NIST Special Publication 1019-5

Fire Dynamics Simulator (Version 5) User's Guide

Kevin McGrattan
Bryan Klein
Simo Hostikka
Jason Floyd

In cooperation with:
VTT Technical Research Centre of Finland

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

AUTHENTICATED
U.S. GOVERNMENT
INFORMATION
GPO

NIST Special Publication 1019-5

Fire Dynamics Simulator (Version 5) User's Guide

Kevin McGrattan
Bryan Klein
NIST Building and Fire Research Laboratory
Gaithersburg, Maryland, USA

Simo Hostikka
VTT Technical Research Centre of Finland
Espoo, Finland

Jason Floyd
Hughes Associates, Inc.
Baltimore, Maryland, USA

October 1, 2007
SVNRepository Revision : 726



U.S. Department of Commerce
Carlos M. Gutierrez, Secretary

National Institute of Standards and Technology
James M. Turner, Acting 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 1019-5
Natl. Inst. Stand. Technol. Spec. Publ. 1019-5, 186 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

Preface

This guide describes how to use the Fire Dynamics Simulator (FDS), Version 5. It does not provide the background theory. A companion document, called the FDS Technical Reference Guide [1], contains details about the governing equations and numerical methods. 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 "User's Guide for Smokeview Version 5" [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.

About the Authors

Kevin McGrattan is a mathematician in the Building and Fire Research 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.

Simo Hostikka is a Senior Research Scientist at VTT Technical Research Centre of Finland. He is the principal developer of the radiation and solid phase sub-models within FDS.

Jason Floyd is a Senior Engineer at Hughes Associates, Inc., in Baltimore, Maryland. He received a bachelors of science degree and a doctorate from 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 currently funded by NIST under grant 60NANB5D1205 from the Fire Research Grants Program (15 USC 278f). He is the principal developer of the multi-parameter mixture fraction combustion model and control logic within FDS.

Bryan Klein is an Information Technology Specialist in the Building and Fire Research Laboratory of NIST. Before coming to NIST, Bryan worked for five years with Western Fire Center, Inc., performing a wide range of activities including fire modeling, data acquisition programming, and quantitative fire measurements. His current focus is on FDS development and user support, along with experimental model validation work.

Acknowledgments

The Fire Dynamics Simulator, in various forms, has been under development for almost 25 years. However, the publicly released software has only existed since 2000. Since its first release, continued improvements have been made to the software based largely on feedback from its users. Included here are some who made important contributions.

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.

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 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.

Thanks to Chris Lautenburger and Carlos Fernandez-Pello for their assistance with the "two-reaction" test case.

Thanks to Ian Thomas, Khalid Moinuddin, and Ian Bennetts for their description of and data for the ethanol pan fire example. 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.

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

FDS 5 Beta Testers	
Nick Agnew	Maunsell, Australia
Camille Azzi	Universities of Glasgow and Strathclyde, Scotland
Matthew Bilson	Maunsell, Australia
George Braga	Federal District Fire Department, Brazil
Keith Calder	Senez Reed Calder Engineering, Canada
Steven Chi Heng Lam	Hoare Lea Fire Engineering, UK
Doo Chan Choi	Rolf Jensen & Associates, Inc., USA
Marco Cigolini	Italferr spa, Italy
John Cutonilli	Hughes Associates, Inc., USA
Sylvain Desanghere	CTICM (Centre Technique Industriel de la Construction Métallique), France
Montu L. Das	Gage-Babcock & Associates, USA and Canada
Franck Didieux	Laboratoire National de Métrologie et d'Essais (LNE), France
Johannes Dimyadi	AstraVision-Solutions, New Zealand
Bill Ferrante	Roosevelt Fire District, USA
Paul Fuss	NIST, USA
Andreas Gerndt	University of Louisiana, USA
Emanuele Gissi	Corpo Nazionale dei Vigili del Fuoco, Comando Prov. di Genova, Italy
Paul Hart	Swiss Re, GAP Services, USA
Hsiao, Li Kai (Gary)	Fire Bureau, Taipei, Taiwan
Hu Zhi-Xin	University of Maryland, USA
Ilya N. Karkin	SITIS Ltd., Russia
Susanne Kilian	hhpberlin, Fire Safety Engineers, Germany
Sung Chan Kim	School of Mechanical Engineering, Chung Ang University, Korea
Pierre-Louis Lamballais	Flashover-Backdraft, France
A. Leonardi	StIL (Studio di Ingegneria Leonardi), Italy
Davy Leroy	Arup Fire, UK
Jason Liu	Warrington Fire Research, Australia
Timothy Liu	Locke Carey Fire Consultants, UK
Dave McGill	Seneca College, Ontario, Canada
Ken Miller	Las Vegas Fire & Rescue, USA
Pete Muir	Safe Consulting, UK
Stephen Olenick	Combustion Science & Engineering, Inc., USA
Kristopher Overholt	University of Houston-Downtown, USA
PENG Wei	State Key Labortory of Fire Science, China
Andrew Purchase	Maunsell, Australia
Christian Rogsch	University of Wuppertal, Germany
Michael Roth	RWDI, Canada
Ahmed Salem	Alexandria University, Egypt
Robert Schmidt	Combustion Science & Engineering, Inc., USA
Joe Skaggs	CASE Forensics, USA
Piotr Smardz	Ahearne Fire Engineering Consultants, Ireland
Jamie Stern-Gottfried	Arup Fire, UK
Boris Stock	BFT Cognos Gmb, Germany
Blair Stratton	Beca, New Zealand

Csaba Szilagyi	Szolnok Fire Department, Hungary
Charlie Thornton	Thunderhead Engineering, USA
Sebastian Ukleja	University of Ulster, Northern Ireland
Giacomo Villi	Dipartimento Fisica Tecnica (DfT) UNIPd, Italy
Andreas Vischer	RWTH Aachen University, Germany
Karl Wallasch	Hoare Lea Fire Engineering, UK
Kaoru Wakatsuki	National Research Institute of Fire and Disaster, Japan
Yang Shan-You	State Key Laboratory of Fire Science, China
Robin Zevotek	C&S Engineers, Life Safety Services, Syracuse, New York, USA
	GIDAI, University of Cantabria, Spain

Contents

Preface i

Disclaimer iii

About the Authors v

Acknowledgments vii

I Running FDS 1

1 Introduction 3
1.1 Features of FDS . . . 3
1.2 What's New in FDS 5? . . . 4

2 Getting Started 7
2.1 How to Acquire FDS and Smokeview . . . 7
2.2 Computer Hardware Requirements . . . 7
2.3 Computer Operating System (OS) and Software Requirements . . . 8

3 Running FDS 9
3.1 Starting an FDS Calculation . . . 9
3.1.1 Starting an FDS Calculation (Single Processor Version) . . . 9
3.1.2 Starting an FDS Calculation (Multiple Processor Version) . . . 10
3.2 Monitoring Progress . . . 12

4 User Support 13
4.1 The Version Number . . . 13
4.2 Common Error Statements . . . 14
4.3 Support Requests and Bug Tracking . . . 15

II Writing an FDS Input File 17

5 The Basic Structure of an Input File 19
5.1 Naming the Job . . . 19
5.2 Namelist Formatting . . . 19
5.3 Input File Structure . . . 20

6 Setting the Bounds of Time and Space **23**
6.1 Naming the Job: The HEAD Namelist Group (Table 13.6) . . . 23
6.2 Simulation Time: The TIME Namelist Group (Table 13.24) . . . 23
6.3 Computational Meshes: The MESH Namelist Group (Table 13.11) . . . 25
6.3.1 Two-Dimensional and Axially-Symmetric Calculations . . . 25
6.3.2 Multiple Meshes and Parallel Processing . . . 26
6.3.3 Mesh Stretching: The TRNX, TRNY and/or TRNZ Namelist Groups (Table 13.25) . . . 28
6.3.4 Choosing the Right Mesh Dimensions . . . 29
6.4 Miscellaneous Parameters: The MISC Namelist Group (Table 13.12) . . . 31
6.4.1 Stopping and Restarting Calculations . . . 31
6.4.2 Special Topic: Defying Gravity . . . 32
6.4.3 Special Topic: Restoring the Baroclinic Vorticity . . . 32
6.4.4 Special Topic: Stack Effect . . . 33
6.4.5 Special Topic: Large Eddy Simulation Parameters . . . 33
6.4.6 Special Topic: Numerical Stability Parameters . . . 33
6.5 Special Topic: Unusual Initial Conditions: The INIT Namelist Group (Table 13.8) . . . 35
6.6 Special Topic: Setting Limits: The CLIP Namelist Group (Table 13.2) . . . 35

