,0,0 /
&SURF ID='SCREEN', THICKNESS=0.00015, GEOMETRY='CYLINDRICAL',
      LENGTH=5., CONVECTION_LENGTH_SCALE=0.0003, MATL_ID='ALUMINUM' /
&MATL ID='ALUMINUM', DENSITY=2700., CONDUCTIVITY=200., SPECIFIC_HEAT=900. /
```

The `INIT` line designates the plane of the screen using the sextuplet `XB`. A single particle is located in each cell with the parameter `N_PARTICLES_PER_CELL=1`. The particles should be defined with a `SURF_ID` containing the material properties of the screen. A special drag law for screens is specified via `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` on the `SURF` line, when multiplied by the diameter, results in 60 % of the cell area. The pressure drop across the screen is given by

$$\Delta P = \Delta x \left(\frac{\mu}{K} u + \rho \frac{Y}{\sqrt{K}} u^2 \right) \quad (14.3)$$

where Δx is the screen thickness and Y and K are empirical constants given by

$$K = 3.44 \times 10^{-9} \text{ FREE_AREA_FRACTION}^{1.6} \quad (14.4)$$

$$Y = 0.043 \text{ FREE_AREA_FRACTION}^{2.13} \quad (14.5)$$

14.5 Special Topic: 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.

14.5.1 Velocity on Solid Surfaces

When a droplet strikes a solid surface, 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 downwards. The rate at which the droplets move over the horizontal and vertical surfaces is difficult to quantify. The parameters `HORIZONTAL_VELOCITY` and `VERTICAL_VELOCITY` on the `PART` line allow you to control the rate at which droplets move horizontally or vertically (downward). The defaults are 0.2 m/s and 0.5 m/s, respectively.

There are some applications, like the suppression of racked storage commodities, 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 = .TRUE.` on the `MISC` line. It is normally false.

Be aware 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`.

A useful sample case to demonstrate various features of droplet motion on solid obstructions is the test case called `cascade.fds`. Figure 14.5 shows a stream of water droplets impinging on the top of a box followed by the cascading of water droplets over the top edge.

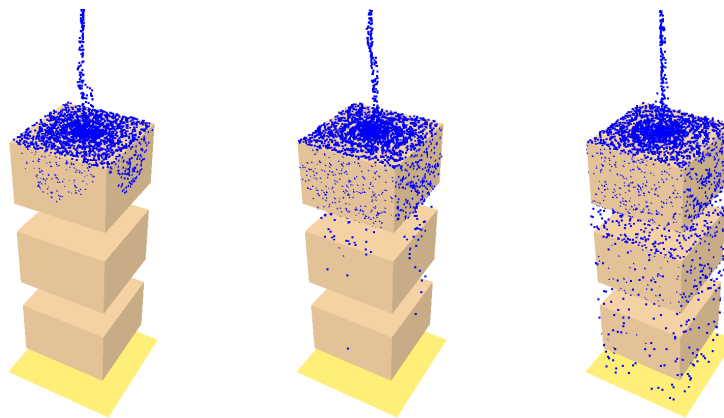


Figure 14.5: Smokeview rendering of the **cascade** test case.

If you do not want droplets to accumulate on solid surfaces, set `ALLOW_UNDERSIDE_PARTICLES = .FALSE.` on the `MISC` line. It is normally true.

14.5.2 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

$$\dot{m}_f''(t) = \dot{m}_{f,0}''(t) e^{-\int k(t) dt} \quad (14.6)$$

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/m².

$$k(t) = \text{E_COEFFICIENT } m_w''(t) \quad \text{s}^{-1} \quad (14.7)$$

The parameter `E_COEFFICIENT` must be obtained experimentally, and it is expressed in units of m²/kg/s. Usually, this type of suppression algorithm is invoked when the fuel is complicated, like a cartoned commodity.

Chapter 15

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.

Versions of FDS prior to FDS 5 used device-specific namelist groups, like `SPRK`, `HEAT`, `SMOD`, and `THCP`, but the number and variety of fire-specific sensing and measurement devices continues to expand, and the data structures in FDS could not easily accommodate all possibilities. In addition, the logic associated with sensor activation and subsequent actions, like a vent opening, had become too complicated and prone to bugs. Starting in FDS 5, 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 namelists groups called `CTRL` (Control) and `PROP` (Properties).

15.1 Device Location and Orientation: The `DEVC` Namelist Group (Table 16.5)

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`. 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_devc.csv**.

Some devices have a particular orientation which can be specified with various parameters; `IOR`, `ORIENTATION`, `ROTATION`. `IOR` or the Index of Orientation, is necessary 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”, where 1 is parallel to the x -axis, 2 is parallel to the y -axis and 3 is parallel to the z -axis.

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=1,0,0 /
```

the sprinkler would point in the positive x direction. 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.

15.2 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 Section 19.3 for output file format). This file can be imported into most spread sheet 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 csv file. If your spreadsheet applicate allows fewer columns than the number of DEVC or CTRL in your input file then specify COLUMN_DUMP_LIMIT=.TRUE. 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.

By default, the DEVC output is written to a file every DT_DEVC seconds. This time increment is specified on the DUMP line. Also, by default, a time-averaged value is written out for each quantity of interest. To prevent FDS from time-averaging the DEVC output, add TIME_AVERAGED=.FALSE. to the DEVC line.

A useful option for the DEVC line is to add RELATIVE=.TRUE., 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 change the values of the output by multiplying by CONVERSION_FACTOR and changing the character string UNITS.

If you do not want the DEVC QUANTITY to be included in the output file, set OUTPUT=.FALSE. 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 .FALSE. automatically.

All devices must have a specified QUANTITY. Some special devices (Section 15.3) 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.

15.3 Special Device Properties: The PROP Namelist Group (Table 16.20)

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 ID might be the manufacturer's name, like 'ACME Sprinkler 123', for example.

The best way to describe the PROP group is to list the various special devices and their properties.

15.3.1 Sprinklers

Here is a very simple example of sprinkler:

```
&PROP ID='K-11', QUANTITY='SPRINKLER LINK TEMPERATURE', RTI=148., C_FACTOR=0.7,  
      ACTIVATION_TEMPERATURE=74., OFFSET=0.10,PART_ID='water drops', FLOW_RATE=189.3,  
      PARTICLE_VELOCITY=10., SPRAY_ANGLE=30.,80. /  
  
&DEVC ID='Spr_60', XYZ=22.88,19.76,7.46, PROP_ID='K-11' /  
&DEVC ID='Spr_61', XYZ=22.88,21.76,7.46, PROP_ID='K-11' /
```

A sprinkler, known as 'Spr_60', 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 as meaningful to you as possible. 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 algorithm. The various sprinkler properties will be discussed below.¹

Properties associated with sprinklers included in the PROP group are:

RTI Response Time Index in units of $\sqrt{\text{m} \cdot \text{s}}$. (Default 100.)

C_FACTOR in units of $\sqrt{\text{m/s}}$. (Default 0.)

ACTIVATION_TEMPERATURE in units of °C. (Default 74 °C)

INITIAL_TEMPERATURE of the link in units of °C. (Default TMPA)

FLOW_RATE **or** MASS_FLOW_RATE in units of L/min or kg/s. An alternative is to provide the K_FACTOR in units of $\text{L/min/bar}^{\frac{1}{2}}$ and the OPERATING_PRESSURE in units of bar. The flow rate is then given by $\dot{m}_w = K\sqrt{p}$. Note that 1 bar is equivalent to 14.5 psi, 1 gpm is equivalent to 3.785 L/min, 1 $\text{gpm/psi}^{\frac{1}{2}}$ is equivalent to 14.41 $\text{L/min/bar}^{\frac{1}{2}}$. If MASS_FLOW_RATE is given then PARTICLE_VELOCITY must also be defined.

OFFSET Radius of a sphere (m) surrounding the sprinkler where the water droplets are initially placed in the simulation. It is assumed that at and beyond the OFFSET the droplets have completely broken up and are transported independently of each other. (Default 0.05 m)

PARTICLE_VELOCITY Initial droplet velocity. (Default 0 m/s)

¹Prior to FDS 5, a separate file was used to store properties of a given sprinkler. This file is no longer used.

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 the user must fine-tune **PARTICLE_VELOCITY** in order to reproduce certain spray profile. The **ORIFICE_DIAMETER** input 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 droplets to leave the sprinkler through a band between 60° and 10° south latitude, assuming the orientation of the sprinkler is (0,0,-1), the default.

SPRAY_PATTERN_SHAPE determines how the droplets are distributed within this band. Choices are 'UNIFORM' for uniform distribution and 'GAUSSIAN'. Parameter **SPRAY_PATTERN_MU** controls the latitude of maximum density of droplets for the 'GAUSSIAN' distribution. Width of the distribution is controlled by 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 14 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.

Be aware that sprinklers produce many droplets that need to be tracked in the calculation. To limit the computational cost, sprinkler droplets disappear when they hit the lower boundary of the computational domain, regardless of whether it is solid or not. To stop FDS from removing sprinkler droplets from the lower boundary of the computational domain, add the phrase **POROUS_FLOOR=.FALSE.** to the **MISC** (Section 6.4) 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.

For more information about sprinklers, their activation and spray dynamics, read the FDS Technical Reference Guide [1].

Special Topic: Specifying Complex Spray Patterns

If a more complex spray pattern is desired than can be achieved by using **SPRAY_ANGLE**, **PARTICLE_VELOCITY**, and **FLOW_RATE**, then a **SPRAY_PATTERN_TABLE** can be specified using the **TABL** (Section 10.3) namelist group. For a spray pattern, specify the total flow using **FLOW_RATE** of the **PROP** line, the name of the spray pattern using **SPRAY_PATTERN_TABLE** and then one or more **TABL** lines of the format:

```
&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 downwards, 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 pattern is defined as two jets, each with a velocity of 10 m/s and a flow rate of 60 L/min. The first jet contains 0.2 of the total flow, the second, 0.8 of the total. The jets are centered at points 30° below the “equator,” and are separated by 180°.

```
&PROP ID='K-11', QUANTITY='SPRINKLER LINK TEMPERATURE', OFFSET=0.10, PART_ID='water_drops',
      FLOW_RATE=60., SPRAY_PATTERN_TABLE='TABLE1', SMOKEVIEW_ID='sprinkler_upright',
      PARTICLE_VELOCITY=10. /

&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.

Special Topic: 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 towards the design value and below that when 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:

```
&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 = 16.0 /
&RAMP ID = 'PR1' T = 2, F = 10.0 /
&RAMP ID = 'PR1' T = 3, F = 8.0 /
```

These lines would indicate that the pressure is 16.0 bar when the first sprinkler having properties of `My nozzle` activates, 10.0 bar when two sprinklers are active, and 8.0 bar when three or more sprinklers are active. When counting the number of active sprinklers, FDS accounts for all active sprinklers or nozzles with `PART_ID` associated with them.

When pressure ramps are used, both `FLOW_RATE` and `PARTICLE_VELOCITY` are associated with 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 $\dot{m}_w = K\sqrt{p}$ and droplet velocity by $v = C\sqrt{p}$. If spray pattern table is used, the coefficient C is determined separately for each line of the table. 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, the `PRESSURE_RAMP` for the water sprinklers would not count any active fuel spray nozzles.

15.3.2 Nozzles

Nozzles are very much like sprinklers, only they do not activate based on the standard RTI 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: flow_rate

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 left part of Figure 15.1. 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

$$N_n K \sqrt{p} = 10 \text{ l/min} \Rightarrow p = \left(\frac{10 \text{ l/min}}{N_n K} \right)^2 \quad (15.1)$$

where N_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.,
SMOKEVIEW_ID='sprinkler_upright' /

&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 accumulated water is tracked using a device measuring the accumulated mass per unit area, integrated over the total floor area. The total mass of accumulated water should increase from zero to 10 kg in 60 s. A comparison of the FDS prediction and this analytical result is shown in the right side of Figure 15.1. The small delay of the FDS result is caused by the time it takes from the droplets to fall down on the floor.

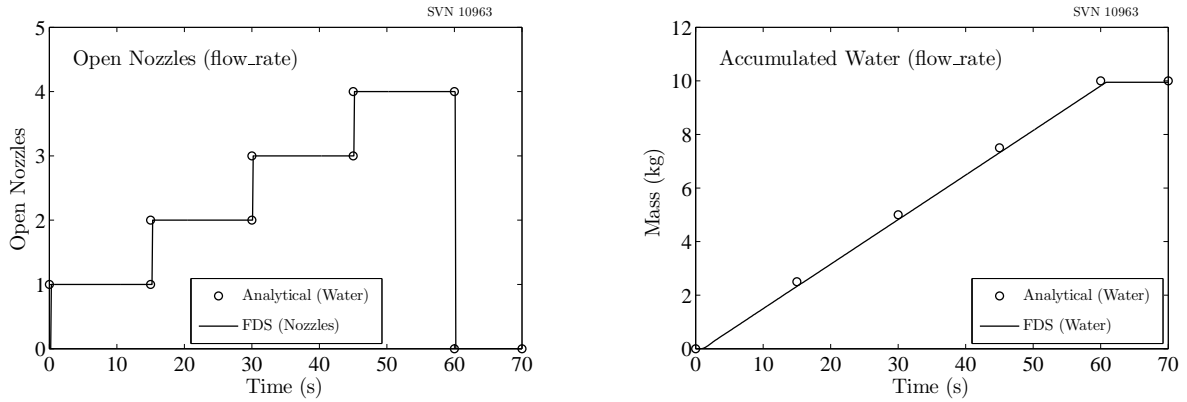


Figure 15.1: Output of the **flow_rate** test case.

15.3.3 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 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 15.4. The user specifies 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):

$$u_k(\mathbf{r}) = \underbrace{(u_k)_0}_{P0} + r_i \underbrace{\left(\frac{\partial u_k}{\partial x_i}\right)_0}_{PX(1:3)} + \frac{r_i r_j}{2} \underbrace{\left(\frac{\partial^2 u_k}{\partial x_i \partial x_j}\right)_0}_{PXX(1:3, 1:3)} \quad (15.2)$$

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'. It is up to the user to ensure that the device box encompasses the staggered storage locations (see the theory manual [1] for a discussion on the face-centered velocity storage locations).

15.3.4 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 $\sqrt{\text{m}\cdot\text{s}}$. ACTIVATION_TEMPERATURE is the link activation temperature in degrees C (Default 74 °C). INITIAL_TEMPERATURE is the initial temperature of the link in units of °C (Default TMPA).

15.3.5 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=3.28 /
```

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=3.28 /
```

where the two characteristic filling or “lag” times are of the form:

$$\delta t_e = \alpha_e u^{\beta_e} \quad ; \quad \delta t_c = \alpha_c u^{\beta_c} \quad (15.3)$$

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 15.1. 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 setting is 3.28 %/m (1 %/ft).

Table 15.1: Suggested Values for Smoke Detector Model. See Ref. [22] 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	—

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 smouldering 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. Use the following lines:

```
&SPEC ID='MY SMOKE', MW=29., MASS_EXTINCTION_COEFFICIENT=8700. /
&SURF ID='SMOULDER', TMP_FRONT=1000., MASS_FLUX(1)=0.0001, COLOR='RED' /
&VENT XB=0.6,1.0,0.3,0.7,0.0,0.0, SURF_ID='SMOULDER' /

&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' /

&DUMP SMOKE3D_QUANTITY='MY SMOKE', DT_PL3D=30. /
```

The same smoke detector model is used that was described above. Only now, the mass fraction of your 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. Note also that if you specify explicitly a smoke surrogate, you should set `SOOT_YIELD=0` on the `REAC` line to prevent FDS from including smoke as a component of the combustion product lumped species.

15.3.6 Beam Detection Systems

A beam detector can be defined by specifying the endpoints, (x_1, y_1, z_1) and (x_2, y_2, z_2) , of the beam and the total percent obscuration at which the detector activates. The two endpoints must lie in the same mesh. FDS determines which mesh cells lie along the linear path defined by the two endpoints. The beam detector response is evaluated as

$$\text{Obscuration} = \left(1 - \exp \left(-K_m \sum_{i=1}^N \rho_{\text{soot},i} \Delta x_i \right) \right) \times 100 \% \quad (15.4)$$

where i is a mesh cell along the path of the beam, $\rho_{\text{soot},i}$ 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=0.33 /
```

Since a single linear path cannot span more than one mesh, having a beam detector that crosses multiple meshes will require post processing. Break the beam detector path into multiple DEVC lines, one for each mesh that the beam crosses. The total obscuration is given by

$$\text{Obscuration} = \left[1 - \prod_{i=1}^N (1 - \text{output}_i / 100) \right] \times 100\% \quad (15.5)$$

where output_i is the FDS output for the beam detector of the i th path (note that the bracketed term contains a product rather than a sum).

Example Case: beam_detector

A 10 m by 10 m by 4 m compartment is filled with smoke, represented as 0.006 kg/kg of the lumped species variable, LUMPED SPECIES 1 footnote. In its default mode, the mixing controlled combustion model requires transport equations for two gas species – LUMPED SPECIES 1 and LUMPED SPECIES 2. The first is just the fuel gas, the second is a combination of the products of combustion. The default soot yield is 0.01 kg/kg, resulting in a uniform soot density of 71.9 mg/m³. Using the default mass extinction coefficient of 8700 m²/kg, the optical depth is calculated to be 0.626 m⁻¹. The compartment has a series of obstructions located at increasing distance from the front in increments of 1 m. The correlation for the output quantity VISIBILITY, Eq. (15.13), produces a visibility distance of 4.8 m. When viewing the smoke levels with Smokeview, you should just barely see the fifth obstacle which is at a distance of 5 m from the front of the compartment. If this is the case, Smokeview is properly displaying the obscuration of the smoke. Three beam detectors are also placed in the compartment. These all have a path length of 10 m, but are at different orientations within the compartment. Using the optical depth of 0.626 m⁻¹ and the path length of 10 m, the expected total obscuration is 99.81 %, which is the result computed by FDS for each of the three detectors.

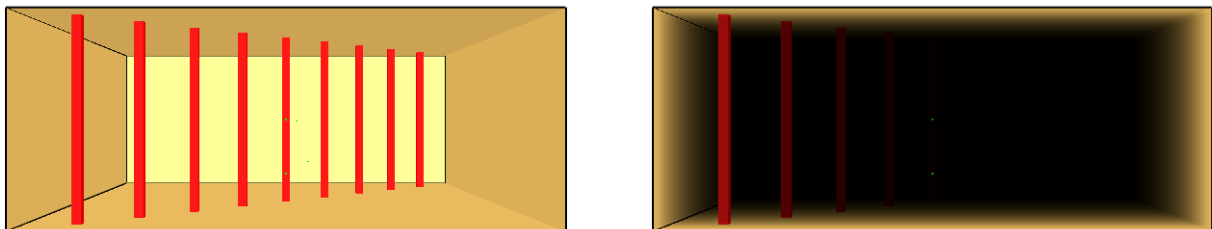


Figure 15.2: Output of the **beam_detector** test case.

15.3.7 Aspiration Detection Systems

An aspiration detection system groups together a series of soot 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

$$\text{Obscuration} = \left(1 - \exp \left(-K_m \frac{\sum_{i=1}^N \rho_{\text{soot},i}(t - t_{d,i}) \dot{m}_i}{\sum_{i=1}^N \dot{m}_i} \right) \right) \times 100 \text{ \% / m} \quad (15.6)$$

where \dot{m}_i is the mass `FLOWRATE` of the i th sampling location, $\rho_{\text{soot},i}(t - t_{d,i})$ is the soot density at the i th sampling location $t_{d,i}$ s prior (`DELAY`) to the current time t , and K_m is the `MASS_EXTINCTION_COEFFICIENT` associated with visible light.

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. 15.3.

15.3.8 Electrical Cable Failure

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 homogenous cylinder [23]. 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.

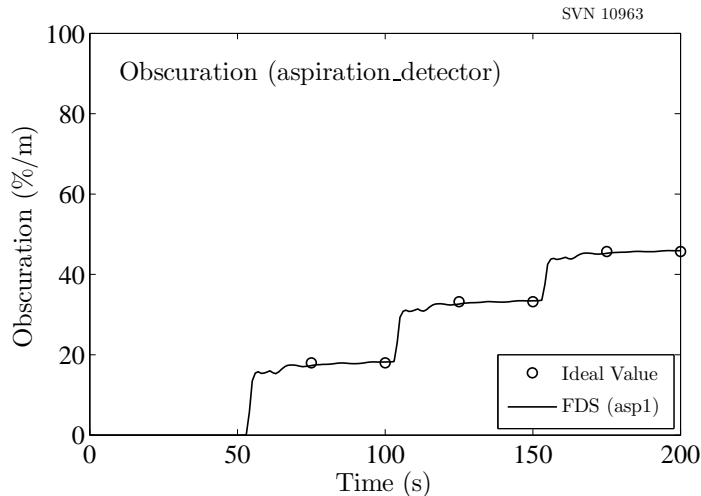


Figure 15.3: Output of **aspiration_detector** test case.

2. The cable is homogenous 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 homogenous 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 [24]. 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 centermost 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 centermost 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(1,1:3)=0.,0.,1.,
    STATIC=.TRUE. /
&INIT ID='Cable', XB=0.01,0.01,0.,0.,0.,0., N_PARTICLES=1, PART_ID='Cable Segment' /
&DEVC XYZ=0.01,0.00,0.000, ID='Cable Temp', INIT_ID='Cable',
    QUANTITY='INSIDE WALL TEMPERATURE', DEPTH=0.00152 /
```

The THIEF model assumes that the cable plastic material has a thermal conductivity of 0.2 W/m/K and a specific heat of 1.5 kJ/kg/K. 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. The first argument indicates that there can be more than one ORIENTATION. STATIC=.TRUE. 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 1.52 mm below the surface. This is typically the jacket thickness.

15.4 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 choice is dictated by the given `TRIP_DIRECTION`, which is just a positive or negative integer. 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 state when its value increases past the setpoint and a negative integer means the device will change state when its value decreases past the setpoint. The default value is +1.

LATCH If this logical value is set to `.TRUE.` the device will only change state once. The default value is `.TRUE..`

INITIAL_STATE This logical value is the initial state of the device. The default value is `.FALSE.` For example, if an obstruction associated with the device is to disappear, set `INITIAL_STATE=.TRUE.`

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 15.5 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 `.FALSE.` to `.TRUE.` 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 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:

$$\bar{x}^n = \bar{x}^{n-1} \text{SMOOTHING_FACTOR} + x(1 - \text{SMOOTHING_FACTOR}) \quad (15.7)$$

where n is the timestep, x is the instantaneous device output and \bar{x} is the smoothed output. The `SMOOTHING_FACTOR` defaults to 0 which means no smoothing is being performed.