7 Building the Model **37**
7.1 Creating Obstructions: The OBST Namelist Group (Table 13.13) . . . 37
7.1.1 Non-rectangular Geometry and Sloped Ceilings . . . 38
7.2 Creating Voids: The HOLE Namelist Group (Table 13.7) . . . 39
7.3 Applying Surface Properties: The VENT Namelist Group (Table 13.26) . . . 40
7.3.1 Special VENTs . . . 41
7.3.2 Controlling VENTs . . . 41
7.3.3 Trouble-Shooting VENTs . . . 41

8 Boundary Conditions **43**
8.1 Basics . . . 43
8.2 Describing the Bounding Surfaces: The SURF Namelist Group (Table 13.22) . . . 44
8.2.1 Specifying a Fire with a Known Heat Release Rate . . . 44
8.2.2 Simple Thermal Boundary Conditions . . . 45
8.2.3 Velocity and Total Mass Flux Boundary Conditions . . . 45
8.2.4 Species and Species Mass Flux Boundary Conditions . . . 46
8.2.5 Special Topic: Fires and Flows in the Outdoors . . . 47
8.2.6 Special Topic: A Radially-Spreading Fire . . . 47
8.2.7 Special Topic: Non-Planar Walls and Targets . . . 48
8.3 Pressure-Related Effects: The ZONE Namelist Group (Table 13.26) . . . 49
8.3.1 Leaks . . . 49
8.3.2 Fans . . . 50
8.4 Describing Real Materials: The MATL Namelist Group . . . 52
8.4.1 Thermal Properties . . . 53
8.4.2 Pyrolysis Models . . . 53
8.4.3 Special topic: Making Fuels Disappear (BURN_AWAY) . . . 57
8.4.4 Special Topic: Initial and Backside Boundary Conditions . . . 58
8.4.5 Special Topic: Numerical Accuracy and Stability . . . 58
8.5 User-Specified Functions: The RAMP and TABL Namelist Groups . . . 59
8.5.1 Time-Dependent Functions . . . 59

8.5.2 Temperature-Dependent Functions . . . 60
8.5.3 Tabular Functions . . . 61
8.6 Coloring Obstructions, Vents, Surfaces and Meshes . . . 62
8.6.1 Texture Mapping . . . 62
8.7 Verifying the Solid Phase Properties . . . 64

9 Combustion and Radiation **65**
9.1 Mixture Fraction Combustion: The REAC Namelist Group . . . 65
9.1.1 Important Issues Related to the Mixture Fraction Models . . . 67
9.2 Extra Gas Species: The SPEC Namelist Group . . . 69
9.3 Finite-Rate Combustion . . . 71
9.4 Radiation Transport: The RADI Namelist Group . . . 71

10 Particles and Droplets: The PART Namelist Group **73**
10.1 Basics . . . 73
10.2 Controlling Particles and Droplets . . . 74
10.3 Particle and Droplet Properties . . . 75
10.4 Special Types of Particles and Droplets . . . 76
10.5 Coloring Particles and Droplets . . . 77
10.6 Special Topic: Droplet Fuel Sprays . . . 77
10.7 Special Topic: Suppression by Water (Mixture Fraction Model Only) . . . 78

11 Devices and Control Logic **79**
11.1 Device Location and Orientation: The DEVC Namelist Group (Table 13.4) . . . 79
11.2 Device Output . . . 80
11.3 Special Devices and their Properties: The PROP Namelist Group (Table 13.16) . . . 80
11.3.1 Sprinklers . . . 80
11.3.2 Nozzles . . . 82
11.3.3 Heat Detectors . . . 82
11.3.4 Smoke Detectors . . . 83
11.3.5 Beam Detection Systems . . . 83
11.3.6 Aspiration Detection Systems . . . 84
11.4 Basic Control Logic . . . 85
11.4.1 Creating and Removing Obstructions . . . 85
11.4.2 Activating and Deactivating Vents . . . 86
11.5 Advanced Control Functions: The CTRL Namelist Group . . . 88
11.5.1 Control Functions: ANY, ALL, ONLY, and AT_LEAST . . . 88
11.5.2 Control Function: TIME_DELAY . . . 89
11.5.3 Control Function: DEADBAND . . . 89
11.5.4 Control Function: RESTART and KILL . . . 90
11.5.5 Control Function: CUSTOM . . . 90
11.5.6 Combining Control Functions: A Pre-Action Sprinkler System . . . 91
11.5.7 Combining Control Functions: A Dry Pipe Sprinkler System . . . 91

12 Output Data **93**
12.1 Output Control Parameters: The DUMP Namelist Group . . . 93
12.2 Output Options . . . 94
12.2.1 Point Measurement Devices . . . 94

12.2.2 Integrated (non-pointwise) Measurement Devices . . . 94
12.2.3 Output Statistics . . . 95
12.2.4 Quantities within Solids: The PROF Namelist Group . . . 95
12.2.5 Animated Planar Slices: The SLCF Namelist Group . . . 95
12.2.6 Animated Boundary Quantities: The BNDF Namelist Group . . . 96
12.2.7 Animated Isosurfaces: The ISOF Namelist Group . . . 96
12.2.8 Plot3D Static Data Dumps . . . 96
12.3 Special Output Quantities . . . 98
12.3.1 Heat Release Rate . . . 98
12.3.2 Visibility and Obscuration . . . 98
12.3.3 Layer Height and the Average Upper and Lower Layer Temperatures . . . 99
12.3.4 The True Gas Temperature vs. the Measured Gas Temperature . . . 100
12.3.5 Heat Fluxes . . . 100
12.3.6 Droplet Output Quantities . . . 101
12.3.7 Interfacing with Structural Models . . . 101
12.3.8 Integrated Mass and Energy Fluxes through Openings . . . 101
12.3.9 Wind and the Pressure Coefficient . . . 102
12.4 Extracting Numbers from the Output Data Files . . . 102
12.5 Summary of Output Quantities . . . 103

13 Alphabetical List of Input Parameters **105**
13.1 BNDF (Boundary File Parameters) . . . 106
13.2 CLIP (MIN/MAX Clipping Parameters) . . . 106
13.3 CTRL (Control Function Parameters) . . . 107
13.4 DEVC (Device Parameters) . . . 108
13.5 DUMP (Output Parameters) . . . 109
13.6 HEAD (Header Parameters) . . . 109
13.7 HOLE (Obstruction Cutout Parameters) . . . 110
13.8 INIT (Initial Conditions) . . . 110
13.9 ISOF (Isosurface Parameters) . . . 110
13.10 MATL (Material Properties) . . . 111
13.11 MESH (Mesh Parameters) . . . 112
13.12 MISC (Miscellaneous Parameters) . . . 113
13.13 OBST (Obstruction Parameters) . . . 114
13.14 PART (Lagrangian Particles/Droplets) . . . 115
13.15 PROF (Wall Profile Parameters) . . . 116
13.16 PROP (Device Properties) . . . 117
13.17 RADI (Radiation Parameters) . . . 118
13.18 RAMP (Ramp Function Parameters) . . . 118
13.19 REAC (Reaction Parameters) . . . 119
13.20 SLCF (Slice File Parameters) . . . 120
13.21 SPEC (Species Parameters) . . . 120
13.22 SURF (Surface Properties) . . . 121
13.23 TABL (Table Parameters) . . . 122
13.24 TIME (Time Parameters) . . . 122
13.25 TRNX, TRNY, TRNZ (MESH Transformations) . . . 123
13.26 VENT (Vent Parameters) . . . 123
13.27 ZONE (Pressure Zone Parameters) . . . 124

14 Conversion of Old Input Files to FDS 5 **125**
14.1 Numerical Domain Parameters: GRID and PDIM . . . 125
14.2 Obstructions, Vents, and Holes: OBST, VENT, and HOLE . . . 125
14.3 Surface Parameters: SURF . . . 125
14.4 Reaction Parameters: REAC . . . 126
14.5 Device Parameters: SPRK, SMOD, HEAT, THCP . . . 127

III Sample Cases and Verification **129**

15 Forms of Verification **131**
15.1 Comparison with Analytical Solutions . . . 131
15.2 Code Checking . . . 132
15.3 Numerical Tests . . . 134