15.4.1 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` 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 `.FALSE.` or `.TRUE.`, respectively. For example, the lines


```
&OBST XB=..., SURF_ID='...', DEVC_ID='det2' /
.
.
&DEVC XYZ=..., PROP_ID='...', ID='det1' /
&DEVC XYZ=..., PROP_ID='...', ID='det2', INITIAL_STATE=.TRUE. /
```

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 OBSTstructions to appear/disappear at the various designated times. These lines are part of the simple test case called **create_remove.fds**.

```
&HOLE XB=0.25,0.45,0.20,0.30,0.20,0.30, COLOR='RED', DEVC_ID='timer 1' /
&HOLE XB=0.25,0.45,0.70,0.80,0.70,0.80, COLOR='GREEN', DEVC_ID='timer 2' /
&OBST XB=0.70,0.80,0.20,0.30,0.20,0.30, COLOR='BLUE', DEVC_ID='timer 3' /
&OBST XB=0.70,0.80,0.60,0.70,0.60,0.70, COLOR='PINK', DEVC_ID='timer 4' /

&DEVC XYZ=..., ID='timer 1', SETPOINT= 1., QUANTITY='TIME', INITIAL_STATE=.FALSE./
&DEVC XYZ=..., ID='timer 2', SETPOINT= 2., QUANTITY='TIME', INITIAL_STATE=.TRUE. /
&DEVC XYZ=..., ID='timer 3', SETPOINT= 3., QUANTITY='TIME', INITIAL_STATE=.FALSE./
&DEVC XYZ=..., ID='timer 4', SETPOINT= 4., QUANTITY='TIME', INITIAL_STATE=.TRUE. /
```

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 it does exist initially. The red hole is created at 1 s because it does not exist initially (it is filled in with a red obstruction). The green hole is filled in at 2 s because it does exist (as a hole) initially. **You should always try a simple example first before embarking on a complicated creation/removal scheme for obstructions and holes.**

To learn how to create and remove obstructions multiple times, see Section 15.5.5 for information about the custom control feature.

An obstruction that makes up the boundary of a “pressure zone” (see Section 9.3) *can* be created or removed.

15.4.2 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 SURF_ID. For example, to control a fan with the device 'det2', do the following:

```
&SURF ID='FAN', VOLUME_FLUX=5. /
&VENT XB=..., SURF_ID='FAN', DEVC_ID='det2' /
&DEVC ID='det2', XYZ=..., QUANTITY='TIME', SETPOINT=30., INITIAL_STATE=.FALSE. /
```

Note that at 30 s, the “state” of the 'FAN' changes from .FALSE. to .TRUE., 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 in this example INITIAL_STATE=.TRUE., then the fan should “deactivate,” or 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.

Until further notice, 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.

15.5 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. 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 can not 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 activates cooling.

While HVAC is not the primary purpose of FDS, there are numerous situations where a system responds to the fire in a non-trivial way. The CTRL input is used to define these more complicated behaviors. A control function will take as input the outputs of one or more devices and/or control functions. In this manner, complicated behaviors can be simulated by making functions of other functions. For most of the control function types, the logical value output of the devices and control functions and the time they last changed state are used as the inputs.

For any object for which a DEVC_ID can be specified (such as OBST or VENT), a CTRL_ID can be specified instead. A control is identified by the ID parameter. The inputs to the control are identified by

Table 15.2: Control function types for CTRL

Function Type	Description
ANY	Changes state if any INPUTs are .TRUE.
ALL	Changes state if <u>all</u> INPUTs are .TRUE.
ONLY	Changes state if and <u>only</u> if N INPUTs are .TRUE.
AT_LEAST	Changes state if <u>at least</u> N INPUTs are .TRUE.
TIME_DELAY	Changes state DELAY s after INPUT becomes .TRUE.
CUSTOM	Changes state based on evaluating a RAMP of the function’s input
DEADBAND	Behaves like a thermostat
KILL	Terminates code execution if its sole INPUT is .TRUE.
RESTART	Dumps restart files if its sole INPUT is .TRUE.
SUM	Sums the outputs of the INPUTs
SUBTRACT	Subtracts the second INPUT from the first
MULTIPLY	Multiplies the outputs of the INPUTs
DIVIDE	Divides the first INPUT by the second
POWER	The first INPUT to the power of the second
PID	A Proportional-Integral-Derivative control function

the INPUT_ID parameter. INPUT_ID would be passed one or more ID strings from either devices or other controls.

If you want to design a system of controls and devices that involves multiple changes of state, include the attribute LATCH=.FALSE. on the relevant DEVC or CTRL input lines. By default, devices and controls may only change state once, like a sprinkler activating or smoke detector alarming. LATCH=.TRUE. by default for both devices and controls.

The output value of mathematical control functions are available for use by defining a DEVC with QUANTITY='CONTROL' and setting the CTRL_ID on the DEVC input to the ID of the control function.

15.5.1 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=.TRUE. /
```

The INITIAL_STATE of the control function SD is .TRUE., 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 .FALSE., 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 .FALSE. to .TRUE. 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 or “TRUE” inputs 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 .FALSE. to .TRUE. when 3, and only 3, detectors activate.

15.5.2 Control Function: TIME_DELAY

There is often a time delay 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', CTRL_ID='dry pipe' /
&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.

15.5.3 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, a `DEADBAND`, and the mode of operation by `ON_BOUND`. If `ON_BOUND='LOWER'`, the function changes state from its `INITIAL_VALUE` when the value of the `INPUT_ID` drops below the lower value in `DEADBAND` 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. like a cooling system.

For an HVAC example, the following lines of input would set up a simple thermostat:

```
&SURF ID='FAN', TMP_FRONT=40., VOLUME_FLUX=-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=.FALSE./
```

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 `.FALSE.`

15.5.4 Control Function: RESTART and KILL

There are times when you might only want to run a simulation until some goal is reached. Previously this could generally only be done by constantly monitoring the simulation's output and manually stopping the calculation when you determine that the goal has been reached. The `KILL` control function can do this automatically.

Additionally there are analyses where 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). 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 and output restart files 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.

15.5.5 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 .FALSE.) 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.

In the HVAC example above, we could set the system to function on a fixed cycle by using a CUSTOM control function based on time:

```
&SURF ID='FAN', TMP_FRONT=40., VOLUME_FLUX=-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 lines like

```
&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.

15.5.6 Control Function: Math Operations

The control functions that perform simple math operations (SUM, SUBTRACT, MULTIPLY, DIVIDE, and POWER) 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 would change when the square of the velocity exceeded 10 (see 15.4 for 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 /
```

15.5.7 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

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{de(t)}{dt} \quad (15.8)$$

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; and $u(t)$ is the output.

15.5.8 Combining Control Functions: A Pre-Action Sprinkler System

For a pre-action 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. 2 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 Figure 15.4.

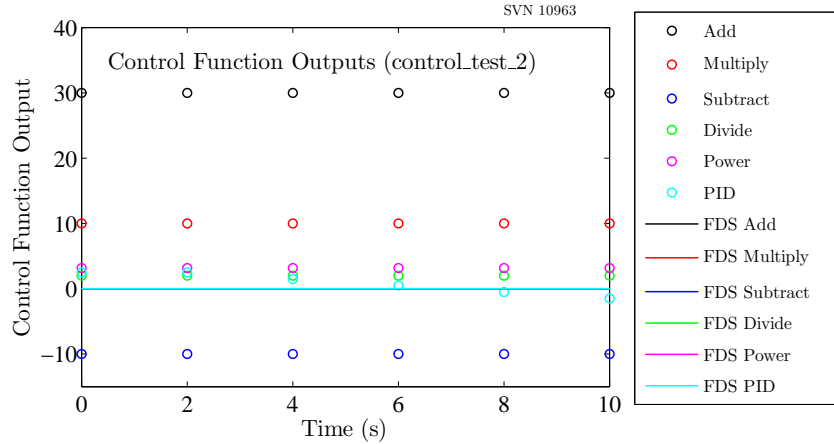


Figure 15.4: Results of the `control_test_2` case.

15.5.9 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'/'
```

15.5.10 Example Case: activate_vents

The simple test case called **activate_vents** demonstrates the several of the control functions. Figure 15.5 shows seven multiply-colored vents that activate at different times, depending on the particular timing or control function.

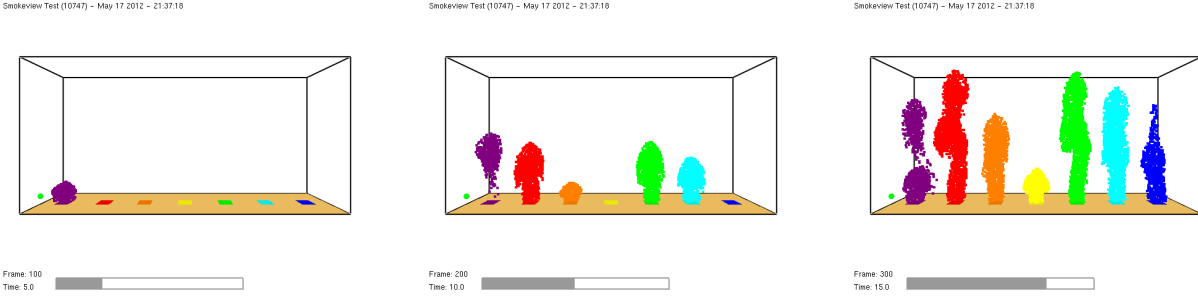


Figure 15.5: Output of the **activate_vents** test case at 5, 10 and 15 s.

15.6 Controlling a RAMP

15.6.1 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 one on of the RAMP input lines. When this is done, the current output of the DEVC is used, see 15.4 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, 15.5.6, or a PID function, 15.5.7). 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 /
```

15.6.2 Freezing the Output Value, Example Case: freeze_hrr

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 .TRUE. 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 15.6 shows the result of these inputs in the test case **freeze_device** 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=0.5,0.5,0.8, QUANTITY='TEMPERATURE', SETPOINT=200., INITIAL_STATE=.FALSE., ID='TEMP' /
&DEVC XYZ=0.5,0.5,0.8, QUANTITY='TIME', NO_UPDATE_DEVC_ID='TEMP', ID='FREEZE TIME' /
```

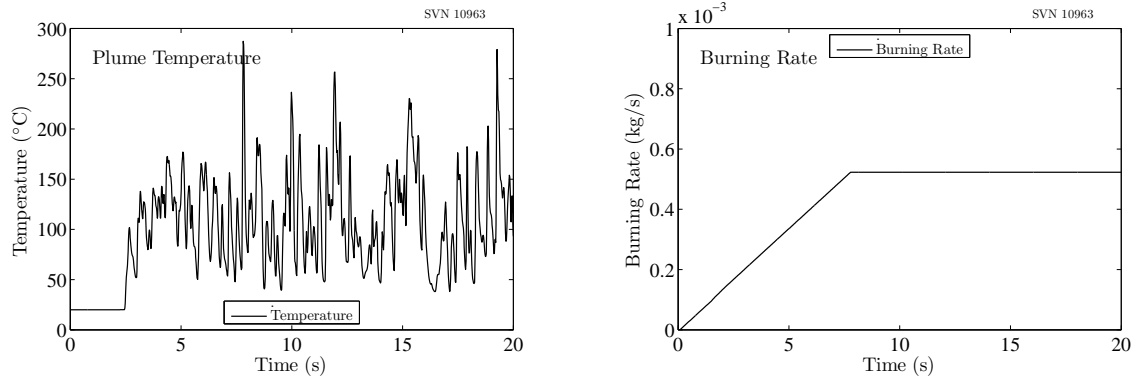



Figure 15.6: Temperature (left) and burning rate (right) outputs of the **freeze_hrr** test case

15.7 Visualizing FDS Devices Using Smokeview Objects

Smokeview generates visual representations of FDS devices using instructions found in a data file named `objects.svo`. These instructions correspond to OpenGL library calls, the same type of calls Smokeview uses to visualize FDS cases. New objects may be designed and drawn without modifying Smokeview and more importantly may be created by any user not just the FDS/Smokeview developers. This section gives an overview of Smokeview objects detailing what objects are available and how to modify them. Further documentation giving underlying technical details may be found in the Smokeview User's Guide [2].

Smokeview objects may be static or dynamic. A static object is defined entirely in terms of data and instructions found in the `objects.svo` file. For example, the `sensor` object is static, it is drawn as a small green sphere with a fixed diameter. Its appearance remains the same regardless of how an FDS input file is set up. A dynamic object is also defined using instructions found in `objects.svo` but in addition uses data specified on the `&PROP` namelist statement and/or data contained in a particle file. As a result, the appearance of dynamic objects depends on the particular FDS case that is run. For example, the `tsphere` object is dynamic. The diameter and an image used to cover the sphere (known as a texture map) is specified in an FDS input file.

15.7.1 Static Smokeview Objects

Smokeview objects consist of one or more frames or views. Smokeview then displays FDS devices in a normal/inactive state or in an active state. A sprinkler, for example, is drawn differently depending on whether it has activated or not. 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 frame. Tables 15.3 and 15.4 give a list of various static objects. Each entry shows an image of the object in its normal/inactive state and in its active state if it has one. The intersection of black tubes indicate the origin, the part of the device displayed at the (x,y,z) coordinate specified on the `&DEVC` input line.

The `SMOKEVIEW_ID` keyword found on the `&PROP` namelist statement is used to associate an FDS device with a Smokeview object. The following FDS input file lines were used to display the target device in Table 15.3.

```
&PROP ID='target' SMOKEVIEW_ID='target' /  
&DEVC XYZ=0.5,0.8,0.6, QUANTITY='TEMPERATURE' PROP_ID='target' /
```

Table 15.3: Single Frame Static Objects



SMOKEVIEW_ID	Image
sensor	
target	

Table 15.4: Dual Frame Static Objects

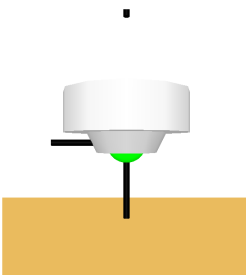
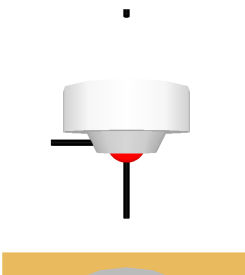


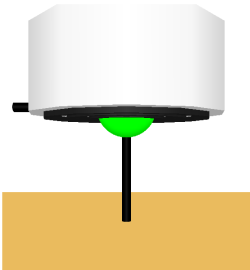
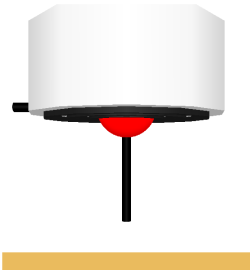
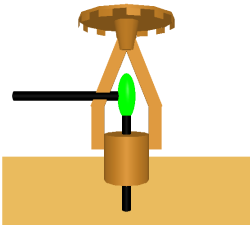
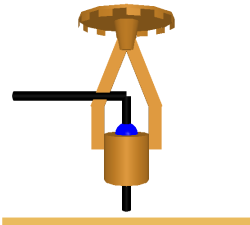
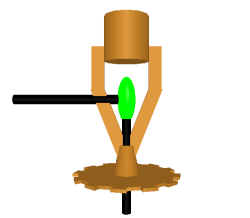
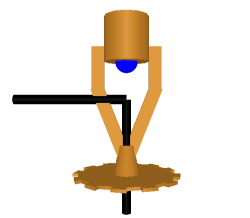
SMOKEVIEW_ID	Image	
	inactive	active
heat_detector		
nozzle		

Table 15.4: Dual Frame Static Objects (continued)

SMOKEVIEW_ID	Image	
	inactive	active
smoke_detector		
sprinkler_upright		
sprinkler_pendent		

15.7.2 Dynamic Smokeview Objects - Customized Using &PROP Parameters

The appearance of several Smokeview objects may be modified using data specified on the &PROP namelist statement in an FDS input file. Objects may also be customized using data stored in a particle file. This is discussed in the next section.

The SMOKEVIEW_PARAMETERS keyword on the &PROP namelist statement is used to customize the appearance of Smokeview objects. For example, the &DEVC and &PROP statements:

```
&PROP ID='sphere' SMOKEVIEW_PARAMETERS(1:4)='R=0','G=255','B=0',
          'D=0.5' SMOKEVIEW_ID='sphere' /
&DEVC XYZ=0.5,0.8,1.5, QUANTITY='TEMPERATURE' PROP_ID='sphere' /
```

create an FDS device drawn as a sphere colored green with diameter 0.5. Each parameter specified using the SMOKEVIEW_PARAMETERS keyword is a text string enclosed in single quotes. The text string is of the

form 'keyword=value' where possible keywords are found in the `objects.svo` file (labels beginning with '.'). For example, R, G, B and D may be used as keywords to customize the following sphere object:

```
OBJECTDEF // object for particle file sphere
sphere
:R=0 :G=0 :B=0 :D=0.1
$R $G $B setrgb
$D drawsphere
```

Another, Smokeview object, the `tsphere`, uses a texture map or picture to alter the appearance of the object. The texture map is specified using `SMOKEVIEW_PARAMETERS` keyword by placing the characters `t%` before the texture file name (e.g. `t%texturefile.jpg`).

Table 15.5 gives a list of dynamic objects and the keyword/parameter pairs used to specify them. Each entry shows an image of the object and the parameters used to customize its appearance.

Table 15.5: Dynamic Objects - Customized using `SMOKEVIEW_PARAMETERS` on a `&PROP` line

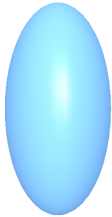
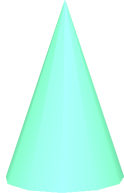
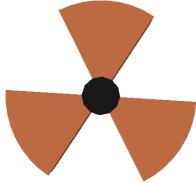



SMOKEVIEW_ID	SMOKEVIEW_PARAMETERS	Image
ball	<p><code>SMOKEVIEW_PARAMETERS (1:6) =</code> <code>'R=128','G=192','B=255',</code> <code>'DX=0.5','DY=.75','DZ=1.0'</code></p> <p>R, G, B - red, green, blue color components ranging from 0 to 255 DX, DY, DZ - amount ball is stretched along x, y, z axis respectively</p>	
cone	<p><code>SMOKEVIEW_PARAMETERS (1:5) =</code> <code>'R=128','G=255','B=192',</code> <code>'D=0.4','H=0.6'</code></p> <p>R, G, B - red, green, blue color components ranging from 0 to 255 D, H - diameter and length of cone respectively</p>	
fan	<p><code>SMOKEVIEW_PARAMETERS (1:11) =</code> <code>'HUB_R=0','HUB_G=0','HUB_B=0',</code> <code>'HUB_D=0.1','HUB_L=0.12',</code> <code>'BLADE_R=128','BLADE_G=64',</code> <code>'BLADE_B=32','BLADE_ANGLE=60.0',</code> <code>'BLADE_D=0.5','BLADE_H=0.09'</code></p> <p>HUB_R, HUB_G, HUB_B - red, green, blue color components of fan hub ranging from 0 to 255 HUB_D, HUB_L - diameter and length of fan hub BLADE_R, BLADE_G, BLADE_B - red, green, blue color components of fan blades ranging from 0 to 255 BLADE_ANGLE, BLADE_D, BLADE_H - angle, diameter and height of a fan blade</p>	

Table 15.5: Dynamic Objects (continued)

SMOKEVIEW_ID	SMOKEVIEW_PARAMETERS	Image
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 - red, green, blue color components ranging from 0 to 255 AX0, ELEV0, ROT0 - initial azimuth, elevation and rotation angle respectively ROTATION_RATE - rotation rate about z axis in degrees per second D - diameter of sphere tfile - name of texture map file</p>	
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 - red, green, blue color components ranging from 0 to 255 W, H - width and height of vent respectively ROT - angle that vent is rotated</p>	 <p>inactive vent</p>  <p>active vent</p>

15.7.3 Dynamic Smokeview Objects - Customized Using &PROP Parameters and Particle File Data

Particle file data may be used to customize the appearance of Smokeview objects. Any objects that have color labels named R, G, B (including those objects in Table 15.5) may be colored using particle file data. In addition, objects that use variable names that match shortened particle file quantity names² may be customized. This data may be used to alter the geometry or structure of the object using particle file data. For example, U-VEL, V-VEL, W-VEL and temp are shortened quantity names that correspond to the full names U-VELOCITY, V-VELOCITY, W-VELOCITY and TEMPERATURE. These full names are documented in Table 15.9 in this guide.

²Short forms of particle file quantity names appear in the Smokeview colorbar label and in the Smokeview File/Bounds dialog box.

The first three lines of the `velegg` object definition are:

```
OBJECTDEF // color with FDS quantity, stretch with velocity
velegg
:R=0 :G=0 :B=0 :D :U-VEL=1.0 :V-VEL=1.0 :W-VEL=1.0
```

The variables `U-VEL`, `V-VEL`, and `W-VEL` in the above line are also particle file quantities (shortened version) that may be selected in an FDS input file. If they are selected, then the `velegg` object may be used to display particle file information. This object colors the sphere using the currently selected Smokeview particle variable and stretches a sphere in the x, y and z directions using the `U-VEL`, `V-VEL`, and `W-VEL` velocity particle data respectively.

Table 15.6 documents those objects that can be customized using particle file data. These objects may be customized as before using data specified with the `SMOKEVIEW_PARAMETERS` keyword or using particle file data.

Table 15.6: Dynamic Objects - Customized using `SMOKEVIEW_PARAMETERS` on a `&PROP` line and Particle File Data

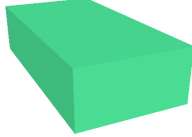

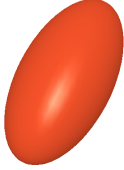
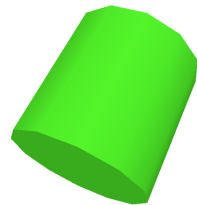
SMOKEVIEW_ID	SMOKEVIEW_PARAMETERS	Image
box	<p><code>SMOKEVIEW_PARAMETERS (1:6) =</code> <code>'R=192','G=255','B=128',</code> <code>'DX=0.25','DY=.5','DZ=0.125'</code></p> <p>R, G, B - red, green, blue color components ranging from 0 to 255 DX, DY, DZ - amount box is stretched along x, y, z axis respectively</p>	
tube	<p><code>SMOKEVIEW_PARAMETERS (1:5) =</code> <code>'R=255','G=0','B=0',</code> <code>'D=0.2','L=0.6'</code></p> <p>R, G, B - red, green, blue color components ranging from 0 to 255 D, L - diameter and length of tube respectively</p>	
velegg	<p><code>SMOKEVIEW_PARAMETERS (1:9) =</code> <code>:R=0 :G=0 :B=0</code> <code>:U-VEL=1.0 :V-VEL=1.0 :W-VEL=1.0</code> <code>:VELMIN :VELMAX :D</code></p> <p>R, G, B - red, green, blue color components ranging from 0 to 255 U-VEL, V-VEL, W-VEL - u, v, w components of velocity VELMIN, VELMAX - minimum and maximum velocity D - diameter of egg at maximum velocity</p>	

Table 15.6: Dynamic Objects (continued)

SMOKEVIEW_ID	SMOKEVIEW_PARAMETERS	Image
veltube	<p>SMOKEVIEW_PARAMETERS (1:9) = :R=0 :G=0 :B=0 :U-VEL=1.0 :V-VEL=1.0 :W-VEL=1.0 :VELMIN :VELMAX :D</p> <p>R, G, B - red, green, blue color components ranging from 0 to 255 U-VEL, V-VEL, W-VEL - u, v, w components of velocity VELMIN, VELMAX - minimum and maximum velocity D - diameter of tube at VELMAX</p>	

Before a calculation is started, carefully consider what information should be saved. All output quantities must be specified at the start of the calculation. In most cases, there is no way to retrieve information after the calculation ends if it was not specified from the start. There are several different ways of visualizing the results of a calculation. Most familiar to experimentalists is to save a given quantity at a single point in space so that this quantity can be plotted as a function of time, like a thermocouple temperature measurement. The namelist group `DEV`, described previously, is used to specify point measurements.

To visualize the flow patterns better, save planar slices of data, either in the gas or solid phases, by using the `SLCF` (SLiCe File) or `BNDF` (BouNDary File) namelist group. Both of these output formats permit you to animate these quantities in time.

For static pictures of the flow field, use the `PLot3D` files that are automatically generated 5 times a run. `Plot3D` format is used by many CFD programs as a simple way to store specified quantities over the entire mesh at one instant in time.

Finally, tracer particles can be injected into the flow field from vents or obstacles, and then viewed in Smokeview. Use the `PART` namelist group to control the injection rate, sampling rate and other parameters associated with particles.

Note: unlike in FDS version 1, particles are no longer used to introduce heat into the flow, thus particles no longer are ejected automatically from burning surfaces.

15.8 Output Control Parameters: The `DUMP` Namelist Group

The namelist group `DUMP` contains parameters (Table 16.6) that control the rate at which output files are written, and various other global parameters associated with output files. This namelist group is new starting in FDS 5, although its parameters have been specified via other namelist groups in past versions.

NFRAMES Number of output dumps per calculation. The default is 1000. Device data, slice data, particle data, isosurface data, 3D smoke data, boundary data, solid phase profile data, and control function data are saved every $(T_END - T_BEGIN) / NFRAMES$ seconds unless otherwise specified using `DT_DEV`, `DT_SLCF`, `DT_PART`, `DT_ISO`, `DT_BNDF`, `DT_PROF`, or `DT_CTRL`. Note that `DT_SLCF` controls Smoke3D output. `DT_HRR` controls the output of heat release rate and associated quantities.

MASS_FILE If `.TRUE.`, produce an output file listing the total masses of all gas species as a function of time. It is `.FALSE.` by default because the calculation of all gas species in all mesh cells is time-consuming. The parameter `DT_MASS` controls the frequency of output.

MAXIMUM_PARTICLES Maximum number of Lagrangian particles that can be included on any mesh at any given time. (Default 500000)

SMOKE3D If `.FALSE.`, do not produce an animation of the smoke and fire. It is `.TRUE.` by default.

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 `.FALSE.`, 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 `.TRUE.`, 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 `.FALSE.` by default.

15.9 Output File Types

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. It is important to remember that you must explicitly declare in the input file most of the FDS output data. A considerable amount of the input file is usually devoted to this.

15.9.1 Device Output: The `DEVC` Namelist Group

For many commonly used measurement devices there is no need to associate a specific `PROP` line to the `DEVC` entry. In such cases, use the character string `QUANTITY` to indicate that a particular gas or solid phase quantity at the point should be recorded in the output file with the suffix `_devc.csv`. The quantities are listed in Table 15.9. Many of the gas phase quantities are self-explanatory. For example, if you just want to record the time history of the temperature at a given point, add

```
&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'. In this case, the `ID` has no other role than as a column label in the output file. Note that versions of FDS prior to version 5 used an 8 cell linear interpolation for a given *gas phase* point measurement. In other words, if you specified a point via the triplet of real numbers, `XYZ`, FDS would calculate the value of the quantity by linearly interpolating the values defined at the centers of the 8 nearest cells. Starting in FDS 5, this is no longer done. Instead, FDS reports the value of the `QUANTITY` in the cell where the point `XYZ` is located.

Devices on Solid Surfaces

When prescribing a solid phase quantity, be sure to position the probe 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 the `STATISTICS` feature described in Section 15.10.10 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 `IOR=1`, negative x direction `IOR=-1`, positive y `IOR=2`, negative `IOR=-2`, positive z `IOR=3`, and negative z `IOR=-3`. 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 probe and try again. 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.

Non-Pointwise Devices

In addition to point measurements, the `DEVC` group can be used to report integrated quantities (See Table 15.9). For example, you may want to know the mass flow out of a door or window. To report this, add the line

```
&DEVC XB=0.3,0.5,2.1,2.5,3.0,3.0, QUANTITY='MASS FLOW', ID='whatever' /
```

Note that in this case, a plane is specified rather than a point. The sextuplet `XB` is used for this purpose. Notice when a flow is desired, two of the six coordinates need to be the same. Another `QUANTITY`, `HRR`, can

be used to compute the total heat release rate within a subset of the domain. In this case, the sextuplet XB ought to define a volume rather than a plane. Specification of the plane or volume over which the integration is to take place can only be done using XB – avoid planes or volumes that cross multiple mesh boundaries. FDS has to decide which mesh to use in the integration, and it chooses the finest mesh overlapping the centroid of the designated plane or volume.

Linear Array of Point Devices

The DEVC input construct enables you to specify a linear array of devices. By adding the parameter POINTS and using the sextuple coordinate array XB, you can direct FDS to create a line of devices from (x_1, y_1, z_1) to (x_2, y_2, z_2) as follows:

```
&DEVC XB=X1,X2,Y1,Y2,Z1,Z2, QUANTITY='TEMPERATURE', ID='TC Tree', POINTS=20 /
```

In a file called **CHID_line.csv**, there will be between 1 and 4 columns of data associated with this single DEVC line. If X1 is different than X2, 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 presented as a running average. The averaging time is given by DT_DEVC_LINE on the DUMP line. It is 0.25 times the total simulation time by default. This is a convenient way to output a time-averaged linear profile of a quantity, like an array of thermocouples.

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=.TRUE. to the DEVC line. This is convenient if you have multiple arrays of data that use the same coordinates.

Typically, a *line* of devices records a steady-state profile of a particular quantity. For example, the velocity profile in a doorway or the average concentration of a particular scalar quantity spanning the width of a buoyant plume are convenient means to compare model and experiment. In addition, if you set STATISTICS='RMS' on the DEVC line, the output will be the *root mean square* value of the quantity rather than its mean, $\bar{\phi}$:

$$\phi_{\text{rms}} = \sqrt{\frac{\sum_{i=1}^n (\phi_i - \bar{\phi})^2}{n}} \quad (15.9)$$

Note that the 'RMS' statistic is only appropriate for a line of devices. It is not meant to be used with single devices that record a time history of a given quantity. The *root mean square* is only meaningful in terms of a steady-state output quantity.

15.9.2 Quantities within Solids: The PROF Namelist Group

FDS uses a fine, non-uniform, one-dimensional mesh at each boundary cell to compute heat transfer within a solid. The parameters (Table 16.19) to specify a given PROFILE are similar to those used to specify a surface quantity in the DEVC group. XYZ designates the triplet of coordinates, QUANTITY is the physical quantity to monitor, 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 given solid obstruction:

```
&PROF XYZ=..., QUANTITY='TEMPERATURE', ID='TU1SA_FDS', IOR=3 /
```

Other possible quantities are the total density of the wall (QUANTITY = 'DENSITY') or densities of solid material components (QUANTITY = 'MATL_ID'), where MATL_ID is the name of the material.

Each PROF line creates a separate file. This may be more than is needed. Sometimes, all you want to know is the temperature at a certain depth. To get an inner wall temperature, you can also just use a device as follows:

```
&DEVC XYZ=..., QUANTITY='INSIDE WALL TEMPERATURE', DEPTH=0.005, ID='Temp_1', IOR=3 /
```

The parameter `DEPTH` (m) indicates the distance inside the solid surface. Note that this `QUANTITY` is allowed only as a `DEVC`, not a `BNDP`, output. Also note that if the wall thickness is decreasing over time due to the solid phase reactions, the distance is measured from the current surface, and the measurement point is 'moving' towards the back side of the solid. Eventually, the measurement point may get out of the solid, in which case it starts to show ambient 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.

15.9.3 Animated Planar Slices: The `SLCF` Namelist Group

The `SLCF` ("slice file") namelist group parameters (Table 16.24) 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.

If the "slice" is a volume specified with `XB`, then its output frequency is controlled by the parameter `DT_SL3D`. By default, FDS sets `DT_SL3D=(T_END-T_BEGIN)/5`, the same default frequency as `Plot3D` file output. The user may specify a different value of `DT_SL3D` on `DUMP`. The user is cautioned that 3D 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=.TRUE.` If two `SLCF` entries are in the same plane, then only one of the lines needs to have `VECTOR=.TRUE.` 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=.TRUE.` 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. Note also that this feature should only be used for scalar quantities that are computed at cell centers, like temperatures, mass fractions, *etc.*

Slice file information is recorded in files (See Section 19.6) labeled **CHID_{*n*}.sf**, where n is the index of the slice file. A short fortran program **fds2ascii.f** produces a text file from a line, plane or volume of data. See Section 15.11 for more details.

15.9.4 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 15.9. 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.

Note that `BNDF` files (Section 19.8) can become very large, so be careful in prescribing the time interval. 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=.FALSE.` is included, the obstruction is not colored. To turn off all boundary drawing, set `BNDF_DEFAULT=.FALSE.` on the `MISC` line. Then individual obstructions can be turned back on with `BNDF_OBST=.TRUE.` 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).

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=.TRUE.` 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 because it enables you to visualize the values that FDS actually computes.

15.9.5 Animated Isosurfaces: The `ISO` Namelist Group

The `ISO` (“ISOsurface File”) namelist group is used to specify the output of gas phase scalar quantities, as three dimensional animated contours. For example, a 300 °C temperature isosurface shows where the gas temperature is 300 °C. Three different values of the temperature can be saved via the line:

```
&ISO 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_ISO` on the `DUMP` line).

Any gas phase quantity can 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` or `COLOR_SPEC_ID` corresponding to the given `QUANTITY` if necessary.

15.9.6 Plot3D Static Data Dumps

By default, flow field data in Plot3D format is output 5 times a run. 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. The first and fifth quantities can be changed with the parameters `PLOT3D_QUANTITY(1)` and `PLOT3D_QUANTITY(5)` on the `DUMP` line. 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)`.

Note that there can only be one DUMP line.

Data stored in Plot3D [25] files (See Section 19.7) use a format developed by NASA and used by many CFD programs for representing simulation results. Plot3D data is visualized in three ways: as 2D 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. 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=.TRUE.` 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.

15.9.7 SMOKE3D: Realistic Smoke and Fire

When you do a fire simulation, FDS automatically creates two output files that are rendered by Smokeview as realistic looking smoke and fire. By default, the output quantities are the 'MASS FRACTION' of 'SOOT' and 'HRRPUV' (Heat Release Rate Per Unit Volume) are used in the visualization. You have the option of rendering any other species mass fraction instead of 'SOOT', so long as the `MASS_EXTINCTION_COEFFICIENT` (either from the REAC line, or over-ridden by the value on the SPEC line) is appropriate in describing the attenuation of visible light by the specified gas species. The alternative gas species is given by `SMOKE3D_QUANTITY` on the DUMP line. If the specified quantity requires the additional specification of a particular species, use `SMOKE3D_SPEC_ID` to provide the `SPEC_ID`. See the Smokeview User's Guide for more details on how these quantities are rendered.

Here is an example of how to control 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.

```
&SPEC ID='MY SMOKE', MW=29., MASS_EXTINCTION_COEFFICIENT=8700. /
&SURF ID='NO FIRE', TMP_FRONT=1000., MASS_FLUX(1)=0.0001, COLOR='RED' /
&VENT XB=0.6,1.0,0.3,0.7,0.0,0.0, SURF_ID='NO FIRE' /
&DUMP SMOKE3D_QUANTITY='MASS FRACTION', SMOKE3D_SPEC_ID='MY SMOKE' /
```

The production rate of 'MY SMOKE' is 0.0001 kg/m²/s, applied to an area of 0.16 m². The `MASS_EXTINCTION_COEFFICIENT` is passed to Smokeview to be used for visualization.

15.10 Special Output Quantities

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.

15.10.1 Heat Release Rate

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 parameter associated with it is `DT_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:

$$\underbrace{\frac{\partial}{\partial t} \int \rho h_s dV}_{Q_ENTH} = \underbrace{- \int \rho \mathbf{u} h_s \cdot d\mathbf{S}}_{Q_CONV} + \underbrace{\int k \nabla T \cdot d\mathbf{S}}_{Q_COND} + \underbrace{\sum_{\alpha} \int h_{s,\alpha} \rho D_{\alpha} \nabla Y_{\alpha} \cdot d\mathbf{S}}_{Q_DIFF} \\ - \underbrace{\int \dot{\mathbf{q}}_r'' \cdot d\mathbf{S}}_{Q_RADI} + \underbrace{\int \dot{q}''' dV}_{HRR} + \underbrace{\int \frac{d\bar{p}}{dt} dV}_{Q_PRES} + \underbrace{\int (-\dot{q}_b''' + \dot{m}_b''' (h_{s,b} - h_s)) dV}_{Q_PART} \quad (15.10)$$

An additional column, `Q_TOTAL`, includes the sum of the terms on the right hand side of the equation. Ideally, this sum should equal `Q_ENTH`. All terms are reported in units of kW.

The other columns in the 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.

15.10.2 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 [26]. The intensity of monochromatic light passing a distance L through smoke is attenuated according to

$$I/I_0 = e^{-KL} \quad (15.11)$$

The light extinction coefficient, K , is a product of the density of smoke particulate, ρY_s , and a mass specific extinction coefficient that is fuel dependent

$$K = K_m \rho Y_s \quad (15.12)$$

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 Section 15.3.7, Section 15.3.5, and Section 15.3.6

Estimates of visibility through smoke can be made by using the equation

$$S = C/K \quad (15.13)$$

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 [26]. Since K varies from point

to point in the domain, the visibility S does as well. Keep in mind that FDS can only track smoke whose production rate and composition are specified. Predicting either is beyond the capability of the present version of the model.

Three parameter control smoke production and visibility; each parameter is input on the REAC line. The first parameter is SOOT_YIELD, which is the fraction of fuel mass that is converted to soot if the simple chemistry approach is being used. The second parameter is called the MASS_EXTINCTION_COEFFICIENT, and it is the K_m in Eq. (15.12). The default value is 8700 m²/kg, a value suggested for flaming combustion of wood and plastics³. The third parameter is called the VISIBILITY_FACTOR, the constant C in Eq. (15.13). It is 3 by default.

The gas phase output quantity 'EXTINCTION COEFFICIENT' is K . A similar quantity is the 'OPTICAL DENSITY', $K/2.3$, the result of using \log_{10} in the definition

$$D \equiv -\frac{1}{L} \log_{10} \left(\frac{I}{I_0} \right) = K \log_{10} e \quad (15.14)$$

The visibility S is output via the keyword 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.

15.10.3 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 [28] 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:

$$\begin{aligned} (H - z_{int}) T_u + z_{int} T_l &= \int_0^H T(z) dz = I_1 \\ (H - z_{int}) \frac{1}{T_u} + z_{int} \frac{1}{T_l} &= \int_0^H \frac{1}{T(z)} dz = I_2 \end{aligned}$$

Solve for z_{int} :

$$z_{int} = \frac{T_l(I_1 I_2 - H^2)}{I_1 + I_2 T_l^2 - 2 T_l H} \quad (15.15)$$

Let T_l be the temperature in the lowest mesh cell and, using Simpson's Rule, perform the numerical integration of I_1 and I_2 . T_u is defined as the average upper layer temperature via

$$(H - z_{int}) T_u = \int_{z_{int}}^H T(z) dz \quad (15.16)$$

³For most flaming fuels, a suggested value for K_m is 8700 m²/kg \pm 1100 m²/kg at a wavelength of 633 nm [27]

Further discussion of similar procedures can be found in Ref. [29].

The quantities `LAYER HEIGHT`, `UPPER TEMPERATURE` and `LOWER TEMPERATURE` can be designated via “device” (`DEVC`) lines in the input file⁴. For example, the entry

```
&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 $z = 3$. If multiple meshes are being used, the vertical path *cannot* cross mesh boundaries.

15.10.4 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} [30]

$$\rho_{TC} c_{TC} \frac{dT_{TC}}{dt} = \epsilon_{TC} (U/4 - \sigma T_{TC}^4) + h(T_g - T_{TC}) = 0 \quad (15.17)$$

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, $h = k Nu/d_{TC}$. The bead `BEAD_DIAMETER`, `BEAD_EMISSIVITY`, `BEAD_DENSITY`, and `BEAD_SPECIFIC_HEAT` are given on the associated `PROP` line. To over-ride the calculated value of the heat transfer coefficient, set `BEAD_H_FIXED` on the `PROP` line ($W/m^2/K$). 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.

15.10.5 Heat Fluxes and Thermal Radiation

There are various ways of recording the heat flux at a solid boundary. If you want to record the *net* heat flux to the surface, $\dot{q}_c'' + \dot{q}_r''$, use the `QUANTITY` called 'NET HEAT FLUX'. The individual components, the *net* convective and radiative fluxes, are 'CONVECTIVE HEAT FLUX' and 'RADIATIVE HEAT FLUX', respectively. If you want to compare predicted heat flux with a measurement, you often need to use 'GAUGE HEAT FLUX'. The difference between 'NET HEAT FLUX' and 'GAUGE HEAT FLUX' is that the former is the rate at which energy is absorbed by the solid surface; the latter is the amount of energy that would be absorbed if the surface were cold (or some specified temperature T_G):

$$\dot{q}_r''/\epsilon + \dot{q}_c'' + h(T_w - T_G) + \sigma(T_w^4 - T_G^4)$$

If the heat flux gauge used in an experiment has a temperature other than ambient, set `GAUGE_TEMPERATURE` (T_G) on the `PROP` line associated with the device. When comparing against a radiometer measurement, use `RADIOMETER`:

$$\dot{q}_r''/\epsilon + \sigma(T_w^4 - T_\infty^4)$$

For diagnostic purposes it is sometimes convenient to output the 'INCIDENT HEAT FLUX':

$$\dot{q}_r''/\epsilon + \sigma T_w^4 + \dot{q}_c''$$

Note that the sign of the output of heat flux is different than the sign of the input of a heat flux. A positive output quantity for heat flux means heat is being transferred into the surface.

⁴Note that in FDS 5 and beyond, these quantities are no longer available as slice files.

All of the above heat flux output quantities are defined at a solid surface. To record the heat flux away from a solid surface, use a Lagrangian particle as a surrogate for a heat flux gauge, as in the following example:

```
&DEVC ID='flux', INIT_ID='f1', QUANTITY='RADIATIVE HEAT FLUX' /
&INIT ID='f1', XYZ=0.45,0.0,0.3, N_PARTICLES=1, PART_ID='rad gauge' /
&PART ID='rad gauge', STATIC=.TRUE., ORIENTATION(1,1:3)=-1,0,0, SURF_ID='target' /
&SURF ID='target', RADIUS=0.001, GEOMETRY='SPHERICAL' /
```

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”.

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=0.45,0.0,0.3, N_PARTICLES=34, DX=0.05, PART_ID='rad gauge' /
```

For more information about specifying arrays of devices via the parameter `POINTS`, see Section 15.9.1. Note also the parameter `DX` on the `INIT` line that creates a line of particles starting at the point `XYZ` and repeating every 0.05 m.

15.10.6 Droplet Output Quantities

This section lists various output quantities associated with droplets and particles. Note whether or not the quantity is appropriate for the gas or solid surfaces.

Droplet Quantities on Solid Surfaces

It is possible to record various properties of droplets and particles. Some of the output quantities are associated with solid boundaries. For example, ‘`MPUA`’ is the Mass Per Unit Area of the droplets named `PART_ID`. Likewise, ‘`AMPUA`’ is the Accumulated Mass Per Unit Area. Both of these are given in units of kg/m^2 . Think of these outputs as measures of the instantaneous mass density per unit area, and the accumulated total, respectively. It should be noted that these quantities are not identical measures. 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. AMPUA will only count each particle or droplet once and only count the particle or droplet when it reaches the floor.⁵ MPUA counts a particle whenever it is on any solid surface, including the walls. If the particle or droplet moves from one solid wall cell to another, then 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/m^2 . Since a typical sprinkler simulation only tracks a small fraction of the droplets emitted from a sprinkler, both MPUA and CPUTA also perform an exponential smoothing. This avoids having spotted distributions on surfaces due to the infrequent arrival of particles that likely have a high weighting factor.

⁵Be aware of the fact that the default behavior for droplets hitting the “floor,” that is, the plane $z = Z_{\text{MIN}}$, is to disappear (`POROUS_FLOOR=.TRUE.` 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.

In the test case **bucket_test**, a single sprinkler is 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

$$180 \frac{\text{L}}{\text{min}} \times 1 \frac{\text{kg}}{\text{L}} \times \frac{1}{60} \frac{\text{min}}{\text{s}} \times 30 \text{ s} = 90 \text{ kg} \quad (15.18)$$

In the simulation, the quantity 'AMPUA' with STATISTICS='SURFACE INTEGRAL' is applied to the DEVC line. 'AMPUA' records the accumulated water mass per unit area (kg/m^2), analogous to actual buckets the size of a grid cell. Summing the values of 'AMPUA' over the entire floor yields a total of 90 kg (Fig. 15.7). Note that there really is no need to time-average the results. The quantity is inherently accumulating.

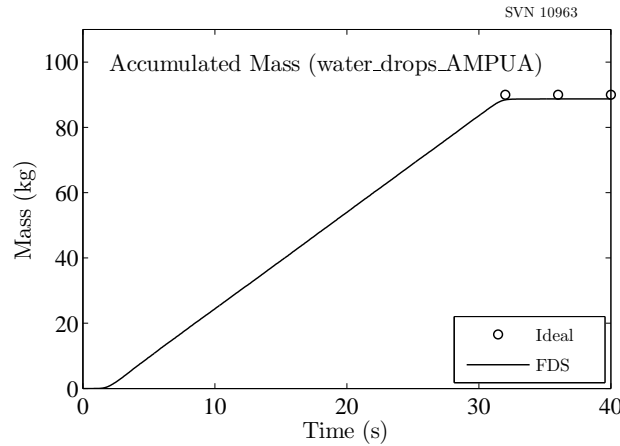


Figure 15.7: Accumulated water collected at the floor in the `bucket_test` case.

Droplet Mass and Fluxes in Gas Phase

Away from solid surfaces, 'MPUV' is the Mass Per Unit Volume of the droplets as they fly through the air, in units of kg/m^3 . 'PARTICLE FLUX X', 'PARTICLE FLUX Y', and 'PARTICLE FLUX Z' produce *only slice* and *Plot3D* files of the mass flux of droplets in the x , y , and z directions, respectively, in units of $\text{kg/m}^2/\text{s}$.

Local Spray Properties

Detailed experimental measurements of sprays 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. Note that in FDS, the PDPA device cannot produce complete droplet size distributions, but only various mean properties.

By default, the PDPA device output at time t is computed as a time integral

$$F(t) = \frac{1}{\min(t, t_e) - t_s} \int_{t_s}^{\min(t, t_e)} f(t) dt \quad (15.19)$$

but instantaneous values can be obtained by setting `PDPA_INTEGRATE = .FALSE.` on the corresponding PROP line, in which case

$$F(t) = f(t) \quad (15.20)$$

The function $f(t)$ has two forms:

$$f_1(t) = \frac{\sum_i n_i d_i^m x}{\sum_i n_i d_i^n} \quad ; \quad f_2(t) = \frac{\sum_i n_i x}{V} \quad (15.21)$$

where n_i is the number of real particles represented by the single simulated particle, d_i is the particle diameter, and x is the quantity to be measured. In each case, the summation goes over all the particles within a sphere with radius PDPA_RADIUS and centered at the location given by the device XYZ.

The first form $f_1(t)$ is used for the computation of various mean diameters, with associated properties defined using the following keywords on the PROP line:

PDPA_M m , exponent m of diameter.

PDPA_N n , exponent n of diameter. In case $m = n$, the exponent $1/(m - n)$ is removed from the formula.

The second form ($f_2(t)$) is used for the computation of mass and energy related variables that do not include the diameter weighting. The concentrations are based on the sampling volume, V , defined by PDPA_RADIUS. The quantity used for x can be chosen with the keyword QUANTITY. A summary of the available PDPA quantities is shown in Table 15.7.

Table 15.7: Output quantities available for PDPA output.

QUANTITY	x	$f(x)$	Unit
'DIAMETER' (default)	1	f_1	μm
'ENTHALPY'	$(4/3)\rho_i r_i^3 (c_{p,i}(T_i)T_i - c_{p,i}(T_m)T_m)$	f_2	kJ/m^3
'PARTICLE FLUX X'	$(4/3)\rho_i r_i^3 u_i$	f_2	$\text{kg/m}^2\text{s}$
'PARTICLE FLUX Y'	$(4/3)\rho_i r_i^3 v_i$	f_2	$\text{kg/m}^2\text{s}$
'PARTICLE FLUX Z'	$(4/3)\rho_i r_i^3 w_i$	f_2	$\text{kg/m}^2\text{s}$
'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'	$(u_i^2 + v_i^2 + w_i^2)^{1/2}$	f_1	m/s
'TEMPERATURE'	T_i	f_1	$^\circ\text{C}$
'MASS CONCENTRATION'	$(4/3)\rho_i r_i^3$	f_2	kg/m^3
'NUMBER CONCENTRATION'	1	f_2	

* T_m is the melting temperature of the associated species.

It is also possible to output histograms of PDPA output quantities. When PDPA_HISTOGRAM is set to .TRUE., histogram bin counts are output to a csv file from all devices associated with this PROP. 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 $d_i^m x$. Note that when making a histogram of diameters, the limits must be given in meters, not microns. The output file contains raw bin counts. Values falling outside the histogram limits are included in the counts of the first and last bins. The properties of the PDPA device are defined using the following keywords 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.
.TRUE. 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 .FALSE. to force $V = 1$ in the formula for $f_2(t)$.

QUANTITY Specified on PROP line for choosing the variable x .

PDPA_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  
      PDPA_HISTOGRAM=.TRUE.  
      PDPA_HISTOGRAM_NBINS=20  
      PDPA_HISTOGRAM_LIMITS=0,2000E-6  
      /  
&DEVC XYZ=0.0,0.0,1.0, QUANTITY='PDPA', PROP_ID='pdpa_d' /
```