16 Verification Test Suite **135**
16.1 Hydrodynamics . . . 135
16.1.1 Axially-Symmetric Helium Plume (**helium_2d**) . . . 135
16.1.2 Pressure Rise in a Sealed Enclosure (**pressure_rise**) . . . 136
16.1.3 Leaks and Fans in a Sealed Enclosure (**leak_test** and **leak_test_2**) . . . 137
16.1.4 Two Fans in a Wall (**fan_test**) . . . 138
16.1.5 Stack Effect (**stack_effect**) . . . 139
16.1.6 Sawtooth (**sawtooth**) . . . 140
16.2 Combustion . . . 141
16.2.1 A Simple Under-Ventilated Compartment Fire (**door_crack**) . . . 141
16.3 Radiation . . . 142
16.3.1 Radiation inside a box (**radiation_in_a_box**) . . . 142
16.3.2 Radiation from a plane layer (**radiation_plane_layer**) . . . 143
16.4 Solid Phase Phenomena . . . 144
16.4.1 Simple Heat Conduction Through a Solid Slab (**heat_conduction**) . . . 144
16.4.2 Temperature-Dependent Thermal Properties (**heat_conduction_kc**) . . . 145
16.4.3 A Simple Two-Step Pyrolysis Example (**two_step_solid_reaction**) . . . 146
16.4.4 Wall Internal Radiation (**wall_internal_radiation**) . . . 147
16.4.5 A Liquid Pool Fire (**ethanol_pan**) . . . 148
16.4.6 A Thermoplastic (**thermoplastic**) . . . 149
16.4.7 A Charring Solid (**charring_solid**) . . . 150
16.4.8 Testing the "Burn-Away" Feature (**box_burn_away**) . . . 152
16.4.9 A Couch Fire (**couch**) . . . 153
16.4.10 Flame Spread along a Cable Tray (**cable_tray**) . . . 154
16.5 Detectors . . . 155
16.5.1 Aspiration Detector (**beam_detector**) . . . 155
16.5.2 Aspiration Detector (**aspiration_detector**) . . . 156
16.6 Droplets and Sprays . . . 157
16.6.1 Water Droplet Evaporation (**water_evaporation**) . . . 157
16.6.2 A Liquid Fuel Spray Burner (**spray_burner**) . . . 158
16.6.3 Measuring Water Flux (**bucket_test**) . . . 159
16.6.4 Complex Spray Patterns (**bucket_test_2**) . . . 159
16.7 General Functionality . . . 161

16.7.1 Creating and Removing HOLEs and OBSTructions (**create_remove**) 161

17 Sensitivity Analysis **163**
17.1 Grid Sensitivity 163
17.2 Sensitivity of Large Eddy Simulation Parameters 165
17.3 Sensitivity of Radiation Parameters 165
17.4 Sensitivity of Thermophysical Properties of Solid Fuels 166
17.5 Summary 167

IV Working with the FDS Source Code **169**

18 Compiling FDS **171**
18.1 FDS Source Code 171

19 Output File Formats **173**
19.1 Diagnostic Output 173
19.2 Plot3D Data 174
19.3 Device Output Data 174
19.4 Control Output Data 175
19.5 Gas Mass Data 175
19.6 Mixture Fraction State Relations 175
19.7 Slice Files 176
19.8 Boundary Files 176
19.9 Particle Data 177
19.10 Profile Files 177

Bibliography **179**

Index **183**

Part I

Running 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 the FDS Technical Reference Guide [1].

Smokeview is a separate visualization program that is used to display the results of an FDS simulation. A detailed description of Smokeview is found in the *User's Guide for Smokeview Version 5* [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 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 chemical reaction whose products are tracked via a two-parameter mixture fraction model. The mixture fraction is a conserved scalar quantity that represents the mass fraction of one or more components of the gas at a given point in the flow field. By default, two components of the mixture fraction are explicitly computed. The first is the mass fraction of unburned fuel and the second is the mass fraction of burned fuel (i.e. the mass of the combustion products that originated as fuel.). A two-step chemical reaction with a three parameter mixture fraction decomposition can also be used with the first step being oxidation of fuel to carbon monoxide and the second step the oxidation of carbon monoxide to carbon dioxide. The three mixture fraction components for the two step reaction are unburned fuel, mass of fuel that has completed the first reaction step, and the mass of fuel that has completed the second reaction step. The mass fractions of all of the major reactants and products can be derived from the mixture fraction parameters by means of "state relations,". Lastly, a multiple-step finite rate model is also available.

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 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).

1.2 What's New in FDS 5?

FDS 5 differs from previous versions in its treatment of solid boundaries and gas phase combustion. Among the more important changes are:

Multi-Step Combustion Previous versions of FDS have assumed only one gas phase reaction. Now, multiple-step reaction schemes are available to describe local extinction, CO production, among various other phenomena. The most important improvements to the combustion model are a more accurate heat release rate calculation, and a better treatment of local flame extinction.

Material Layers Past versions of FDS have assumed that solid boundaries consist of a single homogenous layer. Now, solid boundaries can be modeled with multiple layers of materials, with each material specified via a new namelist group called `MATL`. This change makes past input files obsolete.

Command Line Format FDS is still run from the command line, but the syntax is slightly different than in previous versions. See Section 3 for details.

Database Previous versions of FDS used a separate "database" file to store material and reaction parameters. This file is no longer available, and now all parameters must be specified within the input file.

Device Descriptions The method used to describe a device and/or sensor (Sprinkler, Heat Detector, Thermocouple, etc.) has changed. See Section 11.1 for more information on defining devices and their properties. Any device can be used to control sprinkler activation and the creation and removal of vents or obstacles.

Sprinklers The external sprinkler files used in previous versions are no longer used. All information about sprinklers and other fire-specific devices are conveyed in the input file. Sprinklers are now defined with the new method of describing devices mentioned above. See Section 11.1 for more information.

Control Functions A new group of input parameters is available to describe functions that control sprinkler activation, the creation and removal of vents or obstacles, and code execution (termination or dumping of restart files). See Section 11.5 for details.

Numerical Mesh Previous versions of FDS used separate input groups to define the numerical grid and the computational domain. Now the two groups have been merged into a single, simplified `MESH` namelist group. Namelist groups `PDIM` and `GRID` shall no longer be used in the input file. See Section 6.3 for more detail.

Pressure Zones It is possible in FDS 5 to declare individual regions in the computational domain to have background pressures different from ambient, allowing for calculations of leakage, fan curves, and so forth. See Section 8.3 for more details.

Stack Effect and Atmospheric Stratification Improvements have been made to better characterize a stratified atmosphere and the movement of air in a tall building due to temperature differences between inside and outside.

Adiabatic Surface Temperature A new output quantity has been added to facilitate using FDS output in thermal and mechanical finite element models. See Section 8.2.2 for more information.

Development, Distribution and Formal User Support Starting with FDS 5, the open-source development environment SourceForge.net is being used for configuration management (code archiving, revision tracking, bug fixes, user suggestions, and so on). See Section 2.1 for more information.

FDS Verification and Validation Guide Starting with FDS 5, more emphasis has been placed on maintaining a permanent collection of Verification and Validation cases. This improves the quality of each FDS update and release, as a standard test suite will now be used to insure that changes made to the source code do not degrade FDS output. This also provides users with a standard data set to verify their own installation of FDS and to compare the results that FDS is returning on their system to published data.

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` for more information. 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 CPU 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.

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 Mac OS X Tiger FDS .zip archive is available for both the PowerPC and Intel architectures. 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).

Unix, Linux Unix, Linux users can run FDS and Smokeview by downloading the appropriate pre-compiled executables and installing them wherever they see fit. If the pre-compiled FDS executable does not work (usually because of library incompatibilities), the FDS source code can be downloaded and compiled using a Fortran 90 and C compiler (See Appendix 18 for details). If Smokeview does not work on the Linux or Unix workstation, you should use a Windows or Mac PC to view FDS output.

FDS in Parallel For those wishing to run FDS in parallel, MPI (Message Passing Interface) must be installed on each of the computers within the network that will be used for FDS computations. Information about installing MPI on different computer platforms is given on the FDS website. See the Development section of the website for more information.

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 Windows executable is called **fds#.exe**. The parallel executable is called **fds#_mpi.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. Assuming that an input file called **job_name.fds** exists in some directory, run the program either in a DOS or Unix command prompt as follows:

MS Windows

Open up a Command Prompt window, and change directories ("cd") to where the input file for the case is located, then run the code by typing at the command prompt

```
fds5 job_name.fds
```

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

```
fds5 job_name.fds > job_name.err
```

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 `fds5` is aliased to its full path name. You may also need to "chmod + x" to make the file executable. Once this is done, run FDS from the command line by typing:

```
fds5 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:

```
fds5 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 difficult 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 process 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) [3] to allow multiple computers to run a single FDS job. Actually, the job must be broken up into multiple meshes, and a processor is assigned to work on each mesh. Each processor runs an FDS job (called a thread) for its given mesh, and the MPI handles the transfer of information between meshes. 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 thread 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\fds5_mpi.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.