15.10.7 Interfacing with Structural Models

FDS solves a one-dimensional heat conduction equation for each boundary cell marking the interface between gas and solid, assuming that material properties for the material layer(s) are provided. The results can be transferred (via either DEVC or BNDF output) to other models that predict the mechanical response of the walls or structure. For many applications, the 1-D solution of the heat conduction equation is adequate, but in situations where it is not, another approach can be followed. FDS includes a calculation of the Adiabatic Surface Temperature (AST), a quantity that is representative of the heat flux to a solid surface. Following the idea proposed by Ulf Wickstrom [31], the following equation can be solved via a simple iterative technique to determine an effective gas temperature, T_{AST} :

$$\dot{q}_r'' + \dot{q}_c'' = \epsilon \sigma (T_{\text{AST}}^4 - T_w^4) + h(T_{\text{AST}} - T_w) \quad (15.22)$$

The sum $\dot{q}_r'' + \dot{q}_c''$ is the *net* heat flux onto the solid surface, whose temperature is T_w . The heat fluxes and surface temperature are computed in FDS, and they are inter-dependent. The computed wall temperature affects the net heat flux and vice versa. However, because FDS only computes the solution to the 1-D

heat conduction equation in the solid, it may be prone to error if lateral heat conduction within the solid is significant. Thus, in some scenarios neither the FDS-predicted heat fluxes or the surface temperature can be used as an accurate indicator of the thermal insult from the hot, smokey gases onto solid objects.

Of course, both the heat fluxes, \dot{q}_r'' and \dot{q}_c'' , and the surface temperature, T_w can be passed from FDS to the other model, and suitable corrections can be made based on a presumably more accurate prediction of the solid temperature. Alternatively, the single quantity, T_{AST} , can be transferred, as this is the temperature that the solid surface effectively “sees.” It represents the gas phase thermal environment, however complicated, but it does not carry along the uncertainty associated with the simple 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.

15.10.8 Useful Solid Phase Outputs

In addition to the `PROFILE` 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`.

To record the change in a 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'` on a `MATL` line. 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. Note also that `'SOLID DENSITY'` is only available as a `DEVC` (device) quantity.

15.10.9 Fractional Effective Dose (FED) and Fractional Irritant Concentration (FIC)

The Fractional Effective Dose index (FED), developed by Purser [32], is a commonly used measure of human incapacitation due to the combustion gases. The specification of gases and mixtures is explained in Chapter 11. The FED value is calculated as

$$FED_{tot} = (FED_{CO} + FED_{CN} + FED_{NO_x} + FED_{irr}) \times HV_{CO_2} + FED_{O_2} \quad (15.23)$$

The fraction of an incapacitating dose of CO is calculated as

$$FED_{CO} = \int_0^t 2.764 \times 10^{-5} (C_{CO}(t))^{1.036} dt \quad (15.24)$$

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

$$FED_{CN} = \int_0^t \left(\frac{\exp\left(\frac{C_{CN}(t)}{43}\right)}{220} - 0.0045 \right) dt \quad (15.25)$$

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

$$C_{CN} = C_{HCN} - C_{NO_2} \quad (15.26)$$

The fraction of an incapacitating dose of NO_x is calculated as

$$\text{FED}_{\text{NO}_x} = \int_0^t \frac{C_{\text{NO}_x}(t)}{1500} dt \quad (15.27)$$

where t is time in minutes and C_{NO_x} is the sum of NO and NO₂ concentrations (ppm).

The Fractional Lethal Dose (FLD) of irritants is calculated as

$$\text{FLD}_{\text{irr}} = \int_0^t \left(\frac{C_{\text{HCl}}(t)}{F_{\text{FLD,HCl}}} + \frac{C_{\text{HBr}}(t)}{F_{\text{FLD,HBr}}} + \frac{C_{\text{HF}}(t)}{F_{\text{FLD,HF}}} + \frac{C_{\text{SO}_2}(t)}{F_{\text{FLD,SO}_2}} + \frac{C_{\text{NO}_2}(t)}{F_{\text{FLD,NO}_2}} + \frac{C_{\text{C}_3\text{H}_4\text{O}}(t)}{F_{\text{FLD,C}_3\text{H}_4\text{O}}} + \frac{C_{\text{CH}_2\text{O}}(t)}{F_{\text{FLD,CH}_2\text{O}}} \right) dt \quad (15.28)$$

where t is time in minutes, the nominators are the instantaneous concentrations (ppm) of each irritant and the denominators the exposure doses of respective irritants predicted to be lethal to half the population. The lethal exposure doses [32] are given in the table 15.8. To include the effect of an irritant gas not listed in the table, the user should specify F_{FLD} in ppm×min using the `FLD_LETHAL_DOSE` property of the corresponding SPEC line.

Table 15.8: 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

The fraction of an incapacitating dose of low O₂ hypoxia is calculated as

$$\text{FED}_{\text{O}_2} = \int_0^t \frac{dt}{60 \exp[8.13 - 0.54(20.9 - C_{\text{O}_2}(t))]} \quad (15.29)$$

where t is time in minutes and C_{O_2} is the O₂ concentration (volume per cent). The hyperventilation factor induced by carbon dioxide is calculated as

$$\text{HV}_{\text{CO}_2} = \frac{\exp(0.1903 C_{\text{CO}_2}(t) + 2.0004)}{7.1} \quad (15.30)$$

where t is time in minutes and C_{CO_2} is the CO₂ concentration (percent).

The Fractional Irritant Concentration (FIC), also developed by Purser [32], represents the toxic effect which depends upon the immediate concentrations of irritants. The overall irritant concentration FIC is calculated as

$$\text{FIC}_{\text{irr}} = \frac{C_{\text{HCl}}(t)}{F_{\text{FIC,HCl}}} + \frac{C_{\text{HBr}}(t)}{F_{\text{FIC,HBr}}} + \frac{C_{\text{HF}}(t)}{F_{\text{FIC,HF}}} + \frac{C_{\text{SO}_2}(t)}{F_{\text{FIC,SO}_2}} + \frac{C_{\text{NO}_2}(t)}{F_{\text{FIC,NO}_2}} + \frac{C_{\text{C}_3\text{H}_4\text{O}}(t)}{F_{\text{FIC,C}_3\text{H}_4\text{O}}} + \frac{C_{\text{CH}_2\text{O}}(t)}{F_{\text{FIC,CH}_2\text{O}}} \quad (15.31)$$

where the nominators are the instantaneous concentrations of each irritant and the denominators the concentrations of respective irritants expected to cause incapacitation in half the population. The incapacitating concentrations [32] are given in the table 15.8. To include the irritant effect of a gas not listed in the table, the user should specify F_{FIC} in ppm using the `FIC_CONCENTRATION` property on the corresponding SPEC line.

Note that the spatial integration features (Section 15.10.10) cannot be used with FED output because FED makes use of the `TIME INTEGRAL` (Section 15.10.11). For the same reason, FED output is only available as a point measurement.

15.10.10 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=2.3,4.5,2.8,6.7,3.6,7.8, QUANTITY='TEMPERATURE', ID='maxT', STATISTICS='MAX' /
```

causes FDS to write out the maximum gas phase temperature over the volume bounded by XB. Note that it does not compute the maximum over the entire computational domain, just the specified volume, and this volume must lie within a single mesh. Other STATISTICS are discussed below. Note that some are appropriate for gas phase output quantities, some for solid phase, and some for both.

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. Also, be careful not to specify a volume that extends beyond a single mesh. 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.

Use the STATISTICS feature with caution because it demands that FDS evaluate the given QUANTITY in all gas or solid phase cells.

Minimum or Maximum Value

For a given gas phase scalar output quantity defined at the center of each grid cell, ϕ_{ijk} , STATISTICS='MIN' or STATISTICS='MAX' computes the minimum or maximum value, respectively

$$\min_{ijk} \phi_{ijk} \quad ; \quad \max_{ijk} \phi_{ijk} \quad (15.32)$$

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, all of which must be contained within the same mesh.

Average Value

For a given gas phase scalar output quantity defined at the center of each grid cell, ϕ_{ijk} , STATISTICS='MEAN' computes the average value,

$$\frac{1}{N} \sum_{ijk} \phi_{ijk} \quad (15.33)$$

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, all of which must be contained within the same mesh.

Volume-Weighted Mean

For a given gas phase output quantity, $\phi(x,y,z)$, STATISTICS='VOLUME MEAN' produces the discrete analog of

$$\frac{1}{V} \int \phi(x,y,z) dx dy dz \quad (15.34)$$

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, all of which must be contained within the same mesh.

Mass-Weighted Mean

For a given gas phase output quantity, $\phi(x,y,z)$, STATISTICS='MASS MEAN' produces the discrete analog of

$$\frac{\int \rho(x,y,z) \phi(x,y,z) dx dy dz}{\int \rho dx dy dz} \quad (15.35)$$

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, all of which must be contained within the same mesh.

Volume Integral

For a given gas phase output quantity, $\phi(x,y,z)$, STATISTICS='VOLUME INTEGRAL' produces the discrete analog of

$$\int \phi(x,y,z) dx dy dz \quad (15.36)$$

Note that 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. Note also that you must specify a volume to sum over via the coordinate parameters, XB, all of which must be contained within the same mesh.

Mass Integral

For a given gas phase output quantity, $\phi(x,y,z)$, STATISTICS='MASS INTEGRAL' produces the discrete analog of

$$\int \rho(x,y,z) \phi(x,y,z) dx dy dz \quad (15.37)$$

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, all of which must be contained within the same mesh.

Area Integral

For a given gas phase output quantity, $\phi(x,y,z)$, STATISTICS='AREA INTEGRAL' produces the discrete analog of

$$\int \phi(x,y,z) dA \quad (15.38)$$

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, all of which must be contained within the same mesh.

Surface Integral

For a given solid phase output quantity, ϕ , `STATISTICS='SURFACE INTEGRAL'` produces the discrete analog of

$$\int \phi dA \quad (15.39)$$

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.

Volume, Mass, and Heat Flow

The net flow of mass and energy into or out of compartments can be useful for many applications. There are several outputs that address these. All are prescribed via the device (`DEVC`) namelist group only. For example:

```
&DEVC XB=0.3,0.5,2.1,2.5,3.0,3.0, QUANTITY='MASS FLOW', ID='whatever' /
```

outputs the net integrated mass flux through the given planar area, oriented in the positive z direction, in this case. The three flows – ‘`VOLUME FLOW`’, ‘`MASS FLOW`’, and ‘`HEAT FLOW`’ are defined:

$$\begin{aligned} \dot{V} &= \int \mathbf{u} \cdot d\mathbf{S} \\ \dot{m} &= \int \rho \mathbf{u} \cdot d\mathbf{S} \\ \dot{q} &= \int c_p \rho (T - T_\infty) \mathbf{u} \cdot d\mathbf{S} \end{aligned}$$

The addition of a + or – to the `QUANTITY` names yields the integral of the flow in the positive or negative direction only. In other words, if you want to know the mass flow out of a compartment, use ‘`MASS FLOW +`’ or ‘`MASS FLOW -`’, depending on the orientation of the door.

The quantities ‘`MASS FLOW`’ and ‘`HEAT FLOW`’ should not be applied at a solid boundary.

15.10.11 Temporally-Integrated Outputs

In addition to the spatial statistics, a time integral of an `DEVC` output can be computed by specifying `STATISTICS = 'TIME INTEGRAL'` on the `DEVC` line. This produces a discrete analog of

$$\int_{t_0}^t \phi(\tau) d\tau \quad (15.40)$$

Note that the spatial and time integrals can not be used simultaneously.

15.10.12 Wind and the Pressure Coefficient

In the field of wind engineering, a commonly used quantity is known as the `PRESSURE_COEFFICIENT`:

$$C_p = \frac{p - p_\infty}{\frac{1}{2} \rho_\infty U^2} \quad (15.41)$$

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='whatever' /
&DEVC ID='pressure tap', XYZ=..., IOR=2, QUANTITY='PRESSURE COEFFICIENT', PROP_ID='whatever' /
&PROP ID='whatever', 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.

15.10.13 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 the Werner-Wengle wall model [33] for smooth walls and a rough wall log law for rough walls [10]. For the wall models to function properly, the grid resolution near the wall should fall within a certain range of y^+ , the nondimensional distance from the wall expressed in viscous units. To check this, the user may add a boundary file output as follows:

```
&BNDF QUANTITY='YPLUS' /
```

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 local viscous length scale, δ_v [10]:

$$y^+ = \frac{1}{2} \frac{\delta n}{\delta_v}; \quad \delta_v = \frac{\mu/\rho}{u_\tau}; \quad u_\tau = \sqrt{\tau_w/\rho}, \quad (15.42)$$

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 streamwise velocity component near the wall); the quantity u_τ is the *friction velocity*.

Wall functions for LES are still under development, but as a general guideline it is recommended that the first grid cell fall within the log layer: 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 $y^+ = O(100)$.

15.10.14 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=.TRUE. /
```

15.10.15 Gas Velocity

The gas velocity components, u , v , and w , can be output as slice (SLCF), point device (DEVC), isosurface (ISOFF), 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.

15.10.16 Enthalpy

There are several outputs that report the enthalpy of the gas mixture. First, the 'SPECIFIC ENTHALPY' and the 'SPECIFIC SENSIBLE ENTHALPY' are defined:

$$h = h^0 + \int_0^{T'} c_p dT' \quad ; \quad h_s = \int_0^{T'} c_p dT' \quad (15.43)$$

Both have units of kJ/kg. The quantities 'ENTHALPY' and SPECIFIC ENTHALPY' are $H = \rho h$ and $H_s = \rho h_s$, respectively, in units of kJ/m³.

15.10.17 Computer Performance

There are a variety of ways to test the performance of your computer in running an FDS simulation. In no particular order, here is a list inputs and/or outputs to help:

DEBUG If set to .TRUE. on the DUMP line, this parameter will cause FDS to print out debugging information concerning the progress of an MPI calculation.

TIMING If set to .TRUE. on the DUMP line, this parameter will cause FDS to print out timing information concerning the data exchanges in an MPI calculation.

VELOCITY_ERROR_FILE If set to .TRUE. 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.

15.10.18 Output File Precision

There are several different output files that have the format of a comma-delimited spreadsheet (.csv). 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.

15.10.19 A Posteriori Mesh Quality Metrics

The quality of a particular simulation is most directly tied to grid resolution. Three slice file output quantities are suggested here for measuring errors in the velocity and scalar fields:

1. A model for the fraction of unresolved kinetic energy called the *measure of turbulence resolution* (similar to what is often called the 'Pope criterion' [34]), MTR
2. A model for the fraction of unresolved scalar energy fluctuations called *measure of scalar resolution* [35], MSR
3. A *wavelet-based error measure* [36], WEM

Examples:

```
&SLCF PBY=0, QUANTITY='TURBULENCE RESOLUTION' /
&SLCF PBY=0, QUANTITY='SCALAR RESOLUTION', QUANTITY2='MASS FRACTION', SPEC_ID='HELIUM' /
&SLCF PBY=0, QUANTITY='WAVELET ERROR', QUANTITY2='HRRPUV', CELL_CENTERED=.TRUE. /
```

Note that an additional scalar `QUANTITY2` is required for MSR and WEM. `QUANTITY2` may be any output quantity appropriate for SLCF. Also, `CELL_CENTERED` is optional for any of the three metrics.

Measure of Turbulence Resolution

In FDS, the user may output a scalar quantity which we refer to as the *measure of turbulence resolution*, defined locally as

$$\text{MTR}(\mathbf{x}, t) = \frac{k_{sgs}}{k_{res} + k_{sgs}} \quad (15.44)$$

where

$$k_{res} = \frac{1}{2} \tilde{u}_i \tilde{u}_i \quad (15.45)$$

$$k_{sgs} = \frac{1}{2} (\tilde{u}_i - \hat{\tilde{u}}_i)(\tilde{u}_i - \hat{\tilde{u}}_i) \quad (15.46)$$

Here, \tilde{u}_i is the resolved LES velocity and $\hat{\tilde{u}}_i$ is test filtered at a scale 2Δ where Δ is the LES filter width (in FDS, $\Delta = \delta x$). The model for the SGS fluctuations is taken from scale similarity [37]. Cross-term energy is ignored. The basic idea is to provide the user with an approximation to the Pope criterion [34], M , which is easily accessible in Smokeview (the FDS visualization tool). In Smokeview, the user may readily time average MTR in a specified plane. The time average of MTR is a reasonable estimate of M . The measure falls within the range [0,1], with 0 indicating perfect resolution and 1 indicating poor resolution. The concept is illustrated in Figure 15.8. 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.

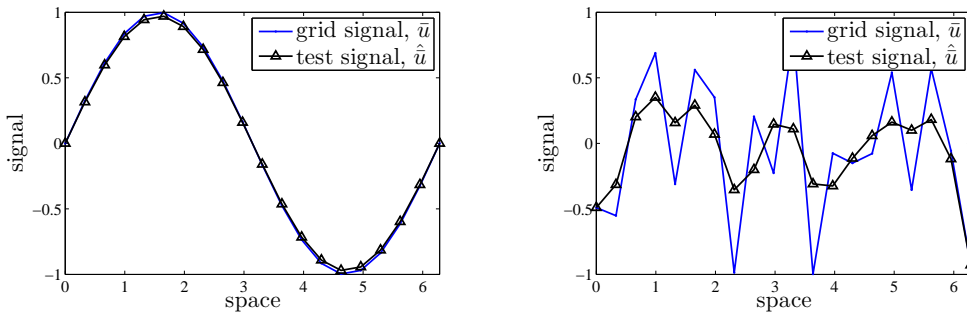


Figure 15.8: (Left) Resolved signal, MTR is small. (Right) Unresolved signal, MTR is close to unity.

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 [36], maintaining mean values of MTR near 0.2 indeed provides satisfactory results (simulation results within experimental error bounds) for mean velocities and species concentrations in nonreacting, buoyant plumes.

Measure of Scalar Resolution

The *measure of scalar resolution* is defined locally as

$$\text{MSR}(\mathbf{x}, t) = \frac{T_{sgs}}{T_{res} + T_{sgs}} \quad (15.47)$$

where

$$T_{res} = \tilde{\phi}^2 \quad (15.48)$$

$$T_{sgs} = (\tilde{\phi} - \hat{\phi})^2 \quad (15.49)$$

Here again the model for the SGS scalar energy fluctuations is taken from scale similarity [37]. The field $\hat{\phi}$ is test filtered at a scale 2Δ . The cross-term energy (i.e. $\langle 2\tilde{\phi}\phi' \rangle$, where $\phi' = \phi - \tilde{\phi}$) is ignored, but this does not affect the bounds of the measure. Further, it can be shown that this term is small if sufficient resolution is used. There is evidence to suggest that the requirements for scalar resolution may be somewhat more stringent than for the velocity field [35]. Therefore, currently the best advice is to keep the mean value of MSR less than 0.2.

Wavelet Error Measure

We begin by providing background on the simple Haar wavelet [38]. For a thorough and more sophisticated review of wavelet methods, the reader is referred to Schneider and Vasilyev [39].

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

$$\Phi_{[r_1, r_2]} = \begin{cases} 1 & \text{if } r_1 \leq r < r_2 \\ 0 & \text{otherwise} \end{cases} \quad (15.50)$$

the simplest possible reconstruction of the signal is the step function approximation

$$f(r) \approx \sum_j s_j \Phi_{[r_j, r_j+h]}(r) \quad (15.51)$$

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

$$f_{[r_j, r_j+2h]}(r) = \underbrace{\frac{s_j + s_{j+1}}{2}}_a \Phi_{[r_j, r_j+2h]}(r) + \underbrace{\frac{s_j - s_{j+1}}{2}}_c \Psi_{[r_j, r_j+2h]}(r) \quad (15.52)$$

where a is as the *average* coefficient and c is as the *wavelet* coefficient. The Haar *mother wavelet* (Figure 15.9) is identified as

$$\Psi_{[r_1, r_2]}(r) = \begin{cases} 1 & \text{if } r_1 \leq r < \frac{1}{2}(r_1 + r_2) \\ -1 & \text{if } \frac{1}{2}(r_1 + r_2) \leq r < r_2 \end{cases} \quad (15.53)$$

The decomposition of the signal shown in (15.52) 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 (15.52) 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.