LAM-MPI

On the Linux cluster in the Building and Fire Research Lab at NIST, LAM-MPI, a free implemenation from Indiana University, is installed.[1] With LAM/MPI, the computers to be used are linked prior to the actual execution of FDS with a separate command called a "lamboot." FDS is then run using the command

```
mpirun -np 5 fds5_mpi job_name.fds
```

where the 5 indicates that 5 processors are to be used. In this case, the executable **fds5_mpi** is located in the working directory. To make the process run in the background

```
mpirun -np 5 fds5_mpi 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.

[1] http://www.lam-mpi.org

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 6.4.1. 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 11.5.

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 a serious calculation that pushes 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 much 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. The examples described in Part III should help you to develop larger, more complicated simulations from smaller building blocks.

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, feature requests 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 dramatically 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:

```
fds5
```

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.

Simply 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 user-prescribed 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 1'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 point 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, 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 the user's 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 Windows PC with 4 GB RAM, only 2 GB will be available to FDS, based on user feedback.

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=..." 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 SourceForge Support 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.

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.

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 input text[1] 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 a good idea to use commas or line breaks. Some machines do not recognize the spaces. 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.

[1] ASCII – American Standard Code for Information Interchange

[2] A *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','INERT','INERT'
```

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.

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 file (WTC_05_v5.fds) removed for clarity.

```
&HEAD CHID='WTC_05_v5', TITLE='WTC Phase 1, Test 5, FDS version 5' /
&MESH IJK=90,36,38, XB=-1.0,8.0,-1.8,1.8,0.0,3.82 /
&TIME T\_END=5400. /
```

```
&MISC SURF_DEFAULT='MARINITE BOARD', TMPA=20., POROUS_FLOOR=.FALSE. /
&DUMP NFRAMES=1800, DT_HRR=10., DT_DEVC=10., DT_PROF=30. /

&REAC ID         = 'HEPTANE TO CO2'
      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 strongly 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.

Table 5.1: Namelist Group Reference Table

Group Name	Namelist Group Description	Reference Section	Parameter Table
BNDF	Boundary File Output	12.2.6	13.1
CTRL	Control Function Parameters	11.5	13.3
DEVC	Device Parameters	11.1	13.4
DUMP	Output Parameters	12.1	13.5
HEAD	Input File Header	6.1	13.6
HOLE	Obstruction Cutout	7.2	13.7
INIT	Initial Condition	6.5	13.8
ISOF	Isosurface File Output	12.2.7	13.9
MATL	Material Property	8.4	13.10
MESH	Mesh Parameters	6.3	13.11
MISC	Miscellaneous	6.4	13.12
OBST	Obstruction	7.1	13.13
PART	Lagrangian Particle	10	13.14
PROF	Profile Output	12.2.4	13.15
PROP	Device Property	11.3	13.16
RADI	Radiation	9.4	13.17
RAMP	Ramp Profile	8.5	13.18
REAC	Reactions	9.1	13.19
SLCF	Slice File Output	12.2.5	13.20
SPEC	Species Parameters	9.2	13.21
SURF	Surface Properties	8.2	13.22
TIME	Simulation Time	6.2	13.24
TRNX	Mesh Stretching	6.3.3	13.25
VENT	Vent Parameters	7.3	13.26
ZONE	Pressure Zone Parameters	8.3	13.27

Chapter 6

Setting the Bounds of Time and Space

6.1 Naming the Job: The HEAD Namelist Group (Table 13.6)

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 can help organize the effort. The namelist group HEAD contains two parameters, as in this example:

```
&HEAD CHID='WTC_05_v5', TITLE='WTC Phase 1, Test 5, FDS version 5' /
```

CHID is a string of 30 characters or less used to tag the output files. If, for example, CHID='WTC_05_v5', 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.

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 13.24)

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. Usually, only the duration of the simulation is required on this line, via the parameter T_END (Time End). The default is 1 s. Note: TWFINTWFIN will still work but it has been deprecated, it is recommended that T_END be used now instead.

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 timeline to start at a number other than zero, you can ues the parameter T_BEGIN (Time Begin) to specify the time written to file for the first timestep. This would be useful for matching timelines of experimental data or or video recordings.

This does not run any of the simulation prior to the T_BEGIN value. It is only used to offset the reported starting time from zero.

Time based RAMPs are evaluated using the actual time if the RAMP activation time is the same as T_BEGIN; otherwise, the 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 were 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.

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.

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 6.3.2. The default value of SYNCHRONIZE is .TRUE.

6.3 Computational Meshes: The MESH Namelist Group (Table 13.11)

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. 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 6.3.3).

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 either.

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
```

6.3.1 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 the V&V Guide.

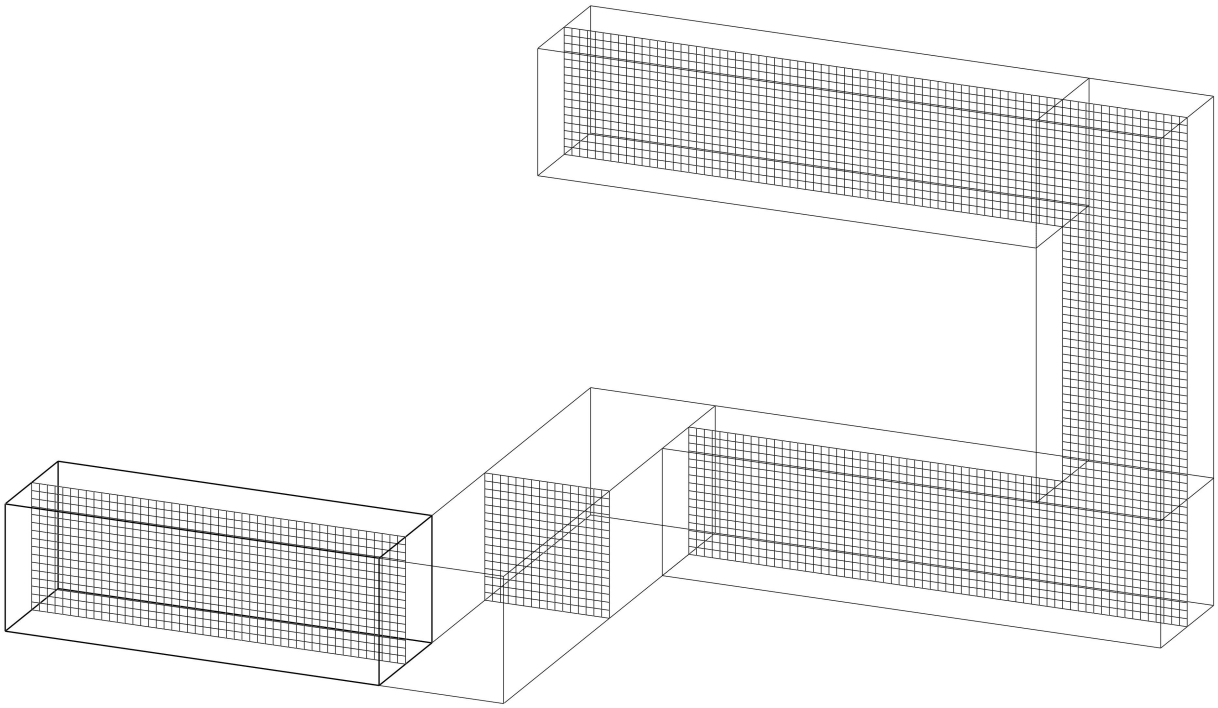


Figure 6.1: **An example of a multiple-mesh geometry.**

6.3.2 Multiple Meshes and Parallel Processing

The term "multiple meshes" means that the computational domain consists of more than one computational mesh, usually connected although this is not required. In each mesh, the governing equations can 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. **To run FDS in parallel, you need to break up the computational domain into multiple meshes so that each processor receives one mesh to work on.** 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. Here is a list of guidelines and warnings about the use of multiple meshes.

- 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.

- 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.

- Information from other meshes is received only at the exterior boundary of a given mesh. 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.

- 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.

- 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 7.3 for more details.

- It is possible, starting with FDS 5, to have a background pressure rise in multiple pressure zones, even if the pressure zones cross mesh boundaries. See Section 8.3 for more information.

- In a parallel calculation, you can force the time steps in all meshes to be the same by setting `SYNCHRONIZE=.TRUE.` on the `TIME` line.

Starting in FDS 5, this is the default for all modes of operation.

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.

- 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.

- 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.