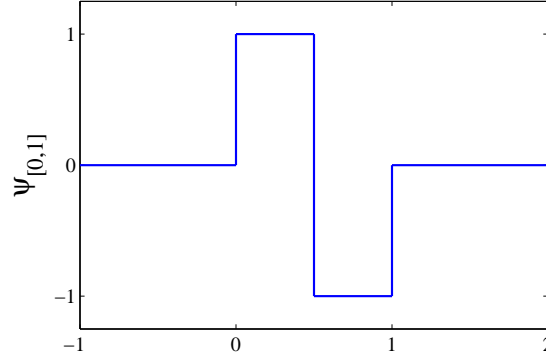


Figure 15.9: Haar mother wavelet on the interval $[0,1]$.

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 15.10. 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:

$$\text{WEM}(\mathbf{x}, t) = \max_{x,y,z} \left(\frac{|c_{11} + c_{12}| - |c_{21}|}{|a_{21}|} \right) \quad (15.54)$$

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.

Looking now at the two plots on the bottom of Figure 15.10, the signal on the left, which may indicate spurious oscillations or unresolved turbulent motion, leads to $\text{WEM} = 2$ (note that this limit differs from the upper bound of unity for MTR and MSR). 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 [36], the time average of WEM was reported for LES of a nonreacting buoyant plume at three grid resolutions. From this study, the best advice currently is to maintain average values of WEM less than 0.5.

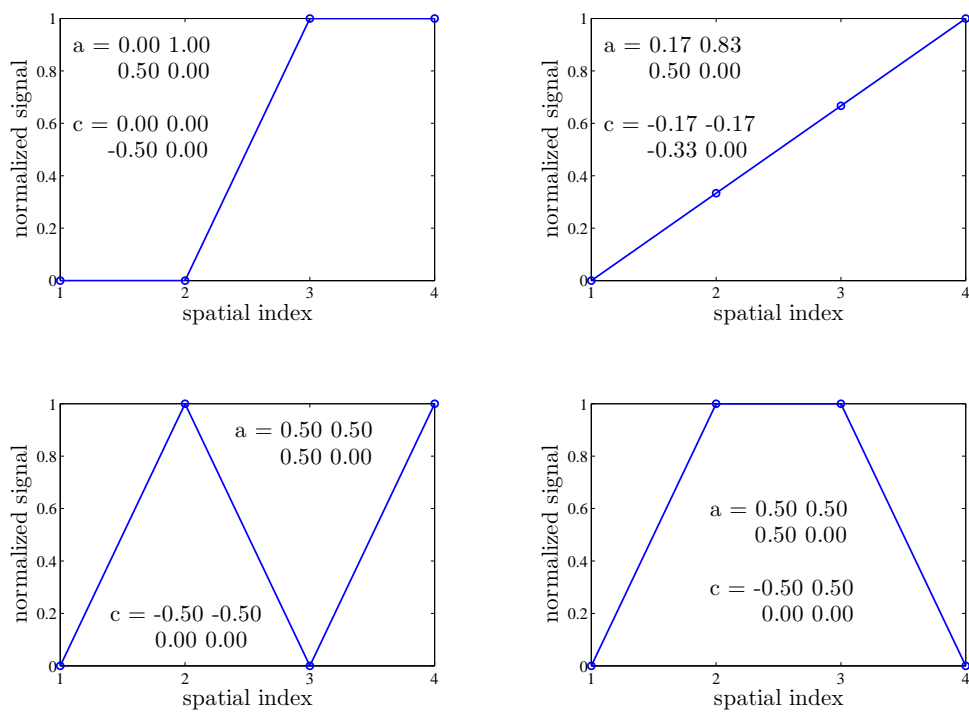


Figure 15.10: Averages and coefficients for local Haar wavelet transforms on four typical signals.

15.11 Extracting Numbers from the Output Data Files

Often it is desired to present results of calculations in some form other than those offered by Smokeview. In this case, there is a short Fortran 90 program called **fds2ascii.f90**, with a PC compiled version called **fds2ascii.exe**. 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
  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_fds2ascii.csv
  Writing to file...      bucket_test_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 upwards (+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.

15.12 Summary of Frequently-Used Output Quantities

Table 15.9, spread over the following pages, summarizes the various Output Quantities. The column “File Type” lists the allowed output files for the quantities. “B” is for Boundary (BNDF), “D” is for Device (DEVC), “I” is for Iso-surface (ISOF), “P” is for Plot3D, “PA” for PArticle (PART), “S” is for Slice (SLCF). Be careful when specifying complicated quantities for Iso-surface or Plot3D files, as it requires computation in every gas phase cell.