6.3.3 Mesh Stretching: The TRNX, TRNY and/or TRNZ Namelist Groups (Table 13.25)

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 co-

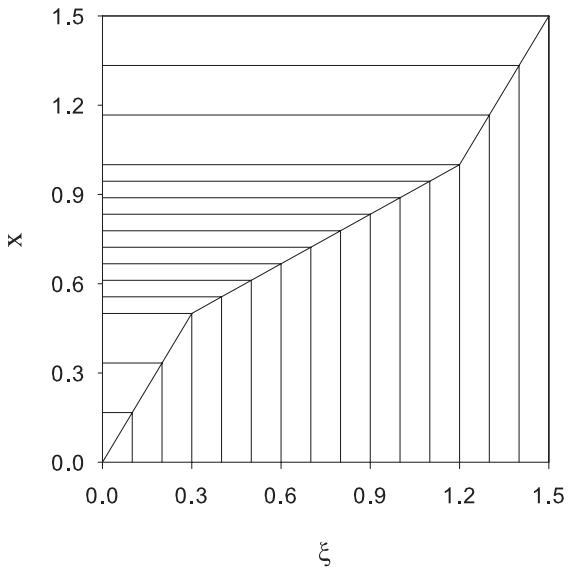


Figure 6.2: **Piecewise-Linear Transformation.**

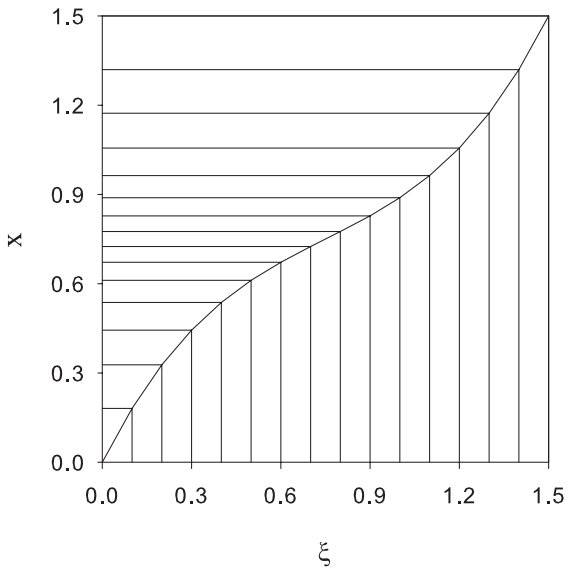


Figure 6.3: **Polynomial Transformation.**

ordinate 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.

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) $0 \le \xi \le 1.5$ to the *Physical Coordinate* (PC) $0 \le x \le 1.5$. 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 6.2 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

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

Figure 6.3 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.

6.3.4 Choosing the Right Mesh Dimensions

A common question asked by new FDS users is, "What size mesh should I use?" 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. Formally, 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

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

and δx is the nominal size of a mesh cell[1]. 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

[1]The 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.

spanning the fire, the better the resolution of the calculation. It is better to assess the quality of the mesh in terms of this non-dimensional parameter, rather than an absolute mesh cell size. For example, a cell size of 10 cm may be "adequate," in some sense, for evaluating the spread of smoke and heat through a building from a sizable fire, but may not be appropriate to study a very small, smoldering source.

As an example, in the mesh sensitivity study for NUREG 1824 [4], the $D^*/\delta x$ values ranged from 4 to 16. These values were used to adequately resolve plume dynamics, along with other geometrical characteristics of the models as well. This range does not indicate what values to use for all models, only what values worked well for that particular set of models.

6.4 Miscellaneous Parameters: The MISC Namelist Group (Table 13.12)

MISC is the namelist group of global miscellaneous input parameters. Only one MISC line should be entered in the data file. For example, the input line

```
&MISC SURF_DEFAULT='CONCRETE',TMPA=25. /
```

establishes that all bounding surfaces are to be made of CONCRETE unless otherwise specified, and that the ambient temperature is 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).

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

HUMIDITY Relative humidity, in units of %. This need only be specified if there is a source of water in the simulation other than the fire itself. Otherwise, water vapor is not explicitly tracked. Default 40 %.

ISOTHERMAL A logical parameter that indicates that the calculation does not involve any changes in temperature or radiation heat transfer, thus reducing the number of equations that must be solved, and simplifying those that are. Automatically sets RADIATION to .FALSE.

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.

P_INF Background pressure (at the ground) in Pa. The default is 101325 Pa.

RADIATION A logical parameter indicating whether radiation transport ought to be calculated. The default is .TRUE.

SUPPRESSION A logical parameter indicating whether FDS should include gas phase flame extinction. The default is .TRUE.

SURF_DEFAULT The SURF line that is to be applied to all boundaries, unless otherwise specified. The default is 'INERT'.

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.

6.4.1 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 11.5) 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.

6.4.2 Special Topic: Defying Gravity

Most users of FDS assume that the acceleration of gravity points toward the negative end of the z axis, 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, 0.0, -9.81 (units are m/s^2). This is the default, but it can be changed to be any direction.

Note: if sprinklers are specified, the gravity vector must not be changed. Much of the logic governing the trajectories of water droplets over solid objects assumes that gravity points in the negative z direction.

There are a few special applications where you might want to vary the gravity vector in time, for instance, for spacecraft applications. The gravity vector, GVEC, can be made a function of time using "ramps" for the individual components, RAMP_GX, RAMP_GY, and RAMP_GZ, all specified on the MISC line. More about RAMPs can be found in Section 8.5.

6.4.3 Special Topic: Restoring the Baroclinic Vorticity

There is an approximation made when solving for the pressure where it is assumed that

$$\nabla \cdot \frac{1}{\rho} \nabla \tilde{p} = \frac{1}{\rho_\infty} \nabla^2 \tilde{p} \tag{6.2}$$

The consequence of this approximation is that the vorticity generated due to the non-alignment of the density and pressure gradients, or the baroclinic torque, is neglected. For most large scale applications, the assumption is justified by the fact that the vorticity generated by buoyancy is the dominant source of vorticity. By

neglecting the baroclinic torque the solution of the elliptic partial differential equation obtained by taking the divergence of the momentum equation is greatly simplified. However, an option exists in the code to restore the baroclinic torque by decomposing the relevant term in the pressure equation

$$\nabla\cdot\frac{1}{\rho^n}\nabla\tilde{p}^n=\nabla\cdot\frac{1}{\overline{\rho}^n}\nabla\tilde{p}^n+\nabla\cdot\left(\frac{1}{\rho^n}-\frac{1}{\overline{\rho}^n}\right)\nabla\tilde{p}^{n-1} \tag{6.3}$$

and evaluating the second term on the right hand side with values of pressure from the previous time step. The expression $\overline{\rho}$ is an average density, equal to $2\rho_{\min}\rho_{\max}/(\rho_{\min}+\rho_{\max})$. To make this correction, simply include the statement `BAROCLINIC=.TRUE.` on the `MISC` line. In a DNS calculation (`DNS=.TRUE.`), the correction is made by default. However, for an LES calculation (the default mode in FDS), the correction must be explicitly invoked. The cost of the correction is not prohibitive – try calculations with and without the correction to determine if its inclusion is warranted.

6.4.4 Special Topic: Stack Effect

Tall buildings often experience buoyancy-induced air movement due to temperature differences between inside and outside, known as *stack effect*. 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. Experiment with different meshing strategies before including any fire or HVAC functionality. Slowly build in complexity. See Section 16.1.5 for an example.

6.4.5 Special Topic: Large Eddy Simulation Parameters

In default mode, FDS uses the Smagorinsky form of Large Eddy Simulation (LES) to model subgrid-scale turbulence. The viscosity μ is modeled

$$\mu_{\text{LES}}=\rho\,(C_s\Delta)^2\left(2\,\overline{\mathbf{S}}_{ij}\cdot\overline{\mathbf{S}}_{ij}-\frac{2}{3}(\nabla\cdot\overline{\mathbf{u}})^2\right)^{\frac{1}{2}} \tag{6.4}$$

where C_s is an empirical constant and Δ is a length on the order of the size of a grid cell. The bar above the various quantities denotes that these are the resolved, or *filtered*, values, meaning that they are computed on a numerical grid. The other diffusive parameters, the thermal conductivity and material diffusivity, are related to the turbulent viscosity by

$$k_{\text{LES}}=\frac{\mu_{\text{LES}}\,c_p}{\text{Pr}_t}\quad;\quad(\rho D)_{l,\text{LES}}=\frac{\mu_{\text{LES}}}{\text{Sc}_t} \tag{6.5}$$

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 $C_s=0.2$, $\text{Pr}_t=0.5$, and $\text{Sc}_t=0.5$ via the parameters `CSMAG`, `PR`, and `SC` on the `MISC` line. A more detailed discussion of these parameters is given in the FDS Technical Reference Guide [1].

6.4.6 Special Topic: Numerical Stability Parameters

The time step of an FDS simulation is constrained by the convective and diffusive transport speeds via two conditions. The first is known as the Courant-Friedrichs-Lewy (CFL) condition. The CFL condition asserts that the solution of the equations cannot be updated with a time step larger than that which would allow a

parcel of fluid to travel further than a single mesh cell. In each mesh cell of dimension δx by δy by δz with velocity components u, v, and w, the CFL number is defined:

$$\mathrm{CFL} = \delta t \ \max\left(\frac{|u|}{\delta x}, \frac{|v|}{\delta y}, \frac{|w|}{\delta z}\right) \tag{6.6}$$

Every time step, the CFL number is computed in each mesh cell, and the time step, δt, is adjusted if the maximum value of the CFL number is not between `CFL_MIN` and `CFL_MAX`, whose default values are 0.8 and 1.0, respectively. These values are included in the `MISC` namelist group.