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 and/or SOOT_YIELD are specified on the REAC line, then 'CARBON MONOXIDE' and 'SOOT' 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',  
      STATISTICS='AREA INTEGRAL' /
```

Here, 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. Also note that the use of underscores in output quantity names has been eliminated – just remember that all output quantity names ought to be in single quotes.

Table 15.9: Summary of frequently used output quantities.

QUANTITY	Symbol	Units	File Type
ABSORPTION COEFFICIENT	κ (Section 13.1)	1/m	D,I,P,S
ACTUATED SPRINKLERS	Number of actuated sprinklers		D
ADIABATIC SURFACE TEMPERATURE	Section 15.10.7	$^{\circ}\text{C}$	B,D
AMPUA**	Section 15.10.6	kg/m^2	B,D
ASPIRATION	Section 15.3.7	%	D
BACK WALL TEMPERATURE	Section 15.9.2	$^{\circ}\text{C}$	B,D
BURNING RATE	\dot{m}_f''	$\text{kg/m}^2/\text{s}$	B,D
CHAMBER OBSCURATION	Section 15.3.5	%/m	D
CONDUCTIVITY	k	W/m/k	D,I,P,S
CONTROL	Section 15.5		D
CONVECTIVE HEAT FLUX	\dot{q}_c'' (Section 15.10.5)	kW/m^2	B,D
CPUA**	Section 15.10.6	kW/m^2	B,D
CPU TIME	Elapsed CPU time	s	D
DENSITY	ρ or ρY_{α} with SPEC_ID	kg/m^3	D,I,P,S
DIVERGENCE	$\nabla \cdot \mathbf{u}$	s^{-1}	D,I,P,S
ENTHALPY	Section 15.10.16	kJ/m^3	D,I,P,S
FED	Section 15.10.9		D
FIC	Section 15.10.9		D,S
GAUGE HEAT FLUX	Section 15.10.5	kW/m^2	B,D
H	$H = \mathbf{u} ^2/2 + \tilde{p}/\rho_0$	$(\text{m/s})^2$	D,I,P,S
HEAT FLOW	Section 15.10.10	kW	D
NET HEAT FLUX	Section 15.10.5	kW/m^2	B,D
HRR	$\int \dot{q}''' dV$	kW	D
HRRPUV	\dot{q}'''	kW/m^3	D,I,P,S
HRRPUA	\dot{q}''	kW/m^2	D
INCIDENT HEAT FLUX	Section 15.10.5	kW/m^2	B,D
INSIDE WALL TEMPERATURE	Section 15.9.2	$^{\circ}\text{C}$	D
ITERATION	Number of time steps		D
LAYER HEIGHT	Section 15.10.3	m	D
LINK TEMPERATURE	Section 15.3.4	$^{\circ}\text{C}$	D
LOWER TEMPERATURE	Section 15.10.3	$^{\circ}\text{C}$	D
MASS FLOW	Section 15.10.10	kg/s	D
MASS FLUX*	Mass flux at solid surface	$\text{kg/m}^2/\text{s}$	B,D
MASS FLUX X*	$\rho u Y_{\alpha}$	$\text{kg/m}^2/\text{s}$	D,I,P,S
MASS FLUX Y*	$\rho v Y_{\alpha}$	$\text{kg/m}^2/\text{s}$	D,I,P,S
MASS FLUX Z*	$\rho w Y_{\alpha}$	$\text{kg/m}^2/\text{s}$	D,I,P,S
MASS FRACTION*	Y_{α}	kg/kg	D,I,P,S
MIXTURE FRACTION	Z	kg/kg	D,I,P,S
MPUA**	Section 15.10.6	kg/m^2	B,D
MPUV**	Section 15.10.6	kg/m^3	D,P,S
NORMAL VELOCITY	Wall normal velocity	m/s	D,B
OPEN NOZZLES	Number of open nozzles		D
OPTICAL DENSITY	$K/2.3$ (Section 15.10.2)	1/m	D,I,P,S

Table 15.9: Summary of frequently used output quantities (continued).

QUANTITY	Symbol	Units	File Type
EXTINCTION COEFFICIENT	K (Section 15.10.2)	1/m	D,I,P,S
PATH OBSCURATION	Section 15.3.6	%	D
PARTICLE AGE	t_d	s	PA
PARTICLE DIAMETER	$2r_d$	μm	PA
PARTICLE FLUX X^{**}	Section 15.10.6	$\text{kg/m}^2/\text{s}$	P,S
PARTICLE FLUX Y^{**}	Section 15.10.6	$\text{kg/m}^2/\text{s}$	P,S
PARTICLE FLUX Z^{**}	Section 15.10.6	$\text{kg/m}^2/\text{s}$	P,S
PARTICLE MASS	m_d	kg	PA
PARTICLE TEMPERATURE	T_d	$^{\circ}\text{C}$	PA
PARTICLE VELOCITY	$ \mathbf{u}_d $	m/s	PA
PRESSURE	Perturbation pressure, \tilde{p}	Pa	D,I,P,S
PRESSURE COEFFICIENT	C_p (Section 15.10.12)		B,D
PRESSURE ZONE	Section 9.3		D,S
RADIATIVE HEAT FLUX	Section 15.10.5	kW/m^2	B,D
RADIOMETER	Section 15.10.5	kW/m^2	B,D
RELATIVE HUMIDITY	Relative humidity	%	D,I,P,S
SCALAR RESOLUTION*****	Section 15.10.19		S
SENSIBLE ENTHALPY	Section 15.10.16	kJ/m^3	D,I,P,S
SOOT VOLUME FRACTION	$\rho Y_s(Z)/\rho_s$	mol/mol	D,I,P,S
SPECIFIC ENTHALPY	Section 15.10.16	kJ/kg	D,I,P,S
SPECIFIC SENSIBLE ENTHALPY	Section 15.10.16	kJ/kg	D,I,P,S
SPECIFIC HEAT	c_p	kJ/kg/K	D,I,P,S
SPRINKLER LINK TEMPERATURE	Section 15.3.1	$^{\circ}\text{C}$	D
SOLID DENSITY	Section 15.10.8	kg/m^3	D
SURFACE DENSITY	Section 15.10.8	kg/m^2	B,D
TEMPERATURE	T (Section 15.10.4)	$^{\circ}\text{C}$	D,I,P,S
THERMOCOUPLE	T_{TC} (Section 15.10.4)	$^{\circ}\text{C}$	D
TIME	t (Section 15.1)	s	D
TIME STEP	δt , Numerical time step	s	D
TURBULENCE RESOLUTION	Section 15.10.19		D,S
U-VELOCITY	u	m/s	D,I,P,S
V-VELOCITY	v	m/s	D,I,P,S
W-VELOCITY	w	m/s	D,I,P,S
UPPER TEMPERATURE	Section 15.10.3	$^{\circ}\text{C}$	D
VELOCITY***	$\sqrt{u^2 + v^2 + w^2}$	m/s	D,I,P,S
VISCOSITY	μ	kg/m/s	D,I,P,S
VISIBILITY	$S = C/K$ (Section 15.10.2)	m	D,I,P,S
VOLUME FLOW	Section 15.10.10	m^3/s	D
VOLUME FRACTION****	X_{α}	mol/mol	D,I,P,S
WALL CLOCK TIME	Elapsed wall clock time	s	D
WALL TEMPERATURE	T_w	$^{\circ}\text{C}$	B,D
WALL THICKNESS	Section 15.10.8	m	B,D
WAVELET ERROR*****	Section 15.10.19		S

Table 15.9: Summary of frequently used output quantities (continued).

QUANTITY	Symbol	Units	File Type
YPLUS	Section 15.10.13		B

- * Quantity requires the specification of a gas species using `SPEC_ID`.
- ** Quantity requires the specification of a particle name using `PART_ID`.
- *** 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.
- **** Quantity requires the specification of a gas species using `SPEC_ID`.
Do not use for `MIXTURE FRACTION`.
- ***** Quantity requires specification of an additional scalar using `QUANTITY2`.

15.13 Summary of Infrequently-Used Output Quantities

Table 15.10 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 [1].

Table 15.10: Summary of *infrequently* used output quantities.

QUANTITY	Symbol	Units	File Type
ADD	Average Droplet Diameter	μm	D,I,P,S
ADT	Average Droplet Temperature	$^{\circ}\text{C}$	D,I,P,S
C_SMAG	Smagorinsky coefficient		D,I,P,S
CABLE TEMPERATURE	Inner temperature of cable	$^{\circ}\text{C}$	D
EMISSION	Surface emissivity (usually constant)		B,D
F_X, F_Y, F_Z	Momentum terms, F_x, F_y, F_z	m/s^2	D,I,P,S
GAS TEMPERATURE	Gas Temperature near wall	$^{\circ}\text{C}$	B,D
HEAT TRANSFER COEFFICIENT	Convective heat transfer	$\text{W/m}^2/\text{K}$	B,D
HRRPUL	$\int \dot{q}''' dx dy$	kW/m	D
INTEGRATED INTENSITY	$U = \int I ds$	kW/m^2	D,I,P,S
KINETIC ENERGY	$(u^2 + v^2 + w^2)/2$	$(\text{m/s})^2$	D,I,P,S
MAXIMUM VELOCITY ERROR	Section 6.6		D
MIXING TIME	Combustion mixing time	s	D,I,P,S
PARTICLE PHASE	Orientation of droplet		PA
PDPA	Droplet diagnostics		D
PRESSURE ITERATIONS	No. pressure iterations		D
RADIATION LOSS	$\nabla \cdot \mathbf{q}_r''$	kW/m^3	D,I,P,S
STRAIN RATE X	$\partial w / \partial y + \partial v / \partial z$	1/s	D,I,P,S
STRAIN RATE Y	$\partial u / \partial z + \partial w / \partial x$	1/s	D,I,P,S
STRAIN RATE Z	$\partial v / \partial x + \partial u / \partial y$	1/s	D,I,P,S
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
PARTICLE RADIATION LOSS	$\nabla \cdot \mathbf{q}_r''$ due to Lagrangian particles	kW/m^3	D,I,P,S

15.14 Summary of HVAC Output Quantities

Table 15.11 summarizes the various Output Quantities for HVAC systems. Quantities for a duct require the specification of a `DUCT_ID`, and quantities for a node require the specification of a `NODE_ID`. Mass and volume fraction outputs also require the specification of a `SPEC_ID`.

Table 15.11: Summary of HVAC output quantities.

QUANTITY	Symbol	Units
AIRCOIL HEAT EXCHANGE	Heat exchange rate for an aircoil	kW
DUCT DENSITY	Density of the flow in a duct	kg/m ³
DUCT MASS FLOW	Mass flow in a duct	kg/s
DUCT MASS FRACTION	Mass fraction of a species in a duct	kg/kg
DUCT TEMPERATURE	Temperature of the flow in a duct	°C
DUCT VELOCITY	Velocity of a duct	m/s
DUCT VOLUME FLOW	Volumetric flow in a duct	m ³ /s
DUCT VOLUME FRACTION	Volume fraction of a species in a duct	mol/mol
FILTER LOADING	Loading of a species in a filter	kg
FILTER LOSS	Flow loss through a filter	
NODE DENSITY	Density of the flow through a node	kg/m ³
NODE MASS FRACTION	Mass fraction of a species in a node	kg/kg
NODE PRESSURE	Pressure of a node	Pa
NODE PRESSURE DIFFERENCE	Pressure difference between two nodes	Pa
NODE TEMPERATURE	Temperature of the flow through a node	°C
NODE VOLUME FRACTION	Volume fraction of a species in a node	mol/mol

Chapter 16

Alphabetical List of Input Parameters

This Appendix 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 Appendix. 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.

16.1 BNDF (Boundary File Parameters)

Table 16.1: For more information see Section 15.9.4.

BNDF (Boundary File Parameters)				
CELL_CENTERED	Logical	Section 15.9.4		.FALSE.
PART_ID	Character	Section 15.12		
PROP_ID	Character	Section 15.9.4		
QUANTITY	Character	Section 15.12		
SPEC_ID	Character	Section 15.12		

16.2 CLIP (MIN/MAX Clipping Parameters)

Table 16.2: For more information see Section 6.7.

CLIP (Specified Upper and Lower Limits)				
MAXIMUM_DENSITY	Real	Section 6.7	kg/m ³	
MAXIMUM_TEMPERATURE	Real	Section 6.7	°C	
MINIMUM_DENSITY	Real	Section 6.7	kg/m ³	
MINIMUM_TEMPERATURE	Real	Section 6.7	°C	

16.3 CSVF (Comma Delimited Output Files)

Table 16.3: For more information see Section 6.4.5.

CSVF (Comma Delimited Output Files)				
UVWFILE	Character	Section 6.4.5		

16.4 CTRL (Control Function Parameters)

Table 16.4: For more information see Section 15.5.

CTRL (Control Function Parameters)				
CONSTANT	Real	Section 15.5.6		
DELAY	Real	Section 15.5.9	s	0.
DIFFERENTIAL_GAIN	Real	Section 15.5.7		0.
EVACUATION	Logical	Reference [40]		.FALSE.
FUNCTION_TYPE	Character	Section 15.4		
ID	Character	Section 15.5		
INITIAL_STATE	Logical	Section 15.4		.FALSE.

Table 16.4: Continued

CTRL (Control Function Parameters)				
INPUT_ID	Char. Array	Section 15.5		
INTEGRAL_GAIN	Real	Section 15.5.7		0.
LATCH	Logical	Section 15.4		.TRUE.
N	Integer	Section 15.5		1
ON_BOUND	Character	Section 15.5.3		LOWER
PROPORTIONAL_GAIN	Real	Section 15.5.7		0.
RAMP_ID	Character	Section 15.5.5		
SETPOINT(2)	Real	Section 15.4		
TARGET_VALUE	Real	Section 15.5.7		0.
TRIP_DIRECTION	Integer	Section 15.4		1

16.5 DEVC (Device Parameters)

Table 16.5: For more information see Section 15.1.

DEVC (Device Parameters)				
BYPASS_FLOWRATE	Real	Section 15.3.7	kg/s	0
CONVERSION_FACTOR	Real	Section 15.2		1
CTRL_ID	Character	Section 15.6.1		
DELAY	Real	Section 15.3.7	s	0
DEPTH	Real	Section 15.10.8	m	0
DEVC_ID	Character	Sections 15.3.7 and 15.6.1		
DRY	Logical	Section 15.10.14		.FALSE.
DUCT_ID	Character	Section 9.2		
EVACUATION	Logical	Reference [40]		.FALSE.
FLOWRATE	Real	Section 15.3.7	kg/s	0
HIDE_COORDINATES	Logical	Section 15.9.1		.FALSE.
ID	Character	Section 15.1		
INITIAL_STATE	Logical	Section 15.4		.FALSE.
INIT_ID	Character	Section 14.4		
IOR	Integer	Section 15.1		
LATCH	Logical	Section 15.4		.TRUE.
MATL_ID	Character	Section 15.10.8		
NODE_ID	Character(2)	Section 9.2		
NO_UPDATE_DEVC_ID	Character	Section 15.6.2		
NO_UPDATE_CTRL_ID	Character	Section 15.6.2		
ORIENTATION	Real Triplet	Section 15.1		0,0,-1
OUTPUT	Logical	Section 15.2		.TRUE.
PART_ID	Character	Section 15.12		
PIPE_INDEX	Integer	Section 15.3.1		1
POINTS	Integer	Section 15.9.1		1
PROP_ID	Character	Section 15.1		

Table 16.5: Continued

DEVC (Device Parameters)				
QUANTITY	Character	Section 15.1		
RELATIVE	Logical	Section 15.2		.FALSE.
ROTATION	Real	Section 15.1	deg.	0
SETPOINT	Real	Section 15.4		
SMOOTHING_FACTOR	Real	Section 15.4		0
SPEC_ID	Character	Section 15.12		
STATISTICS	Character	Section 15.10.10		
SURF_ID	Character	Section 15.10.10		
TIME_AVERAGED	Logical	Section 15.2		.TRUE.
TRIP_DIRECTION	Integer	Section 15.4		1
UNITS	Character	Section 15.2		
VELO_INDEX	Integer	Section 15.10.15		0
XB (6)	Real Sextuplet	Section 15.10.10	m	
XYZ (3)	Real Triplet	Section 15.1	m	
X_ID	Character	Section 15.9.1		ID-x
Y_ID	Character	Section 15.9.1		ID-y
Z_ID	Character	Section 15.9.1		ID-z

16.6 DUMP (Output Parameters)

Table 16.6: For more information see Section 15.8.

DUMP (Output Parameters)				
COLUMN_DUMP_LIMIT	Logical	Section 15.2		.TRUE.
CTRL_COLUMN_LIMIT	Integer	Section 15.2		254
DEBUG	Logical	Section 15.10.17		.FALSE.
DEVC_COLUMN_LIMIT	Integer	Section 15.2		254
DT_BNDF	Real	Section 15.8	s	$2\Delta t / \text{NFRAMES}$
DT_CTRL	Real	Section 15.8	s	$\Delta t / \text{NFRAMES}$
DT_DEVC	Real	Section 15.8	s	$\Delta t / \text{NFRAMES}$
DT_DEVC_LINE	Real	Section 15.9.1		1
DT_FLUSH	Real	Section 15.8	s	$\Delta t / \text{NFRAMES}$
DT_HRR	Real	Section 15.8	s	$\Delta t / \text{NFRAMES}$
DT_ISOF	Real	Section 15.8	s	$\Delta t / \text{NFRAMES}$
DT_MASS	Real	Section 15.8	s	$\Delta t / \text{NFRAMES}$
DT_PART	Real	Section 15.8	s	$\Delta t / \text{NFRAMES}$
DT_PL3D	Real	Section 15.8	s	$\Delta t / 5$
DT_PROF	Real	Section 15.8	s	$\Delta t / \text{NFRAMES}$
DT_RESTART	Real	Section 15.8	s	1000000.
DT_SL3D	Real	Section 15.8	s	$\Delta t / 5$
DT_SLCF	Real	Section 15.8	s	$\Delta t / \text{NFRAMES}$
FLUSH_FILE_BUFFERS	Logical	Section 15.8		.TRUE.

Table 16.6: Continued

DUMP (Output Parameters)				
MASS_FILE	Logical	Section 15.8		.FALSE.
MAXIMUM_PARTICLES	Integer	Section 15.8		500000
NFRAMES	Integer	Section 15.8		1000
PLOT3D_PART_ID (5)	Char. Quint	Section 15.9.6		
PLOT3D_QUANTITY (5)	Char. Quint	Section 15.9.6		
PLOT3D_SPEC_ID (5)	Char. Quint	Section 15.9.6		
PLOT3D_VELO_INDEX	Int. Quint	Section 15.10.15		0
RENDER_FILE	Character	Reference [2]		
SIG_FIGS	Integer	Section 15.10.18		8
SIG_FIGS_EXP	Integer	Section 15.10.18		3
SMOKE3D	Logical	Section 15.9.7		.TRUE.
SMOKE3D_QUANTITY	Character	Section 15.9.7		
SMOKE3D_SPEC_ID	Character	Section 15.9.7		
STATUS_FILES	Logical	Section 15.8		.FALSE.
TIMING	Logical	Section 15.10.17		.FALSE.
UVW_TIMER	Real Vector (10)	Section 6.4.5	s	
VELOCITY_ERROR_FILE	Logical	Section 15.10.17		.FALSE.
WRITE_XYZ	Logical	Section 15.9.6		.FALSE.

$$\Delta t = T_{\text{END}} - T_{\text{BEGIN}}$$

16.7 HEAD (Header Parameters)

Table 16.7: For more information see Section 6.1.

HEAD (Header Parameters)				
CHID	Character	Section 6.1		'output'
TITLE	Character	Section 15.9.6		

16.8 HOLE (Obstruction Cutout Parameters)

Table 16.8: For more information see Section 7.3.

HOLE (Obstruction Cutout Parameters)				
COLOR	Character	Section 7.5		
CTRL_ID	Character	Section 7.3		
DEVC_ID	Character	Section 7.3		
EVACUATION	Logical	Reference [40]		
MESH_ID	Character	Reference [40]		
MULT_ID	Character	Section 7.2.2		

Table 16.8: Continued

HOLE (Obstruction Cutout Parameters)				
RGB (3)	Integer Triplet	Section 7.5		
TRANSPARENCY	Real	Section 7.3		
XB (6)	Real Sextuplet	Section 7.2.2	m	

16.9 HVAC (HVAC System Definition)

Table 16.9: For more information see Section 9.2.

HVAC (HVAC System Definition)				
AIRCOIL_ID	Character	Section 9.2.1		
AMBIENT	Logical	Section 9.2.2		.FALSE.
AREA	Real	Section 9.2.1	m ²	
CLEAN_LOSS	Real	Section 9.2.4		
COOLANT_CP	Real	Section 9.2.5	kJ/kg/K	
COOLANT_MDOT	Real	Section 9.2.5	kg/s	
COOLANT_TEMPERATURE	Real	Section 9.2.5	°C	
CTRL_ID	Character	Sections 9.2.1, 9.2.3, and 9.2.4		
DAMPER	Logical	Section 9.2.1		.FALSE.
DEVC_ID	Character	Sections 9.2.1, 9.2.3, and 9.2.4		
DIAMETER	Real	Section 9.2.1	m	
DUCT_ID	Character Array	Section 9.2.2		
EFFICIENCY	Real Array	Sections 9.2.4 and 9.2.5		1.0
FAN_ID	Character	Section 9.2.1		
FILTER_ID	Character	Section 9.2.2		
FIXED_Q	Real	Section 9.2.5	kW	
ID	Character	Section 9.2		
LENGTH	Real	Section 9.2.1	m	
LOADING	Real Array	Section 9.2.4	kg	0.0
LOADING_MULTIPLIER	Real Array	Section 9.2.4	kg	0.0
LOSS	Real Array	Sections 9.2.1 – 9.2.4		0.0
MAX_FLOW	Real	Section 9.2.3	m ³ /s	
MAX_PRESSURE	Real	Section 9.2.3	Pa	
NODE_ID	Character Doublet	Section 9.2.1		
RAMP_ID	Character	Sections 9.2.1, 9.2.4, and 9.2.3		
REVERSE	Logical	Section 9.2.1		.FALSE.
ROUGHNESS	Real	Section 9.2.1	m	0.0
SPEC_ID	Character	Section 9.2.4		
TAU_FAN	Real	Section 9.2.3	s	1.0
TAU_VF	Real	Section 9.2.1	s	1.0
TYPE_ID	Character	Section 9.2		
VENT_ID	Character	Section 9.2.2		
VOLUME_FLOW	Real	Section 9.2.1 and 9.2.3	m ³ /s	

Table 16.9: Continued

HVAC (HVAC System Definition)				
XYZ	Real Triplet	Section 9.2.2	m	

16.10 INIT (Initial Conditions)

Table 16.10: For more information see Section 6.5.

INIT (Initial Conditions)				
CELL_CENTERED	Logical	Section 14.2.3		.FALSE.
CTRL_ID	Character	Section 14.2.3		
DENSITY	Real	Section 6.5	kg/m ³	Ambient
DEVC_ID	Character	Section 14.2.3		
DT_INSERT	Real	Section 14.2.3	s	
DX	Real	Section 14.2.3	m	0.
DY	Real	Section 14.2.3	m	0.
DZ	Real	Section 14.2.3	m	0.
HRRPUV	Real	Section 6.5	kW/m ³	
ID	Character	Section 14.4		
MASS_FRACTION(N)	Real Array	Section 6.5	kg/kg	Ambient
MASS_PER_TIME	Real	Section 14.2.3	kg/s	
MASS_PER_VOLUME	Real	Section 14.2.3	kg/m ³	1
MULT_ID	Character	Section 7.2.2		
N_PARTICLES	Integer	Section 14.2.3		0
N_PARTICLES_PER_CELL	Integer	Section 14.2.3		0
PART_ID	Character	Section 14.2.3		
SHAPE	Character	Section 14.2.3		' BLOCK'
SPEC_ID(N)	Character Array	Section 6.5		
TEMPERATURE	Real	Section 6.5	°C	TMPA
UVW (3)	Real Triplet	Section 14.2.3	m/s	0.
XB (6)	Real Sextuplet	Section 6.5	m	
XYZ (3)	Real Triplet	Section 14.2.3	m	

16.11 ISOF (Isosurface Parameters)

Table 16.11: For more information see Section 15.9.5.

ISOF (Isosurface Parameters)				
COLOR_SPEC_ID	Character	Section 15.9.5		
QUANTITY	Character	Section 15.9.5		
REDUCE_TRIANGLES	Integer	Reference [2]		1
SPEC_ID	Character	Section 15.9.5		

Table 16.11: Continued

ISO (Isosurface Parameters)				
VALUE (I)	Real Array	Section 15.9.5		
VELO_INDEX	Integer	Section 15.10.15		0

16.12 MATL (Material Properties)

Table 16.12: For more information see Section 8.3.

MATL (Material Properties)				
A (:)	Real array	Section 8.4.4	1/s	
ABSORPTION_COEFFICIENT	Real	Section 8.3.2	1/m	50000.
BOILING_TEMPERATURE	Real	Section 8.4.5	°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 8.4.4	kJ/kmol	
EMISSION	Real	Section 8.3.2		0.9
HEATING_RATE (:)	Real array	Section 8.4.4	°C/min	5.
HEAT_OF_COMBUSTION (:)	Real array	Section 8.4.4	kJ/kg	
HEAT_OF_REACTION (:)	Real array	Section 8.4.4	kJ/kg	0.
ID	Character	Section 8.1		
INITIAL_VAPOR_FLUX	Real	Section 8.4.5	m/s	0.0005
MATL_ID (: , :)	Character	Section 8.4.4		
NU_MATL (: , :)	Real array	Section 8.4.4	kg/kg	0.
NU_SPEC (: , :)	Real array	Section 8.4.4	kg/kg	0.
N_REACTIONS	Character	Section 8.4.4		0
N_S (:)	Real array	Section 8.4.4		1.
N_T (:)	Real array	Section 8.4.4		0.
PCR (:)	Logical array	Section 8.4.4		.FALSE.
PYROLYSIS_RANGE (:)	Real array	Section 8.4.4	°C	80.
REFERENCE_RATE (:)	Real array	Section 8.4.4	1/s	
REFERENCE_TEMPERATURE (:)	Real array	Section 8.4.4	°C	
SPECIFIC_HEAT	Real	Section 8.3.2	kJ/kg/K	0.
SPECIFIC_HEAT_RAMP	Character	Section 8.3.2		
SPEC_ID (: , :)	Character	Section 8.4.4		
THRESHOLD_SIGN (:)	Real array	Section 8.4.4		1.0
THRESHOLD_TEMPERATURE (:)	Real array	Section 8.4.4	°C	-273.15

16.13 MESH (Mesh Parameters)

Table 16.13: For more information see Section 6.3.

MESH (Mesh Parameters)				
COLOR	Character	Section 6.3.3		'BLACK'
CYLINDRICAL	Logical	Section 6.3.2		.FALSE.
EVACUATION	Logical	Reference [40]		.FALSE.
EVAC_HUMANS	Logical	Reference [40]		.FALSE.
EVAC_Z_OFFSET	Real	Reference [40]	m	1
ID	Character	Reference [40]		
IJK	Integer Triplet	Section 6.3.1		10,10,10
LEVEL	Integer	For future use		0
MPI_PROCESS	Integer	Section 6.3.3		
MULT_ID	Character	Section 7.2.2		
RGB	Integer Triplet	Section 6.3.3		0,0,0
SYNCHRONIZE	Logical	Section 6.3.3		.TRUE.
XB (6)	Real Sextuplet	Section 6.3.1	m	0,1,0,1,0,1

16.14 MISC (Miscellaneous Parameters)

Table 16.14: For more information see Section 6.4.

MISC (Miscellaneous Parameters)				
ALLOW_SURFACE_PARTICLES	Logical	Section 14.5.1		.TRUE.
ALLOW_UNDERSIDE_PARTICLES	Logical	Section 14.5.1		.FALSE.
ASSUMED_GAS_TEMPERATURE	Real	Section 8.5		
BAROCLINIC	Logical	Section 6.4.7		.TRUE.
BNDF_DEFAULT	Logical	Section 15.9.4		.TRUE.
CDF_CUTOFF	Real	Section 14.3.3		0.01
CFL_MAX	Real	Section 6.4.9		1.0
CFL_MIN	Real	Section 6.4.9		0.8
CFL_VELOCITY_NORM	Integer	Section 6.4.9		1
CHECK_GR	Logical	Section 6.4.9		.TRUE.
CHECK_HT	Logical	Section 6.4.9		.TRUE.
CHECK_VN	Logical	Section 6.4.9		.TRUE.
CLIP_MASS_FRACTION	Logical	Section 6.7		.FALSE.
C_DEARDORFF	Real	Section 6.4.8		0.1
C_SMAGORINSKY	Real	Section 6.4.8		0.20
C_VREMAN	Real	Section 6.4.8		0.07
C_FORCED	Real	Section 8.2.2		0.037
C_FORCED_CYLINDER	Real	Section 8.2.2		0.664
C_FORCED_SPHERE	Real	Section 8.2.2		0.600
C_G	Real	Section 6.4.8		0.04
C_HORIZONTAL	Real	Section 8.2.2		1.52
C_VERTICAL	Real	Section 8.2.2		1.31
CONSTANT_SPECIFIC_HEAT	Real	Section 6.4.1		.FALSE.

Table 16.14: Continued

MISC (Miscellaneous Parameters)				
DNS	Logical	Section 6.4.1		.FALSE.
EVACUATION_DRILL	Logical	Reference [40]		.FALSE.
EVACUATION_MC_MODE	Logical	Reference [40]		.FALSE.
EVAC_PRESSURE_ITERATIONS	Integer	Reference [40]		50
EVAC_TIME_ITERATIONS	Integer	Reference [40]		50
FLUX_LIMITER	Integer	Reference [15]		4
FORCE_VECTOR(3)	Real	Section 6.4.3		0.
FREEZE_VELOCITY	Logical	Constant velocity		.FALSE.
GAMMA	Real	Section 11.1.2		1.4
GRAVITATIONAL_DEPOSITION	Logical	Section 12.4		.TRUE.
GROUND_LEVEL	Real	Section 9.5	m	0.
GVEC	Real triplet	Section 6.4.6	m/s ²	0,0,-9.81
INITIAL_UNMIXED_FRACTION	Real	Section 12		1.0
LAPSE_RATE	Real	Section 9.5	°C/m	0
LES	Logical	Section 6.4.1		.TRUE.
MAXIMUM_VISIBILITY	Real	Section 15.10.2	m	30
MEAN_FORCING(3)	Logical	Section 6.4.2		.FALSE.
NOISE	Logical	Section 6.4.1		.TRUE.
NOISE_VELOCITY	Real	Section 6.4.1	m/s	0.005
NO_EVACUATION	Logical	Reference [40]		.FALSE.
OVERWRITE	Logical	Section 6.4.1		.TRUE.
POROUS_FLOOR	Logical	Section 15.3.1		.TRUE.
PR	Real	Section 6.4.8		0.5
P_INF	Real	Section 6.4.1	Pa	101325
RAMP_GX	Character	Section 6.4.6		
RAMP_GY	Character	Section 6.4.6		
RAMP_GZ	Character	Section 6.4.6		
RESTART	Logical	Section 6.4.4		.FALSE.
RESTART_CHID	Character	Section 6.4.4		CHID
RICHARDSON_ERROR_TOLERANCE	Real	Section 12		1.0 E-3
RUN_AVG_FAC	Real	Section 14.3.2		0.5
SC	Real	Section 6.4.8		0.5
SHARED_FILE_SYSTEM	Logical	Section 6.3.3		.TRUE.
SMOKE_ALBEDO	Real	Reference [2]		0.3
SOLID_PHASE_ONLY	Logical	Section 8.5		.FALSE.
STRATIFICATION	Logical	Section 9.5		.TRUE.
TEXTURE_ORIGIN(3)	Char. Triplet	Section 7.5.1	m	(0.,0.,0.)
THERMOPHORETIC_DEPOSITION	Logical	Section 12.4		.TRUE.
THICKEN_OBSTRUCTIONS	Logical	Section 7.2.1		.FALSE.
TPMA	Real	Section 6.4.1	°C	20.
TURBULENCE_MODEL	Character	Section 6.4.8		'DEARDORFF'
TURBULENT_DEPOSITION	Logical	Section 12.4		.TRUE.
U0,V0,W0	Reals	Section 6.4.1	m/s	0.

Table 16.14: Continued

MISC (Miscellaneous Parameters)				
VISIBILITY_FACTOR	Real	Section 15.10.2		3
VN_MAX	Real	Section 6.4.9		1.0
VN_MIN	Real	Section 6.4.9		0.8

16.15 MULT (Multiplier Function Parameters)

Table 16.15: For more information see Section 7.2.2.

MULT (Multiplier Function Parameters)				
DX	Real	Spacing in the x direction	m	0.
DXB	Real Sextuplet	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_UPPER	Integer	Upper array bound, x direction		0
J_LOWER	Integer	Lower array bound, y direction		0
J_UPPER	Integer	Upper array bound, y direction		0
K_LOWER	Integer	Lower array bound, z direction		0
K_UPPER	Integer	Upper array bound, z direction		0
N_LOWER	Integer	Lower sequence bound		0
N_UPPER	Integer	Upper sequence bound		0

16.16 OBST (Obstruction Parameters)

Table 16.16: For more information see Section 7.2.

OBST (Obstruction Parameters)				
ALLOW_VENT	Logical	Section 7.2.1		.TRUE.
BNDF_FACE (-3:3)	Logical Array	Section 15.9.4		.TRUE.
BNDF_OBST	Logical	Section 15.9.4		.TRUE.
BULK_DENSITY	Real	Section 8.4.6	kg/m ³	
COLOR	Character	Section 7.2.1		
CTRL_ID	Character	Section 15.4.2		
DEVC_ID	Character	Section 15.4.2		
EVACUATION	Logical	Reference [40]		.FALSE.
ID	Character	Section 7.2.1		

Table 16.16: Continued

OBST (Obstruction Parameters)				
MESH_ID	Character	Reference [40]		
MULT_ID	Character	Section 7.2.2		
OUTLINE	Logical	Section 7.2.1		.FALSE.
PERMIT_HOLE	Logical	Section 7.3		.TRUE.
PROP_ID	Character	Reference [2]		
REMOVABLE	Logical	Section 7.3		.TRUE.
RGB (3)	Integer Triplet	Section 7.2.1		
SAWTOOTH	Logical	Section 7.2.3		.TRUE.
SURF_ID	Character	Section 7.2.1		
SURF_ID6 (6)	Character Sextuplet	Section 7.2.1		
SURF_IDS (3)	Character Triplet	Section 7.2.1		
TEXTURE_ORIGIN (3)	Real Triplet	Section 7.5.1	m	(0.,0.,0.)
THICKEN	Logical	Section 7.2.1		.FALSE.
TRANSPARENCY	Real	Section 7.2.1		1
XB (6)	Real Sextuplet	Section 7.2.1	m	

16.17 PART (Lagrangian Particles/Droplets)

Table 16.17: For more information see Chapter 14.

PART (Lagrangian Particles/Droplets)				
AGE	Real	Section 14.2.4	s	100000.
BREAKUP	Logical	Section 14.3.4		.FALSE.
BREAKUP_CNF_RAMP_ID	Character	Section 14.3.4		