A similar condition, but one constraining the time step when diffusive transport dominates, is sometimes called the Von Neumann condition. The Von Neumann number is defined:

$$\mathrm{VN} = 2 \max\left(\nu, D, \frac{k}{\rho c_p}\right) \ \delta t \left(\frac{1}{\delta x^2} + \frac{1}{dy^2} + \frac{1}{\delta z^2}\right) \tag{6.7}$$

Like the CFL number, VN is computed in each mesh cell, and the time step is adjusted if VN is outside the range between `VN_MIN` and `VN_MAX`, which are 0.8 and 1.0 by default. Note that this constraint is applied to the momentum, mass and energy equations via the relevant diffusion parameter – viscosity, material diffusivity or thermal conductivity. This constraint on the time step is typical of any explicit, second-order numerical scheme for solving a parabolic partial differential equation. To save CPU time, the Von Neumann criterion is only invoked for DNS calculations or for LES calculations with mesh cells smaller than 5 mm.

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

6.5 Special Topic: Unusual Initial Conditions: The INIT Namelist Group (Table 13.8)

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 MASS_FRACTION(N) where N indicates the Nth species listed in the input file.

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

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: Setting Limits: The CLIP Namelist Group (Table 13.2)

On rare occasions you might need to set upper or lower bounds on the density, temperature, or species mass fractions. The parameters listed in Table 13.2 are for diagnostic purposes only.

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 these objects. 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. The next chapter describes how to assign properties to the boundaries.

7.1 Creating Obstructions: The OBST Namelist Group (Table 13.13)

The namelist group OBST contains parameters used to define obstructions. 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 8.2) 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'. 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.
- Unlike earlier versions of FDS, 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 11.4.1 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 `OBST`ruction can be made impenetrable to a `HOLE` using this phrase. By default, `PERMIT_HOLE=.TRUE.`, meaning that an `OBST`ruction is assumed to be penetrable unless otherwise directed.
- 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.
- Obstructions are drawn solid in Smokeview. To draw an outline representation, set `OUTLINE=.TRUE.`

7.1.1 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.

7.2 Creating Voids: The HOLE Namelist Group (Table 13.7)

The HOLE namelist group is used to define parameters (Table 13.7) 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 predetermined times. See Section 11.4.1 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 it is desired that the obstruction(s) to be cut out should have a different color than the original obstruction, set the COLOR or integer triplet RGB on the HOLE line (see Section 8.6).

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 0-1. 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 16.7.1 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.

7.3 Applying Surface Properties: The VENT Namelist Group (Table 13.26)

Whereas the OBST group is used to specify obstructions within the computational domain, the VENT group (Table 13.26) is used to prescribe planes adjacent to obstructions or external walls. 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.

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.

Unlike previous versions of FDS, you can no longer specify a free-standing fan using the VENT construct. A VENT must always be attached to a solid obstruction. See Section 8.3 for instructions on specifying different types of fans.

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 8.6 and Table 8.1. Also, the parameter OUTLINE=.TRUE. causes the VENT to be drawn as an outline in Smokeview.

7.3.1 Special VENTs

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

An OPEN VENT

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'. 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 6.4.4 for more details.

As with exterior temperature, to change the exterior mass fraction of a particular gas species, set MASS_FRACTION(N) on the VENT line, where N denotes the species index. See Section 9.2 for more information about gas species.

Vents to the outside of the computational domain (OPEN vents) may **not** be opened or closed during a simulation. See Section 11.4.2 for details.

A MIRROR VENT

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.

7.3.2 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 11.4.2 for details.

7.3.3 Trouble-Shooting VENTs

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.

Chapter 8

Boundary Conditions

This chapter describes how to specify the properties of the objects that make up the bounding surfaces of the flow domain. **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 8.2). Starting in FDS 5, solids are assumed to consist of layers which can be made of different materials. The properties of each material required are designated via the `MATL` namelist group (Section 8.4). 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 Describing the Bounding Surfaces: The SURF Namelist Group (Table 13.22)

SURF is the namelist group (Table 13.22) 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 cold, inert wall. 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, 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, CONCRETE for example, set SURF_DEFAULT='CONCRETE' on the MISC line.

The default boundary condition INERT does allow for heat loss and is not the same as an adiabatic surface. If you wish to define a surface as adiabatic, then you should set ADIABATIC=.TRUE. on the SURF line.

8.2.1 Specifying a Fire with a Known 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 Section 8.5 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 kg/m^2/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 parameters on the REAC line. All of the species associated with the combustion process are accounted for by way of the mixture fraction variable and should not be explicitly prescribed. The exception to this rule is where a non-reacting gas is introduced into the domain that merely serves as a diluent, like CO_2 from an extinguisher or H_2O from evaporated sprinkler droplets (see Section 9.2 for details). If a finite rate combustion model is desired instead of the default mixture fraction model, see Section 9.3.

Specifying HRRPUA or MLRPUA automatically invokes the mixture fraction combustion model.

8.2.2 Simple Thermal Boundary Conditions

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 simply specify a fixed temperature boundary condition, in which case set TMP_FRONT to be the surface temperature in units of °C. For a solid surface of fixed convective heat flux, set CONVECTIVE_HEAT_FLUX to be the convective heat flux in units of kW/m^2. If CONVECTIVE_HEAT_FLUX is positive, the wall heats up the surrounding gases. If CONVECTIVE_HEAT_FLUX is negative, the wall cools the surrounding gases. The radiative heat flux can be specified only by setting both TMP_FRONT and EMISSIVITY.

Fixed temperature or fixed 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.

It is assumed that the innermost layer backs up to an air gap at ambient temperature (like a sheet of gypsum board attached to wood studs), or it backs up to an insulated material in which case no heat is lost to the backing material, or it backs up to the room on the other side of the wall. By default, it is assumed that the wall liner backs up to an air gap (BACKING='VOID'). If the wall liner is assumed to back up against an insulating material, like a sheet of steel attached to a fiber insulating board, the expression BACKING='INSULATED' on the SURF line prevents any heat loss from the back side of the material.

Finally, if it is desired that the heat transfer through the wall into the space behind the wall, the attribute BACKING='EXPOSED' should be listed. 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 the void.

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., nothing else. FDS will compute a wall temperature so that the sum of the convective and radiative fluxes is zero.

8.2.3 Velocity and Total Mass Flux Boundary Conditions

Velocity boundary conditions affect both the normal and tangential components of the velocity vector at boundaries. The normal component of velocity is controlled by the parameter VEL. If VEL is negative, the flow is entering the computational domain. If VEL is positive, the flow is exiting the domain. Sometimes it is desired that a given volume flux through a vent be prescribed rather than a velocity. If this is the case then VOLUME_FLUX can be prescribed instead of VEL. The units are m^3/s. If the flow is entering the computational domain, VOLUME_FLUX should be a negative number.

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.

The dimensions of the vent that are prescribed usually change because the prescribed vent dimensions are sometimes altered so that the vent edges line up with mesh cells. Also note that a SURF group 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='LOUVER', VOLUME_FLUX=-5.0, VEL_T=2.0,-1.0, COLOR='GREEN' /
&OBST XB=..., SURF_ID6='BRICK','LOUVER','BRICK','BRICK','BRICK','BRICK' /
```

dictates that the forward x-facing surface of the obstruction is to have a velocity equal to 5 m/s divided by the area of the face (as approximated within FDS) flowing into the computational domain.

Finally, 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 the code.

As an example, a simple blowing vent would be described by the line

```
&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.

At other times the user may wish that a specific flux of mass, be added or removed at a vent. This can be accomplished by specifying the quantity MASS_FLUX_TOTAL. MASS_FLUX_TOTAL uses the same sign convention as VEL. 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.

The tangential velocity boundary condition controls how the gas "sticks" to a solid surface. In theory, the tangential component of velocity is zero at the surface, but increases rapidly through a narrow region called the boundary layer . For most practical problems, the mesh is not fine enough to resolve the boundary layer, which is typically a few millimeters thick. For this reason, in an LES calculation, the velocity at the wall is set to be a fraction of its value in the mesh cell adjacent to the wall. Only in a DNS calculation is the velocity at the wall set to zero. To alter these defaults, set a parameter called SLIP_FACTOR. This parameter ranges from -1 to 1. If a no-slip wall is desired, SLIP_FACTOR=-1. If a free-slip wall is desired, SLIP_FACTOR=1. Numbers in between -1 and 1 can represent partial slip conditions, which may be appropriate for simulations involving large mesh cells. (Default SLIP_FACTOR is 0.5 for LES, -1.0 for DNS)

In the case of a blowing vent (or even a solid surface), it is possible to prescribe both the normal and tangential components of the flow (or just the tangential). 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. For example, the line

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

is a boundary condition for a louver vent that pushes air into the space with a normal velocity of 1.2 m/s, and with 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.