BREAKUP_DISTRIBUTION	Character	Section 14.3.4		'ROSIN-RAMMLER-LOGNORMAL'
BREAKUP_GAMMA_D	Real	Section 14.3.4		2.4
BREAKUP_RATIO	Real	Section 14.3.4		$\frac{3}{7}$
BREAKUP_SIGMA_D	Real	Section 14.3.4		
CHECK_DISTRIBUTION	Logical	Section 14.3.3		.FALSE.
CNF_RAMP_ID	Character	Section 14.3.3		
COLOR	Character	Section 14.1.1		'BLACK'
COMPLEX_REFRACTIVE_INDEX	Real	Section 14.3.2		0.01
CTRL_ID	Character	Section 14.2.1		
DENSITY	Real	Section 14.3.1	kg/m ³	1000.
DEVC_ID	Character	Section 14.2.1		
DIAMETER	Real	Section 14.3.3	μm	500.
DISTRIBUTION	Character	Section 14.3.3		'ROSIN-RAMMLER-LOGNORMAL'
DRAW_LAW	Character	Section 14.4.2		'SPHERE'
FREE_AREA_FRACTION	Real	Section 14.4.5		
GAMMA_D	Real	Section 14.3.3		2.4
HEAT_OF_COMBUSTION	Real	Section 14.3.5	kJ/kg	
HORIZONTAL_VELOCITY	Real	Section 14.5.1	m/s	0.2

Table 16.17: Continued

PART (Lagrangian Particles/Droplets)				
ID	Character	Section 14.1		
INITIAL_TEMPERATURE	Real	Section 14.3.1	°C	TMPA
MASSLESS	Logical	Section 14.1.2		.FALSE.
MAXIMUM_DIAMETER	Real	Section 14.3.3	μm	∞
MINIMUM_DIAMETER	Real	Section 14.3.3	μm	20.
MONODISPERSE	Logical	Section 14.3.3		.FALSE.
N_STRATA	Integer	Section 14.3.3		7
ORIENTATION(:, 1:3)	Real Array	Section 14.4		
PROP_ID	Character	Section 14.1		
QUANTITIES(10)	Character	Section 14.1.1		
RADIATIVE_PROPERTY_TABLE	Real	Section 14.3.2		
REAL_REFRACTIVE_INDEX	Real	Section 14.3.2		1.33
RGB(3)	Integers	Section 14.1.1		
SAMPLING_FACTOR	Integer	Section 14.2.4		1
SIGMA_D	Real	Section 14.3.3		
SPEC_ID	Character	Section 14.3.1		
STATIC	Logical	Section 14.4		.FALSE.
SURFACE_TENSION	Real	Section 14.3.4	N/m	$72.8 \cdot 10^3$
SURF_ID	Character	Section 14.4		
USER_DRAG_COEFFICIENT(3)	Real Array	Section 14.4.2		-1.
VERTICAL_VELOCITY	Real	Section 14.5.1	m/s	0.5

16.18 PRES (Pressure Solver Parameters)

Table 16.18: For more information see Section 6.6.

PRES (Pressure Solver Parameters)				
CHECK_POISSON	Logical	Section 6.6		.FALSE.
MAX_PRESSURE_ITERATIONS	Integer	Section 6.6		10000
PRESSURE_RELAX_TIME	Real	Section 6.6	s	1.
RELAXATION_FACTOR	Real	Section 6.6		1.0
VELOCITY_TOLERANCE	Real	Section 6.6	m/s	

16.19 PROF (Wall Profile Parameters)

Table 16.19: For more information see Section 15.9.2.

PROF (Wall Profile Parameters)				
ID	Character	Section 15.9.2		
IOR	Real	Section 15.9.2		

Table 16.19: Continued

PROF (Wall Profile Parameters)				
QUANTITY	Character	Section 15.9.2		
XYZ	Real Triplet	Section 15.9.2	m	

16.20 PROP (Device Properties)

Table 16.20: For more information see Section 15.3.

PROP (Device Properties)				
ACTIVATION_OBSCURATION	Real	Section 15.3.5	%/m	3.28
ACTIVATION_TEMPERATURE	Real	Section 15.3.1	°C	74.
ALPHA_C	Real	Section 15.3.5		1.8
ALPHA_E	Real	Section 15.3.5		0.
BEAD_DENSITY	Real	Section 15.10.4	kg/m ³	8908.
BEAD_DIAMETER	Real	Section 15.10.4	m	0.001
BEAD_EMISSIVITY	Real	Section 15.10.4		0.85
BEAD_H_FIXED	Real	Section 15.10.4	W/m ² /K	
BEAD_SPECIFIC_HEAT	Real	Section 15.10.4	kJ/kg/K	0.44
BETA_C	Real	Section 15.3.5		1.
BETA_E	Real	Section 15.3.5		1.
CHARACTERISTIC_VELOCITY	Real	Section 15.10.12	m/s	1.
C_FACTOR	Real	Section 15.3.1		0.
PARTICLES_PER_SECOND	Integer	Section 15.3.1		5000
PARTICLE_VELOCITY	Real	Section 15.3.1	m/s	0.
FLOW_RAMP	Character	Section 15.3.1		
FLOW_RATE	Real	Section 15.3.1	L/min	
FLOW_TAU	Real	Section 15.3.1		0.
GAUGE_TEMPERATURE	Real	Section 15.10.5	°C	TMPA
ID	Character	Section 15.3		
INITIAL_TEMPERATURE	Real	Section 15.3.1	°C	TMPA
K_FACTOR	Real	Section 15.3.1	L/min/bar ^{1/2}	1.
LENGTH	Real	Section 15.3.5		1.8
MASS_FLOW_RATE	Real	Section 15.3.1	kg/s	
OFFSET	Real	Section 15.3.1	m	0.05
OPERATING_PRESSURE	Real	Section 15.3.1	bar	1.
ORIFICE_DIAMETER	Real	Section 15.3.1	m	0.
P0, PX (3) , PXX (3, 3)	Real	Section 15.3.3	m/s	0.
PART_ID	Character	Section 15.3.1		
PDPA_END	Real	Section 15.10.6	s	T_END
PDPA_HISTOGRAM	Logical	Section 15.10.6		.FALSE.
PDPA_HISTOGRAM_NBINS	Integer	Section 15.10.6		10
PDPA_INTEGRATE	Logical	Section 15.10.6		.TRUE.
PDPA_M	Integer	Section 15.10.6		0

Table 16.20: Continued

PROP (Device Properties)				
PDPA_N	Integer	Section 15.10.6		0
PDPA_NORMALIZE	Logical	Section 15.10.6		.TRUE.
PDPA_RADIUS	Real	Section 15.10.6	m	0.
PDPA_START	Real	Section 15.10.6	s	0.
PRESSURE_RAMP	Character	Section 15.3.1		
QUANTITY	Character	Section 15.3.1		
RTI	Real	Section 15.3.1	$\sqrt{\text{m s}}$	100.
SMOKEVIEW_ID	Char. Array	Section 15.7.1		
SMOKEVIEW_PARAMETERS	Char. Array	Section 15.7.2		
SPEC_ID	Character	Section 15.3.5		
SPRAY_ANGLE (2)	Real	Section 15.3.1	deg.	60.,75.
SPRAY_PATTERN_BETA	Integer	Section 15.3.1	deg.	5
SPRAY_PATTERN_MU	Integer	Section 15.3.1	deg.	0
SPRAY_PATTERN_SHAPE	Character	Section 15.3.1		'GAUSSIAN'
SPRAY_PATTERN_TABLE	Character	Section 15.3.1		
VELOCITY_COMPONENT	Integer	Section 15.3.3		

16.21 RADI (Radiation Parameters)

Table 16.21: For more information see Section 13.1.

RADI (Radiation Parameters)				
ANGLE_INCREMENT	Integer	Section 13.1.2		5
CH4_BANDS	Logical	Section 13.2.3		.FALSE.
KAPPA0	Real	Section 13.2.2	1/m	0
NMIEANG	Integer	Section 13.2.2		15
NUMBER_RADIATION_ANGLES	Integer	Section 13.1.2		100
PATH_LENGTH	Real	Section 13.2.3	m	
RADCAL_FUEL	Character	Section 13.2.1		'METHANE'
RADIATION	Logical	Section 13.1		.TRUE.
RADIATIVE_FRACTION	Real	Section 13.1		0.35
RADTMP	Real	Section 13.2.2	°C	900
TIME_STEP_INCREMENT	Integer	Section 13.1.2		3
WIDE_BAND_MODEL	Logical	Section 13.2.3		.FALSE.

16.22 RAMP (Ramp Function Parameters)

Table 16.22: For more information see Chapter 10.

RAMP (Ramp Function Parameters)				
CTRL_ID	Character	Section 15.6.1		
DEVC_ID	Character	Section 15.6.1		
F	Real	Chapter 10		
ID	Character	Chapter 10		
NUMBER_INTERPOLATION_POINTS	Integer	Chapter 10		5000
T	Real	Chapter 10	s (or °C)	
X	Real	Section 6.4.6	m	

16.23 REAC (Reaction Parameters)

Table 16.23: For more information see Chapter 12.

REAC (Reaction Parameters)				
A	Real	Section 12.3	cm ³ /mol/s	
AUTO_IGNITION_TEMPERATURE	Real	Section 12.1.4	K	0
BETA_EDC	Real	Section 12.1.3		1.0
C	Real	Section 12.1.1		3
CHECK_ATOM_BALANCE	Logical	Section 12.5.3		.TRUE.
CO_YIELD	Real	Section 12.1.1	kg/kg	0
CRITICAL_FLAME_TEMPERATURE	Real	Section 12.1.4	°C	1427
E	Real	Section 12.3	kJ/kmol	
EPUMO2	Real	Section 12.1.2	kJ/kg	13100
EQUATION	Character	Section 12.5.1		
FORMULA	Character	Section 12.1.1		
FUEL	Character	Section 12.1.1		
H	Real	Section 12.1.1		8
HEAT_OF_COMBUSTION	Real	Section 12.1.2	kJ/kg	
HRRPUA_SHEET	Real	Section 12.5.2	kW/m ²	
HRRPUV_AVERAGE	Real	Section 12.5.2	kW/m ³	
HUMIDITY	Real	Section 12.1.1	%	40.
ID	Character	Section 12.1.1		
IDEAL	Logical	Section 12.1.1		.FALSE.
N	Real	Section 12.1.1		0
NU (:)	Real Array	Section 12.3		
N_S (:)	Real Array	Section 12.3		
N_T	Real	Section 12.3		
O	Real	Section 12.1.1		0
REAC_ATOM_ERROR	Real	Section 12.5.3	atoms	1.E-5
REAC_MASS_ERROR	Real	Section 12.5.3	kg/kg	1.E-4
SMIX_ID (:)	Char. Array	Section 11.2		
SOOT_H_FRACTION	Real	Section 12.1.1		0.1
SOOT_YIELD	Real	Section 12.1.1	kg/kg	0.01

Table 16.23: Continued

REAC (Reaction Parameters)				
SPEC_ID (:)	Char. Array	Section 11.2		
SUPPRESSION	Logical	Section 12.1.4		.TRUE.
THRESHOLD_TEMPERATURE	Real	Section 12.3.1	°C	1427
Y_CO2_INFTY	Real	Section 12.1.1	kg/kg	0.0058
Y_O2_INFTY	Real	Section 12.1.1	kg/kg	0.232428
Y_P_MIN_EDC	Real	Section 12.1.3	kg/kg	0.0001

16.24 SLCF (Slice File Parameters)

Table 16.24: For more information see Section 15.9.3.

SLCF (Slice File Parameters)				
CELL_CENTERED	Logical	Section 15.9.3		.FALSE.
EVACUATION	Logical	Reference [40]		.FALSE.
MAXIMUM_VALUE	Real	Reference [2]		
MESH_NUMBER	Integer	Section 15.9.3		
MINIMUM_VALUE	Real	Reference [2]		
PART_ID	Character	Section 15.12		
PBX, PBZ, PBZ	Real	Section 15.9.3		
QUANTITY	Character	Section 15.12		
QUANTITY2	Character	Section 15.12		
SPEC_ID	Character	Section 15.12		
VECTOR	Logical	Section 15.9.3		.FALSE.
VELO_INDEX	Integer	Section 15.10.15		0
XB (6)	Real Sextuplet	Section 15.9.3	m	

16.25 SMIX (Species Mixture (Lumped Species) Parameters)

Table 16.25: For more information see Section 11.2.

SMIX (Species Mixture (Lumped Species) Parameters)				
BACKGROUND	Logical	Section 11.2		.FALSE.
ID	Character	Section 11.2		
SPEC_ID (:)	Character Array	Section 11.2		
MASS_FRACTION (:)	Real Array	Section 11.2		0
MASS_FRACTION_0	Real	Section 11.2		0
VOLUME_FRACTION (:)	Real Array	Section 11.2		

16.26 SPEC (Species Parameters)

Table 16.26: For more information see Section 11.1.

SPEC (Species Parameters)				
ABSORBING	Logical	Section 11.1.2		.FALSE.
AEROSOL	Logical	Section 12.4		.FALSE.
BACKGROUND	Logical	Section 11.1		.FALSE.
CONDUCTIVITY	Real	Section 11.1.2	W/m/K	
CONDUCTIVITY_SOLID	Real	Section 12.4	W/m/K	0.26
DENSITY_LIQUID	Real	Section 14.3.1	kg/m ³	
DENSITY_SOLID	Real	Section 12.4	kg/m ³	1800.
DIFFUSIVITY	Real	Section 11.1.2	m ² /s	
EPSILONKLJ	Real	Section 11.1.2		0
FIC_CONCENTRATION	Real	Section 15.10.9	ppm	0.
FLD_LETHAL_DOSE	Real	Section 15.10.9	ppm×min	0.
FORMULA	Character	Section 11.1.2		
HEAT_OF_VAPORIZATION	Real	Section 14.3.1	kJ/kg	
HUMIDITY	Real	Section 11.1.1	%	40.
H_V_REFERENCE_TEMPERATURE	Real	Section 14.3.1	°C	
ID	Character	Section 11.1.1		
MASS_EXTINCTION_COEFFICIENT	Real	Section 15.3.5		0
MASS_FRACTION_0	Real	Section 11.1.1		0
MEAN_DIAMETER	Real	Section 12.4	m	1.E-6
MELTING_TEMPERATURE	Real	Section 14.3.1	°C	
MW	Real	Section 11.1.2	g/mol	29.
RAMP_CP	Character	Section 11.1.2		
RAMP_CP_L	Character	Section 14.3.1		
RAMP_D	Character	Section 11.1.2		
RAMP_K	Character	Section 11.1.2		
RAMP_MU	Character	Section 11.1.2		
REFERENCE_ENTHALPY	Real	Section 11.1.2	kJ/kg	
REFERENCE_TEMPERATURE	Real	Section 11.1.2	°C	25.
SIGMALJ	Real	Section 11.1.2		0
SMIX_COMPONENT_ONLY	Logical	Section 11.2		.FALSE.
SPECIFIC_HEAT	Real	Section 11.1.2	kJ/kg/K	
SPECIFIC_HEAT_LIQUID	Real	Section 14.3.1	kJ/kg/K	
VAPORIZATION_TEMPERATURE	Real	Section 14.3.1	K	
VISCOSITY	Real	Section 11.1.2	kg/m/s	

16.27 SURF (Surface Properties)

Table 16.27: For more information see Section 7.1.

SURF (Surface Properties)				
ADIABATIC	Logical	Section 8.2.3		.FALSE.
BACKING	Character	Section 8.3.3		'VOID'
BURN_AWAY	Logical	Section 8.4.6		.FALSE.
CELL_SIZE_FACTOR	Real	Section 8.3.7		1.0
COLOR	Character	Section 7.5		
CONVECTIVE_HEAT_FLUX	Real	Section 8.2.2	kW/m ²	0.
CONVECTION_LENGTH_SCALE	Real	Section 8.2.2	m	1.
DEFAULT	Logical	Section 7.1		.FALSE.
DT_INSERT	Real	Section 14.2.1	s	0.01
EMISSIVITY	Real	Section 8.2.2		0.9
EMISSIVITY_BACK	Real	Section 8.3.3		
EVAC_DEFAULT	Logical	Reference [40]		.FALSE.
EXTERNAL_FLUX	Real	Section 8.5	kW/m ²	0.
E_COEFFICIENT	Real	Section 14.5	m ² /kg/s	0.
FREE_SLIP	Logical	Section 9.1.5		.FALSE.
GEOMETRY	Character	Section 8.3.6		'CARTESIAN'
HEAT_OF_VAPORIZATION	Real	Section 8.4.3	kJ/kg	0.
HRRPUA	Real	Section 8.4.1	kW/m ²	0.
H_FIXED	Real	Section 8.2.2	W/m ² /K	
ID	Character	Section 7.1		
IGNITION_TEMPERATURE	Real	Section 8.4.3	°C	5000.
LAYER_DIVIDE	Real	Section 8.3.5		N_LAYERS/2
LEAK_PATH	Int. Pair	Section 9.3.2		
LENGTH	Real	Section 14.4	m	
MASS_FLUX (:)	Real Array	Section 9.1.6	kg/m ² s	0.
MASS_FLUX_TOTAL	Real	Section 9.1.2	kg/m ² s	
MASS_FRACTION (:)	Real Array	Section 9.1.6		
MATL_ID (NL, NC)	Char. Array	Section 8.4.4		
MATL_MASS_FRACTION (NL, NC)	Real Array	Section 8.4.4		
MINIMUM_LAYER_THICKNESS	Real	Section 8.3.7	m	1.E-6
MLRPUA	Real	Section 8.4.1	kg/m ² s	0.
NET_HEAT_FLUX	Real	Section 8.2.2	kW/m ²	0.
NO_SLIP	Logical	Section 9.1.5		.FALSE.
NPPC	Integer	Section 14.2.1		1
PARTICLE_MASS_FLUX	Real	Section 14.2.1	kg/m ² s	0.
PART_ID	Character	Section 14.2.1		
PLE	Real	Section 9.5		0.3
PROFILE	Character	Section 9.5		
RADIUS	Real	Section 14.4	m	
RAMP_EF	Character	Section 10.1		
RAMP_MF (:)	Character	Section 10.1		

Table 16.27: Continued

SURF (Surface Properties)				
RAMP_PART	Character	Section 10.1		
RAMP_Q	Character	Section 10.1		
RAMP_T	Character	Section 10.1		
RAMP_V	Character	Section 10.1		
REGRID_FACTOR	Real	Section 8.3.7		0.9
RGB (3)	Int. Triplet	Section 7.5		255,204,102
ROUGHNESS	Real	Section 9.1.5	m	0.
SPEC_ID	Character	Section 9.1.6		
SPREAD_RATE	Real	Section 8.4.2	m/s	0.
STRETCH_FACTOR (:)	Real	Section 8.3.7		2.
SURFACE_DENSITY	Real	Section 8.4.6	kg/m ²	0.
TAU_EF	Real	Section 10.1	s	1.
TAU_MF (:)	Real	Section 10.1	s	1.
TAU_PART	Real	Section 10.1	s	1.
TAU_Q	Real	Section 10.1	s	1.
TAU_T	Real	Section 10.1	s	1.
TAU_V	Real	Section 10.1	s	1.
TEXTURE_HEIGHT	Real	Section 7.5.1	m	1.
TEXTURE_MAP	Character	Section 7.5.1		
TEXTURE_WIDTH	Real	Section 7.5.1	m	1.
THICKNESS (NL)	Real Array	Section 8.1	m	0.
TMP_BACK	Real	Section 8.3.4	°C	20.
TMP_FRONT	Real	Section 8.2.1	°C	20.
TMP_INNER (:)	Real Array	Section 8.3.4	°C	20.
TRANSPARENCY	Real	Section 7.5		1.
VEL	Real	Section 9.1	m/s	0.
VEL_T	Real Pair	Section 9.1.4	m/s	0.
VOLUME_FLUX	Real	Section 9.1	m ³ /s	0.
WIDTH	Real	Section 14.4	m	
XYZ (3)	Real Triplet	Section 8.4.2	m	
Z0	Real	Section 9.5	m	10.

16.28 TABL (Table Parameters)

Table 16.28: For more information see Section 10.3.

TABL (Table Parameters)				
ID	Character	Section 10.3		
TABLE_DATA (9)	Real Array	Section 10.3		

16.29 TIME (Time Parameters)

Table 16.29: For more information see Section 6.2.

TIME (Time Parameters)				
DT	Real	Section 6.2.2	s	
EVAC_DT_FLOWFIELD	Real	Reference [40]	s	0.01
EVAC_DT_STEADY_STATE	Real	Reference [40]	s	0.05
LIMITING_DT_RATIO	Real	Section 4.2		0.0001
LOCK_TIME_STEP	Logical	Section 6.2.2		.FALSE.
RESTRICT_TIME_STEP	Logical	Section 6.2.2		.TRUE.
SYNCHRONIZE	Logical	Section 6.2.2		.TRUE.
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 8.3.7		2

16.30 TRNX, TRNY, TRNZ (MESH Transformations)

Table 16.30: For more information see Section 6.3.5.

TRNX, TRNY, TRNZ (MESH Transformations)				
CC	Real	Section 6.3.5	m	
IDERIV	Integer	Section 6.3.5		
MESH_NUMBER	Integer	Section 6.3.5		
PC	Real	Section 6.3.5		

16.31 VENT (Vent Parameters)

Table 16.31: For more information see Section 7.4.

VENT (Vent Parameters)				
COLOR	Character	Section 7.5		
CTRL_ID	Character	Section 15.4.2		
DEVC_ID	Character	Section 15.4.2		
DYNAMIC_PRESSURE	Real	Section 9.4	Pa	0.
EVACUATION	Logical	Reference [40]		.FALSE.
ID	Character	Section 7.4.1		
IOR	Integer	Section 7.4.4		
L_EDDY	Real	Section 7.4.5	m	0.
L_EDDY_IJ (3, 3)	Real Array	Section 7.4.5	m	0.
MB	Character	Section 7.4.1		

Table 16.31: Continued

VENT (Vent Parameters)				
MESH_ID	Character	Reference [40]		
N_EDDY	Integer	Section 7.4.5		0
OUTLINE	Logical	Section 7.4.1	.FALSE.	
PBX, PBZ, PBZ	Real	Section 7.4.1		
PRESSURE_RAMP	Character	Section 9.4		
REYNOLDS_STRESS (3, 3)	Real Array	Section 7.4.5	m ² /s ²	0.
RGB (3)	Integer Triplet	Section 7.5		
SPREAD_RATE	Real	Section 8.4.2	m/s	0.0
SURF_ID	Character	Section 7.4.1		' INERT '
TEXTURE_ORIGIN (3)	Real Triplet	Section 7.5.1	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 Triplet	Section 9.2.6	m/s	
VEL_RMS	Real	Section 7.4.5	m/s	0.
XB (6)	Real Sextuplet	Section 7.4.1	m	
XYZ (3)	Real Triplet	Section 8.4.2	m	

16.32 ZONE (Pressure Zone Parameters)

Table 16.32: For more information see Section 9.3.

ZONE (Pressure Zone Parameters)				
ID	Character	Section 9.3.1		
LEAK_AREA (N)	Real	Section 9.3.2	m ²	0
XB (6)	Real Sextuplet	Section 9.3.1	m	

16.33 Modifications of Input Parameters from FDS 5 to FDS 6

This section describes the changes in the input parameters between FDS version 5 and version 6. Table 16.33 lists in alphabetical order parameters from FDS 5 that have changed. Note that this table does not list new parameters in FDS 6.

There has been a limited amount of backward compatibility programmed into FDS 6. In other words, several commonly used parameters and conventions from previous versions still work, but you are encouraged to gradually modify your input files to conform to the new conventions. Gradually, obsolescent features will be removed. Some of the more notable changes in FDS 6 are:

- If you want to model a fire, you *must* include a REAC line with a specified FUEL. See Chapter 12 for details.
- The output quantity 'MIXTURE_FRACTION' no longer exists and there is no substitute in FDS 6.
- The ISOTHERMAL feature is no longer an option.
- There is no longer a STATE_FILE because there is no longer a simple mixture fraction model.
- PRESSURE_CORRECTION has been eliminated. See Section 6.6 for ways to improve the performance of the pressure solver.
- Species mass and volume fraction outputs are no longer invoked using QUANTITY='species name'. Use QUANTITY='MASS FRACTION' or QUANTITY='VOLUME FRACTION' along with SPEC_ID instead.

Table 16.33: Changes to input parameters, FDS version 5 to 6.

Name/ist	FDS 5 Parameter	Name/ist	FDS 6 Parameter	Notes
CLIP	MAXIMUM_MASS_FRACTION		Eliminated	
CLIP	MINIMUM_MASS_FRACTION		Eliminated	
DUMP	STATE_FILE		Eliminated	
MATL	NU_FUEL	MATL	NU_SPEC + SPEC_ID	Section 8.4.4
MATL	NU_GAS	MATL	NU_SPEC + SPEC_ID	Section 8.4.4
MATL	NU_RESIDUE	MATL	NU_MATL	Section 8.4.4
MATL	NU_WATER	MATL	NU_SPEC + SPEC_ID	Section 8.4.4
MATL	RESIDUE	MATL	MATL_ID	Section 8.4.4
MISC	BACKGROUND_SPECIES	SPEC	BACKGROUND=.TRUE.	Section 11.1
MISC	CONDUCTIVITY	SPEC	CONDUCTIVITY	Section 11.1
MISC	CO_PRODUCTION		Eliminated	
MISC	CSMAG	MISC	C_SMAGORINSKY	Same functionality
MISC	HUMIDITY	SPEC	HUMIDITY	Section 11.1
MISC	ISOTHERMAL		Eliminated	
MISC	MW	SPEC	MW	Section 11.1
MISC	PRESSURE_CORRECTION		Eliminated	
MISC	RADIATION	RADI	RADIATION	Same functionality
MISC	EVAC_SURF_DEFAULT	SURF	EVAC_DEFAULT	Section 7.1
MISC	SURF_DEFAULT	SURF	DEFAULT	Section 7.1
MISC	SUPPRESSION	REAC	SUPPRESSION	Section 12.1.4
MISC	VISCOSITY	SPEC	VISCOSITY	Section 11.1
PART	PARTICLES_PER_SECOND	PROP	PARTICLES_PER_SECOND	Section 14.2.2
PART	HEAT_OF_VAPORIZATION	SPEC	HEAT_OF_VAPORIZATION	Section 14.3.1
PART	H_V_REFERENCE_TEMPERATURE	SPEC	H_V_REFERENCE_TEMPERATURE	Section 14.3.1
PART	MELTING_TEMPERATURE	SPEC	MELTING_TEMPERATURE	Section 14.3.1
PART	NUMBER_INITIAL_PARTICLES	INIT	N_PARTICLES	Section 14.2.3
PART	SPECIFIC_HEAT	SPEC	SPECIFIC_HEAT_LIQUID	Section 14.3.1
PART	VAPORIZATION_TEMPERATURE	SPEC	VAPORIZATION_TEMPERATURE	Section 14.3.1

Table 16.34: Changes to input parameters, FDS version 5 to 6 (continued).

Namelist	FDS 5 Parameter	Namelist	FDS 6 Parameter	Notes
PROP	DT_INSERT		Eliminated	
REAC	BOF	REAC	A	Same functionality
REAC	ID	REAC	FUEL	New requirement
REAC	MASS_EXTINCTION_COEFFICIENT	SPEC	MASS_EXTINCTION_COEFFICIENT	Same functionality
REAC	MAXIMUM_VISIBILITY	MISC	MAXIMUM_VISIBILITY	Same functionality
REAC	OXIDIZER		Eliminated	
REAC	VISIBILITY_FACTOR	MISC	MAXIMUM_VISIBILITY	Same functionality
SURF	POROUS		Eliminated	Use HVAC
TIME	TWFIN	TIME	T_END	Section 6.2
VENT	MASS_FRACTION		Eliminated	

Part III

FDS and Smokeview Development Tools

Chapter 17

The FDS/Smokeview Repository

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 entire FDS/Smokeview “Repository.” All project documents are maintained using the online utility [Google Code Project Hosting](#), a free service offered by Google to support software development for open source applications. Google Code uses the [Subversion](#) (SVN) revision management system. Under this system a centralized repository containing all project files resides on a Google Code server. Subversion uses a single integer that identifies the version of the entire repository rather than of a specific file (i.e. anytime a change is made to the repository all files are incremented in version number). A record of version number when a specific file was last changed is maintained.

As an open source program, any individual can obtain a copy of the repository or retrieve specific versions repository. Only the FDS and Smokeview developers can commit changes to the repository.

The current location of the FDS repository is <http://fds-smv.googlecode.com/svn/trunk/>. The repository contains the following files:

1. FDS and Smokeview source code files
2. FDS and Smokeview documentation
3. Input files for software testing (Examples), verification testing, and validation testing
4. Experimental data files used for validation testing
5. Scripts and post-processing utilities used for software testing
6. Web pages and wikis

The wikis are particularly useful in describing the details of how you go about working with the Repository assets.

Chapter 18

Compiling FDS

This section describes what you need to know if you want to compile the FDS source code yourself. It is not a step by step guide, more detailed instructions can be found on Developer section of the web site at <http://fire.nist.gov/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_win_32.exe** is the compiled single processor program for a 32 bit Windows-based PC; thus PC users do not need a Fortran compiler and do not need to compile the source code. For machines for which an executable has not been compiled, you must compile the code. Fortran 90/95 compilers are needed for compilation.

18.1 FDS Source Code

Table 18.1 lists the files that make up the source code. The files with suffix “.f90” contain free form Fortran 90 instructions conforming to the ANSI and ISO standards, with a few exceptions that are discussed below. The source files should be compiled in the order in which they are listed in Table 18.1 because some routines are dependent on others. For Unix/Linux users, **Makefiles** for various platforms are available that assist in the compilation. Compiler options differ from platform to platform. Note the following:

- The source code consists entirely of Fortran 90 statements organized into about 25 files.
- There is only one non-standard call in the Fortran code. The non-standard call is `GETARG`, in **func.f90**. This routine reads the name of the input file off of the command line. This call cannot be simply commented out; a suitable alternative must be found. The only compiler option necessary, in addition to any needed to address the above issues, is for full optimization (usually `-O` or some variant). Some compilers have a standard optimization level, plus various degrees of “aggressive” optimization. Be cautious in using the highest levels of optimization.
- For the single processor version of FDS, compile with **mpis.f90**
- The parallel version of FDS uses **mpip.f90** instead of **mpis.f90**, plus additional MPI libraries need to be installed. More details on MPI can be found at the web site, along with links to the necessary organizations who have developed free MPI libraries.

Table 18.1: **Source Code Files**

File Name	Description
cons.f90	Global arrays and constants
ctrl.f90	Definitions and routines for control functions
data.f90	Data for output quantities and thermophysical properties
devc.f90	Derived type definitions and constants for devices
divg.f90	Compute the flow divergence
dump.f90	Output data dumps into files
evac.f90	Egress computations (future capability)
fire.f90	Combustion routines
func.f90	Global functions and subroutines
ieva.f90	Support routines for evac.f90
init.f90	Initialize variables and Poisson solver
irad.f90	Functions needed for radiation solver, including RadCal
main.f90	Main program for both serial and parallel versions
mass.f90	Mass equation(s) and thermal boundary conditions
mesh.f90	Arrays and constants associated with each mesh
mpip.f90	MPI "include" statement for MPI compilation
mpis.f90	"Dummy" Fortran/MPI bindings for non-MPI compilation
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
radi.f90	Radiation solver
read.f90	Read input parameters
smvv.f90	Routines for computing and outputting 3D smoke and isosurfaces
turb.f90	Experimental routines, mostly involving the turbulence model
type.f90	Derived type definitions
vege.f90	Experimental vegetation model
velo.f90	Momentum equations
wall.f90	Wall boundary conditions

Chapter 19

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 routine **dump.f**, and can easily be modified to accommodate various plotting packages.

19.1 Diagnostic Output

The file **CHID.out** consists of a list of the input parameters, and an accounting of various important quantities, including CPU usage. Typically, diagnostic information is printed out every 100 time steps

```

      :
      :
Iteration   8300   May 16, 2003   08:37:53
-----
Mesh  1, Cycle   3427
CPU/step:      2.272 s, Total CPU:      2.15 hr
Time step:  0.03373 s, Total time:   128.86 s
Max CFL number: 0.86E+00 at ( 21,  9, 80)
Max divergence: 0.24E+01 at ( 25, 30, 22)
Min divergence: -.39E+01 at ( 26, 18, 31)
Number of Sprinkler Droplets:      615
Total Heat Release Rate:      7560.777 kW
Radiation Loss to Boundaries:    6776.244 kW
Mesh  2, Cycle   2914
CPU/step:      1.887 s, Total CPU:      1.53 hr
Time step:  0.03045 s, Total time:   128.87 s
Max CFL number: 0.96E+00 at ( 21, 29, 42)
Max divergence: 0.20E+01 at ( 22, 20, 22)
Min divergence: -.60E+01 at (  7, 26, 48)
Number of Sprinkler Droplets:      301
      :
      :
```

The Iteration number indicates how many time steps the code has run, whereas the Cycle number for a given mesh indicates how many time steps have been taken on that mesh. The date and time (wall clock time) are on the line starting with the word Iteration. The quantity CPU/step is the amount of CPU time required to complete a time step for that mesh; Total CPU is the amount of CPU time elapsed since the start of the run; Time step is the time step size for the given mesh; Total time is the time of the simulation;

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); and Max CFL number is the maximum value of the CFL number. The Radiation Loss to Boundaries is the amount of energy that is being radiated to the boundaries. As compartments heat up, the energy lost to the boundaries can grow to be an appreciable fraction of the Total Heat Release Rate. Finally, Number of Tracer Particles indicates how many passive particles are being tracked at that time.

Following the completion of a successful run, a summary of the CPU usage per subroutine is listed. This is useful in determining where most of the computational effort is being placed.

19.2 Heat Release Rate and Related Quantities

The heat release rate of the fire, plus other global energy-related quantities, are automatically written into a text file called **CHID_hrr.csv**. The format of the file is as follows

```
s, kW, kW, kW, kW, ..., kg/s, Pa, Pa, ...
Time, HRR, Q_RADI, Q_CONV, Q_COND, ..., BURN_RATE, ZONE_01, ZONE_02, ...
0.0000000E+000, 0.0000000E+000, ...
3.5355338E-001, 0.0000000E+000, ...
.
.
.
```

Details of the integrated energy quantities can be found in Section 15.10.1. BURN_RATE is the total mass loss rate of fuel, and ZONE_01, *etc.*, are the background pressures of the various pressure ZONES. Note that the reported BURN_RATE is not adjusted to account for the possibility that each individual material might have a different heat of combustion. It is the actual burning rate of the fuel as predicted by FDS or specified by the user. The background pressure is discussed in Section 9.3.

19.3 Device Output Data

Data associated with particular devices (link temperatures, smoke obscuration, thermocouples, *etc.*) specified in the input file under the namelist group DEVC is output in comma delimited format in a file called **CHID_devc.csv**. The format of the file is as follows

```
s          , UNITS(1) , UNITS(2) , ... , UNITS(N_DEVC)
FDS Time  , ID(1)    , ID(2)    , ... , ID(N_DEVC)
T(1)      , VAL(1,1) , VAL(2,1) , ... , VAL(N_DEVC,1)
T(2)      , VAL(1,2) , VAL(2,2) , ... , VAL(N_DEVC,2)
.
.
.
```

where N_DEVC is the number of devices, ID(I) is the user-defined ID of the Ith device, UNITS(I) the units, T(J) the time of the Jth dump, and VAL(I, J) the value at the Ith device at the Jth time. The files can be imported into Microsoft Excel or almost any other spread sheet program. If the number of columns exceeds 256, the file will automatically be split into smaller files.

19.4 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, status
FDS Time, ID(1), ID(2), ..., ID(N_CTRL)
0.00000E+000, -001, 001, ...
1.11803E-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 = .FALSE.` and `+1 = .TRUE.` of the `I`th control function at the particular time. The files can be imported into Microsoft Excel or almost any other spread sheet program. If the number of columns exceeds 256, the file will automatically be split into smaller files.

19.5 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 = .TRUE.` on the `DUMP` line to create this output file.

19.6 Slice Files

The slice files defined under the namelist group `SLCF` are named **CHID_n.sf** ($n=01,02,\dots$), and are written out unformatted, unless otherwise directed. These files are written out from **dump.f** 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.f**, 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 15.11 for more details about the program **fds2ascii**.

19.7 Plot3D Data

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=.TRUE.` is included on the `DUMP` line, then the mesh data is written out to a file called **CHID.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_****_**.q**, where the stars indicate a time at which 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/m^3). 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.

Smokeyview 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, including AVS, IRIS Explorer and Tecplot ¹.

19.8 Boundary Files

The boundary files defined under the namelist group `BNDF` are named **CHID_n.bf** ($n=0001,0002\dots$), and are written out unformatted. These files are written out from **dump.f** with the following lines:

```
WRITE(LUBF) QUANTITY
WRITE(LUBF) SHORT_NAME
WRITE(LUBF) UNITS
WRITE(LUBF) NPATCH
WRITE(LUBF) I1,I2,J1,J2,K1,K2,IOR,NB,NM
WRITE(LUBF) I1,I2,J1,J2,K1,K2,IOR,NB,NM
.
```

¹With the exception of Smokeyview, the graphics packages referred to in this document are not included with the source code, but are commercially available.

```

      .
      .
WRITE(LUBF) TIME
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
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 lengths 60, 30 and 30, respectively. NPATCH is the number of planes (or “patches”) that make up the solid boundaries plus the external walls. The sextuplet (I1, I2, J1, J2, K1, K2) defines the cell nodes of each patch. IOR is an integer indicating the orientation of the patch ($\pm 1, \pm 2, \pm 3$). You do not prescribe these. NB is the number of the boundary (zero for external walls) and NM is the number of the mesh. Note that the data is planar, thus one pair of cell nodes is the same. Presently, Smokeview is the only program available to view the boundary files.

19.9 Particle Data

Coordinates and specified quantities related to tracer particles, sprinkler droplets, and other Lagrangian particles are written to a FORTRAN unformatted (binary) file called **CHID.prt5**. 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:

```

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.

19.10 Profile Files

The profile files defined under the namelist group `PROF` are named **CHID_prof_nn.csv** ($nn=01,02\dots$), and are written out formatted. These files are written out from **dump.f** 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.

19.11 3-D Smoke Files

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:

```

! header

WRITE(LU_SMOKE3D) ONE, VERSION, 0, NX-1, 0, NY-1, 0, NZ-1

! time depended data

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 compressed using run-length encoding (RLE).

19.12 Geometry, Isosurface Files

Both immersed geometric surfaces (generalized obstructions) and FDS generated isosurfaces are stored using a file format described in this section. Iso-surface files 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. The file format allows one to specify objects that change with time. Static geometry is defined once and displayed by Smokeview unchanged at each time step. Dynamic geometry is defined at each time step either in terms of nodes and faces or in terms of a translation and two rotations (azimuthal and elevation) of dynamic geometry defined in the first time step. These files are written out from **dump.f90** using lines equivalent to the following:

```
! header

WRITE(LU_GEOM) ONE
WRITE(LU_GEOM) VERSION
WRITE(LU_GEOM) N_FLOATS
IF (N_FLOATS>0) WRITE(LU_GEOM) (FLOAT_HEADER(I), I=1, N_FLOATS)
WRITE(LU_GEOM) N_INTS
IF (N_INTS>0) WRITE(LU_GEOM) (INT_HEADER(I), I=1, N_INTS)

! static geometry - geometry specified once and appearing at all time steps

WRITE(LU_GEOM) N_VERT_S, N_FACE_S
IF (N_VERT_S>0) WRITE(LU_GEOM) (Xvert_S(I), Yvert_S(I), Zvert_S(I), I=1, N_VERT_S)
IF (N_FACE_S>0) WRITE(LU_GEOM) (FACE1_S(I), FACE2_S(I), FACE3_S(I), I=1, N_FACE_S)
IF (N_FACE_S>0) WRITE(LU_GEOM) (SURF_S(I), I=1, N_FACE_S)

! dynamic geometry - geometry specified and appearing for each time step

WRITE(LU_GEOM) STIME, GEOM_TYPE
IF (GEOM_TYPE.EQ.0) THEN
  WRITE(LU_GEOM) N_VERT_D, N_FACE_D
  IF (N_VERT_D>0) WRITE(LU_GEOM) (Xvert_D(I), Yvert_D(I), Zvert_D(I), I=1, N_VERT_D)
  IF (N_FACE_D>0) WRITE(LU_GEOM) (FACE1_D(I), FACE2_D(I), FACE3_D(I), I=1, N_FACE_D)
  IF (N_FACE_D>0) WRITE(LU_GEOM) (SURF_D(I), I=1, N_FACE_D)
ELSE IF (GEOM_TYPE.EQ.1) THEN
  ! rotation and translation parameters used to transform geometry from first dynamic time step
  WRITE(LU_GEOM) Xtran, Ytran, Ztran, Xrot0, Yrot0, Zrot0, rot_az, rot_elev
ENDIF
.
.
.
```

- ONE has the value 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 0 and indicates the version number of this file format.
- N_FLOATS, N_INTS The number of floating point and integer data items stored at the beginning of the file.
- FLOAT_HEADER, INT_HEADER Floating point and integer data stored at the beginning of the file.
- STIME is the FDS simulation time.
- N_VERT_S, N_FACE_S, N_VERT_D, N_FACE_D are the number of static and dynamic vertices and faces.
- Xvert_S, Yvert_S, Zvert_S, Xvert_D, Yvert_D, Zvert_D are the static and dynamic vertex coordinates.

- FACE1_S, FACE2_S, FACE3_S, FACE1_D, FACE2_D, FACE3_D are the static and dynamic vertex indices for each face (triangle). The indices are numbered relative to how vertices were written out earlier.
- SURF_S, SURF_D are the static and dynamic SURF indices for each face (triangle).
- GEOM_TYPE is flag indicating how dynamic geometry is represented. If GEOM_TYPE is 0 then time dependent geometry is written out in terms of nodes and faces using the same format as the static geometry. If GEOM_TYPE is 1 then time dependent geometry is written out in terms of a translation and two rotations. These transformations are applied to the dynamic geometry defined at the first time step.
- Xtran, Ytran, Ztran is the translation applied to the initial dynamic geometry (If GEOM_TYPE is 1)
- Xrot0, Yrot0, Zrot0 is the origin about which rotations occur.
- rot_az, rot_elev are the azimuthal and elevation rotation angles (in degrees) applied to the initial dynamic geometry.

19.13 Geometry Data Files

The geometry data file contains a description of data values computed by FDS on an immersed geometrical objects. This file is analogous to the boundary file. The data written out to a geometry data file **MUST** correspond to the geometry written out in the corresponding geometry file. Geometry data files are written out from **dump.f** with the lines equivalent to the following:

```
WRITE(LU_GEOM_DATA) ONE
WRITE(LU_GEOM_DATA) VERSION
WRITE(LU_GEOM_DATA) STIME
WRITE(LU_GEOM_DATA) N_VERT_S_VALS,N_VERT_D_VALS,N_FACE_S_VALS,N_FACE_D_VALS
IF (N_VERT_S_VALS>0) WRITE(LU_GEOM_DATA) (ValVertStatic(I), I=1,N_VERT_S_VALS)
IF (N_VERT_D_VALS>0) WRITE(LU_GEOM_DATA) (ValVertDynamic(I), I=1,N_VERT_D_VALS)
IF (N_FACE_S_VALS>0) WRITE(LU_GEOM_DATA) (ValFaceStatic(I), I=1,N_VERT_S_VALS)
IF (N_FACE_D_VALS>0) WRITE(LU_GEOM_DATA) (ValFaceDynamic(I), I=1,N_FACE_D_VALS)
.
.
.
```

The data values written out in this file correspond to the geometry written out in the geometry file.

- ONE has the value 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 0 and indicates the version number of this file format.
- STIME is the FDS simulation time.
- N_VERT_S_VALS, N_FACE_S_VALS is the number of data values written out for static vertices and faces. One can write out data values located at nodes, located at the center of faces or both.
- N_VERT_D_VALS, N_FACE_D_VALS is the number of dynamic values written out for dynamic vertices and faces. One can write out data values located at nodes, located at the center of faces, both or neither (if there is no dynamic geometry).

- `ValVertStatic`, `ValFaceStatic` static vertex and face data.
- `ValVertDynamic`, `ValFaceDynamic` dynamic vertex and face data.

Bibliography

- [1] K.B. McGrattan, S. Hostikka, J.E. Floyd, W.E. Mell, and R. McDermott. Fire Dynamics Simulator, Technical Reference Guide, Volume 1: Mathematical Model. NIST Special Publication 1018, National Institute of Standards and Technology, Gaithersburg, Maryland, October 2007. [v](#), [3](#), [37](#), [38](#), [89](#), [122](#), [134](#), [156](#), [160](#), [212](#)
- [2] G.P. Forney. Smokeview (Version 5), A Tool for Visualizing Fire Dynamics Simulation Data, Volume I: User's Guide. NIST Special Publication 1017-1, National Institute of Standards and Technology, Gaithersburg, Maryland, August 2007. [v](#), [3](#), [5](#), [176](#), [219](#), [221](#), [224](#), [226](#), [231](#)
- [3] R. McDermott, K.B. McGrattan, S. Hostikka, and J.E. Floyd. Fire Dynamics Simulator, Technical Reference Guide, Volume 2: Verification. NIST Special Publication 1018, National Institute of Standards and Technology, Gaithersburg, Maryland, October 2007. [3](#)
- [4] K.B. McGrattan, S. Hostikka, J.E. Floyd, and R. McDermott. Fire Dynamics Simulator, Technical Reference Guide, Volume 3: Experimental Validation. NIST Special Publication 1018, National Institute of Standards and Technology, Gaithersburg, Maryland, October 2007. [3](#)
- [5] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI – Portable Parallel Programming with the Message-Passing Interface*. MIT Press, Cambridge, Massachusetts, 2nd edition, 1999. [8](#)
- [6] K. Hill, J. Dreisbach, F. Joglar, B. Najafi, K. McGrattan, R. Peacock, and A. Hamins. Verification and Validation of Selected Fire Models for Nuclear Power Plant Applications. NUREG 1824, United States Nuclear Regulatory Commission, Washington, DC, 2007. [33](#)
- [7] E. Kalnay. *Atmospheric Modeling, Data Assimilation, and Predictability*. Cambridge, 2003. [34](#)
- [8] 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. [37](#)
- [9] J.W. Deardorff. Stratocumulus-capped mixed layers derived from a three-dimensional model. *Boundary-Layer Meteorol.*, 18:495–527, 1980. [37](#), [38](#)
- [10] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000. [37](#), [38](#), [201](#)
- [11] J. Smagorinsky. General Circulation Experiments with the Primitive Equations. I. The Basic Experiment. *Monthly Weather Review*, 91(3):99–164, March 1963. [38](#)
- [12] M. Germano, U. Piomelli, P. Moin, and W. Cabot. A dynamic subgrid-scale eddy viscosity model. *Phys. Fluids A*, 3(7):1760–1765, 1991. [38](#)
- [13] 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. [38](#)

- [14] B. Vreman. An eddy-viscosity subgrid-scale model for turbulent shear flow: Algebraic theory and applications. *Phys. Fluids*, 16(10):3670–3681, 2004. [38](#)
- [15] National Institute of Standards and Technology, Gaithersburg, Maryland, USA, and VTT Technical Research Centre of Finland, Espoo, Finland. *Fire Dynamics Simulator, Technical Reference Guide*, October 2007. NIST Special Publication 1018 (Four volume set). [52](#), [135](#), [224](#)
- [16] N. Jarrin. *Synthetic Inflow Boundary Conditions for the Numerical Simulation of Turbulence*. PhD thesis, The University of Manchester, Manchester M60 1QD, United Kingdom, 2008. [54](#)
- [17] J.P. Holman. *Heat Transfer*. McGraw-Hill, New York, 7th edition, 1990. [61](#)
- [18] J.H. Klotz 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. [104](#)
- [19] R.C. Reid, J.M. Prausnitz, and B.E. Poling. *Properties of Gases and Liquids*. McGraw-Hill, New York, 4th edition, 1987. [117](#)
- [20] A. Tewarson. *SFPE Handbook of Fire Protection Engineering*, chapter Generation of Heat and Gaseous, Liquid, and Solid Products in Fires. National Fire Protection Association, Quincy, Massachusetts, fourth edition, 2008. [124](#), [125](#), [129](#)
- [21] 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. [126](#)
- [22] P.J. DiNenno, editor. *SFPE Handbook of Fire Protection Engineering*. National Fire Protection Association, Quincy, Massachusetts, 3rd edition, 2002. [161](#)
- [23] 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. [163](#)
- [24] 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. [164](#)
- [25] Pamela P. Walatka and Pieter G. Buning. PLOT3D User’s Manual, version 3.5. NASA Technical Memorandum 101067, NASA, 1989. [188](#)
- [26] G.W. Mulholland. *SFPE Handbook of Fire Protection Engineering*, chapter Smoke Production and Properties. National Fire Protection Association, Quincy, Massachusetts, 3rd edition, 2002. [189](#)
- [27] G.W. Mulholland and C. Croarkin. Specific Extinction Coefficient of Flame Generated Smoke. *Fire and Materials*, 24:227–230, 2000. [190](#)
- [28] M.L. Janssens and H.C. Tran. Data Reduction of Room Tests for Zone Model Validation. *Journal of Fire Science*, 10:528–555, 1992. [190](#)
- [29] 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. [191](#)
- [30] 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. [191](#)

- [31] 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. [195](#)
- [32] D.A. Purser. *SFPE Handbook of Fire Protection Engineering*, chapter Toxicity Assessment of Combustion Products. National Fire Protection Association, Quincy, Massachusetts, 3rd edition, 2002. [196](#), [197](#)
- [33] H. Werner and H. Wengle. Large-eddy simulation of turbulent flow over and around a cube in a plate channel. In *8th Symposium on Turbulent Shear Flows*, pages 155–168, Munich, Germany, 1991. Technische University Munich. [201](#)
- [34] S.B. Pope. Ten questions concerning the large-eddy simulation of turbulent flows. *New Journal of Physics*, 6:1–24, 2004. [202](#), [203](#)
- [35] L. Vervisch, P. Domingo, G. Lodato, and D. Veynante. Scalar energy fluctuations in Large-Eddy Simulation of turbulent flames: Statistical budgets and mesh quality criterion. *Combust. Flame*, 157:778–789, 2010. [202](#), [204](#)
- [36] 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. [202](#), [203](#), [205](#)
- [37] J. Bardina, J. H. Ferziger, and W. C. Reynolds. Improved subgrid-scale models for large-eddy simulation. *AIAA*, 80:1357, 1980. [203](#), [204](#)
- [38] Y. Nievergelt. *Wavelets Made Easy*. Birkhäuser, 1999. [204](#)
- [39] K. Schneider and O. Vasilyev. Wavelet methods in computational fluid dynamics. *Annu. Rev. Fluid Mech.*, 42:473–503, 2010. [204](#)
- [40] T. Korhonen and S. Hostikka. Fire Dynamics Simulator with Evacuation: FDS+Evac, Technical Reference and User’s Guide. VTT Working Papers 119, VTT Technical Research Centre of Finland, Espoo, Finland, 2009. [216](#), [217](#), [219](#), [223](#), [224](#), [225](#), [226](#), [231](#), [233](#), [235](#), [236](#)