8.2.4 Species and Species Mass Flux Boundary Conditions

There are two species boundary conditions that can be specified (see Section 9.2 for details on inputting and using species). These boundary conditions are MASS_FLUX(N) and MASS_FRACTION(N) where N refers to a given species is via its place in the input file. For example, the second listed species is N=2. If a simple no-flux condition is desired at a solid wall, do not set anything. If the mass fraction of the Nth species is to be some value at a forced flow boundary (VEL or MASS_FLUX_TOTAL) set MASS_FRACTION(N) equal to the desired mass fraction on the appropriate SURF line. If the mass flux of the Nth species is desired, set MASS_FLUX(N) instead of MASS_FRACTION(N). If MASS_FLUX(N) is set, no VEL should be set. It is automatically calculated based on the mass flux. The inputs MASS_FLUX(N) (and typically MASS_FRACTION(N)) should only be used for inflow boundary conditions. MASS_FLUX(N) should be positive with units of kg/m^2/s.

Note that specifying MASS_FRACTION(N), 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(N)

8.2.5 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.

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 °C/m. A negative sign indicates that the temperature *decreases* with height. This need only be set for outdoor calculations where the height of the domain is tens or hundreds of meters. The default value of the LAPSE_RATE is 0 °C/m.

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.

8.2.6 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 in a special way. If the SURF_ID associated with a VENT defines a specified heat release rate, HRRPUA, and time history, RAMP_Q or TAU_Q, you can also specify XYZ and SPREAD_RATE on the VENT line. Then the fire is directed to start at the point XYZ and spread radially at a rate of SPREAD_RATE (m/s). The ramp-up 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.

8.2.7 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 Pressure-Related Effects: The ZONE Namelist Group (Table 13.26)

The basic FDS equation set assumes pressure to be composed of a "background" component, $\overline{p}(z,t)$, plus a perturbation, $\tilde{p}(\mathbf{x},t)$. Most often, $\overline{p}$ is just the hydrostatic pressure, and $\tilde{p}$ is the flow-induced pressure field that FDS calculates at each time step. Originally (FDS v. 1-4), it was possible to create a single, sealed compartment whose walls conformed to the exterior of the computational domain. A fire or fan could increase (or decrease) the background pressure in this single compartment, and a leakage area could be defined between the compartment and the ambient exterior. Flow through the "cracks" was simply a function of the background pressure via the usual empirical rules. This idea has been generalized starting in FDS 5. Now, you can specify any number of sealed portions of the computational domain to have their own "background" pressures, and these zones can be connected via leakage and duct paths.

There are several restrictions to assigning pressure zones. First, the pressure zones must be completely surrounded by obstructions (an external boundary also suffices). Second, the obstructions cannot be removed during the calculation. In other words, the door or window cannot suddenly fly open (equivalently, there must be no OPEN vents if one of the zone boundaries is an external boundary of the computational domain). Third, the pressure zones **can** span multiple meshes, but check the pressure in each mesh to ensure consistency.

8.3.1 Leaks

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

$$\dot{V}_{\text{leak}} = A_L \, \text{sign}(\Delta p) \sqrt{2\frac{|\Delta p|}{\rho_\infty}} \tag{8.1}$$

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. Leakage is inherently a submesh-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 /
```

The first line designates a region of the computational domain to be "Pressure Zone" 1. 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 verse) with a leak area of 0.0002 m^2. At least one of the obstructions that form the walls of Zone 1 must have 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 LEAK_PATH=1,2. This tells FDS to lump all of the leakage over these surfaces. The order of the pressure zones designated by LEAK_PATH is unimportant.

8.3.2 Fans

In Section 8.2 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 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:

$$\dot{V}_{\text{fan}} = A_{\text{duct}} U_{\text{max}} \, \text{sign}(\Delta p_{\text{max}} - \Delta p) \sqrt{\frac{|\Delta p - \Delta p_{\text{max}}|}{\Delta p_{\text{max}}}} \tag{8.2}$$

where A_{duct} is the area of the duct (m^2), U_{max} is the air velocity (m/s), and Δp_{max} is the maximum pressure difference the fan can operate upon. Figure (8.1 displays a typical fan curve.

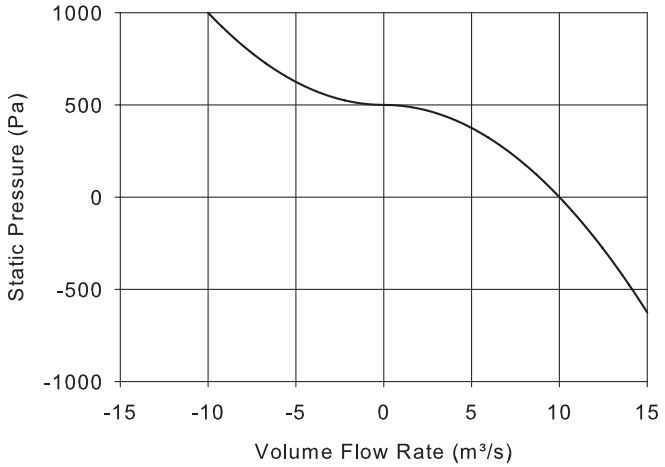


Figure 8.1: **Fan curve corresponding to** `VOLUME_FLUX=10` **and** `MAX_PRESSURE=500`**.**

The ideal velocity of the fan, U_{max}, is specified via the parameter `VEL` on the appropriate `SURF` line. Alternatively, the volume flow rate, $A_{\text{duct}} U_{\text{max}}$, can be specified using `VOLUME_FLUX`. Do not use both. These parameters were already introduced in Section 8.2. To simulate the behavior of a real fan, a few extra parameters need to be specified. To set Δp_{max}, the maximum operating over-pressure, add `MAX_PRESSURE` to the `SURF` line. Note that `MAX_PRESSURE` should always be positive and in units of Pa. If in the simulation the computed pressure exceeds the specified `MAX_PRESSURE`, then there will be a backflow in the duct.

Here is an example how fans can be specified. The actual case (**fan_test**) is included in the V&V Guide. In it, two simple compartments share a common wall. Both compartments are considered as separate "pressure zones." Two fans are mounted in the Partition Wall, blowing in opposite directions.

```
&SURF ID='BLOW LEFT',  POROUS=.TRUE., VEL=-0.2, DUCT_PATH=1,2, MAX_PRESSURE=1000. /
&SURF ID='BLOW RIGHT', POROUS=.TRUE., VEL= 0.4, DUCT_PATH=2,1, MAX_PRESSURE=1000. /

&ZONE XB=-3.0, 0.0,-1.0, 1.0, 0.0, 2.0 /  Pressure Zone 1
&ZONE XB= 0.0, 3.0,-1.0, 1.0, 0.0, 2.0 /  Pressure Zone 2

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

&HOLE XB=-0.1, 0.1,-0.1, 0.1, 0.4, 0.6 /
&OBST XB= 0.0, 0.0,-0.1, 0.1, 0.4, 0.6, ..., SURF_ID='BLOW RIGHT', PERMIT_HOLE=.FALSE. /
```

```
&HOLE XB=-0.1, 0.1,-0.1, 0.1, 1.4, 1.6 /
&OBST XB= 0.0, 0.0,-0.1, 0.1, 1.4, 1.6, ..., SURF_ID='BLOW LEFT', PERMIT_HOLE=.FALSE. /
```

Consider a few of the extra parameters. The attribute `POROUS=.TRUE.` allows hot, smokey gases to pass through the obstructions that represent the fans. These obstructions are merely flat plates, by necessity. The velocity `VEL` associated with a `POROUS` surface is meant to represent the velocity in the positive or negative coordinate direction, as indicated by its sign. This is different than the convention used when the `SURF` is attached to a solid wall. The `DUCT_PATH` defines the pressure `ZONE` downstream and upstream of the fan, respectively. The fan with `ID='BLOW LEFT'`, for example, blows air into `ZONE` 1 from `ZONE` 2. In more complicated scenarios, it is possible to tie the fan behavior to disconnected compartments, where it is assumed that a "virtual" duct connects the two spaces. The `HOLE`s in the Partition Wall serve only to carve out space for the obstructions that represent the fans. Note the obstructions have zero thickness, as required by the `POROUS` surface. The attribute `PERMIT_HOLE=.FALSE.` tells FDS not to reject the obstructions because they are embedded within the Partition Wall.

8.4 Describing Real Materials: The MATL Namelist Group

A solid boundary can consist of multiple layers of different materials, and each layer can consist of multiple material components. These combinations of layers and material 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.

Note also in this example that the BRICK WALL is not symmetric. Be careful when applying this SURF line to an entire obstruction, because the attribute EXPOSED implies that the back surface of the obstruction is different than the front surface.

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

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'
      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
```

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

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

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/m^3. The fact that the wall

density is smaller than the density of pure brick may be confusing, but can be explained easily. If the wall can contain water, the whole volume of the wall can not be pure brick. Instead there are voids (pores) that are filled with water. If the water is taken away, there is only about 1476 kg/m^3 of brick left. To have a density of 1600 kg/m^3 for a partially void wall, a higher density should be used for the pure brick.

8.4.1 Thermal Properties

For any solid material, specify its thermal CONDUCTIVITY (W/m·K), DENSITY (kg/m^3), 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.

Prior to FDS5, the thermal radiation from the gas space was always absorbed at the surface of the solid material and the emission to the gas space took place on the surface. Starting in FDS5, the solid material can be given an ABSORPTION_COEFFIENT (1/m) that allows the radiation penetrate and absorb into the solid. Correspondingly, the emission of the material is based on the internal temperatures, not just the surface.

8.4.2 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. This section provides a description of the input parameters, starting with a general solid.

Solid Fuels

A solid object might contain multiple layers with multiple material components per layer. The solid object is described by a SURF line which contains the names of the various MATLs it is composed of. Each MATL can undergo several reactions that may occur at different temperatures and consume different amounts of heat. Each individual reaction can produce a (single) solid RESIDUE, water vapor, and/or fuel gas. For example, the evaporation of water from a solid material is described by the "reaction" that converts liquid water to water vapor. This reaction occurs close to 100 °C and produces only water vapor. It does not produce a solid RESIDUE nor fuel gas. However, a pyrolyzing solid might undergo a reaction that produces a solid RESIDUE, water vapor, *and* fuel gas.

For each MATL entry in the input file, decide how many reactions it can undergo. It may not undergo any – it may only 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 quite often the empirical observation of multiple reactions does not imply N_REACTIONS > 1, but is caused by the fact that the examined sample is a mixture of multiple materials reacting at different temperatures. Currently, the maximum number of reactions for each material is 10 and the chain of consecutive reactions may contain up to 20 steps.

Next, decide what each reaction produces: a (single) solid RESIDUE, water vapor, and/or fuel gas. This information is conveyed to FDS via the *yields*: NU_RESIDUE(j), NU_WATER(j), and NU_FUEL(j), respectively. Here, j indicates which reaction the parameters pertain to. If, like the evaporation of water, only water vapor is produced, set NU_WATER(j)=1.0 and the other two to zero. The yields are all zero by default. If NU_RESIDUE(j) is non-zero, then you *must* indicate what the solid residue is via RESIDUE(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.

Now you must specify at what temperature the reaction occurs, and how fast the reaction occurs at that temperature. The reaction rate at the temperature, T_s, of the ith material (*i.e.* the MATL that you are currently describing) undergoing its jth reaction is given by:

$$r_{ij} = \frac{\partial}{\partial t}\left(\frac{\rho_{s,i}}{\rho_{s0}}\right) = \left(\frac{\rho_{s,i}}{\rho_{s0}}\right)^{n_{s,ij}} A_{ij}\, \exp\left(-\frac{E_{ij}}{RT_s}\right)\, \max\left[0, T_s - T_{thr,ij}\right]^{n_{t,ij}} \tag{8.4}$$

$\rho_{s,i}$ is the density of the ith material within that particular layer, in the sense of the mass of the ith material divided by the volume of the layer. ρ_{s0} is the initial density of the layer. Thus, $\rho_{s,i}/\rho_{s0}$ is a quantity that increases as the ith material is produced as a residue of some other reaction, and decreases as the ith material decomposes. If the layer is composed of only one material, and if the reactions produce no solid residues, then $\rho_{s,i}/\rho_{s0}$ is always 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), with units of 1/s. 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. A and E are not readily accessible for most real materials. However, if they are known, specify both. Avoid specifying just one because they act as a pair. If A and E are not known, which is usually the case, specify REFERENCE_RATE (1/s) and REFERENCE_TEMPERATURE (°C). This directs FDS to choose A and E so that the reaction rate (REFERENCE_RATE) is achieved at the REFERENCE_TEMPERATURE. The default value of REFERENCE_RATE is 0.1 s^{-1}. It is suggested that unless you have information to the contrary, leave REFERENCE_RATE at its default value and just specify the REFERENCE_TEMPERATURE. Note that the REFERENCE_TEMPERATURE is *not* the same as an ignition temperature. Rather, it is simply the temperature at which the mass fraction of the ith material decreases at a rate of 0.1 s^{-1}. You should check the sensitivity of these parameters following the procedure explained in Section 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). By default, $T_{thr,ij}$ is -273.15 degrees Celsius, $n_{t,j}$ is zero; thus, the last term of Equation 8.4 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. $n_{t,j}$ is prescribed under the name N_T(j).

Remember that all temperatures are specified in degrees Celsius, but are then converted to degrees Kelvin within the program. Thus, the formulae in this section ought to be interpreted in terms of the absolute temperature.

One last issue before a few examples – the most important one of all. Eq. (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.

Here is an example of a material that burns in the neighborhood of 350 °C, converting all its mass to fuel gases (NU_FUEL=1.).

```
&MATL ID                        = 'My Fuel'
      FYI                       = 'Properties completely fabricated'
      SPECIFIC_HEAT             = 1.0
      CONDUCTIVITY              = 0.1
      DENSITY                   = 100.0
      HEAT_OF_COMBUSTION        = 15000.
      N_REACTIONS               = 1
      NU_FUEL(1)                = 1.
      REFERENCE_TEMPERATURE(1)  = 350.
      HEAT_OF_REACTION(1)       = 3000. /
```

Note that the (1) has 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 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.

Several other examples of solid phase reactions can be found in various Verification examples. See Sections 16.4.6 and 16.4.7.

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 what its subsequent burning rate is, 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/m^2 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. 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 being solved 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.

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 only drawback of this approach is that the fuel gases burn regardless of any ignition source. Thus, if a liquid fuel is specified, the fuel begins burning at once. Here is an example of a steel pan filled with a thin layer of ethanol. Note that the material properties have not all been verified.

```
&MATL ID                   = 'ETHANOL LIQUID'
      EMISSIVITY           = 1.0
      NU_FUEL              = 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. /
```

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_FUEL, 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.

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. An example is given in Section 16.4.5.

8.4.3 Special topic: Making Fuels Disappear (BURN_AWAY)

If a burning object is to disappear from the calculation once it is exhausted of fuel, set BURN_AWAY=.TRUE.. Use this 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 as a SURF parameter, then a solid object with this SURF_ID disappears from a calculation as the mass of each of its mesh cells are consumed. The mass of each mesh cell is the volume of the mesh cell multiplied by the DENSITY of the materials making up the the obstruction.

Note also that 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.

Also note that the amount of combustible fuel equals the DENSITY of the designated materials multiplied by the volume of the mesh cell. 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.

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. The shrinking behavior can be prevented in the numerical model by setting SHRINK=.FALSE. on the SURF line. However, setting SHRINK=.FALSE. may introduce numerical problems in the wall solver. 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.

8.4.4 Special Topic: Initial and Backside Boundary Conditions

By default, the initial temperature of the solid material is set to ambient temperature. Use TMP_INNER on the SURF line to specify a different initial temperature. Also, the backside temperature boundary condition of a solid can be set using the parameter TMP_BACK on the SURF line. TMP_BACK is not the actual backside surface temperature, but the gas temperature that the surface transfer heat with. 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).

8.4.5 Special Topic: Numerical Accuracy and Stability

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. The numerical stability of the solid phase solution may then be improved by making the mesh density more uniform inside the material and by making the mesh cells smaller. Adjustments may also be needed in case of extremely transient heat transfer situations. Use STRETCH_FACTOR=1. on the SURF line to have a perfectly uniform mesh. Values between 1 and 2 give different levels of stretching. The size of all the mesh cells can be scaled by setting CELL_SIZE_FACTOR less than 1.0. For example, CELL_SIZE_FACTOR=0.5 makes the mesh cells half the size. Setting WALL_INCREMENT=1 on the TIME line forces the solid phase temperatures to be solved on every time step.

See Section 8.7 for ways to check and improve the accuracy of the solid phase calculation.

8.5 User-Specified Functions: The RAMP and TABL Namelist Groups

Many of the parameters specified in the 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 group RAMP and TABL, as it names imply, allow you to control the behavior of selected parameters. RAMP allows you to specify a function with one independent variable (such as time) is mapped to one dependent variable (such as velocity). TALB allows for the specification of a mapping from multiple independent variables (such as a solid angle) to multiple dependent variables (such as a sprinkler flow rate and droplet speed).

8.5.1 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 burning history 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 interpolation is used to fill in intermediate time points. Be sure that the prescribed function starts at T=0.0.

Note that each set of RAMP lines must have a unique ID and that the lines must be listed with monotonically increasing T.

Note 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 /
```

Now the temperature and velocity of the incoming air stream would follow the same ramp functions.

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

Use TAU_MF(N) or RAMP_MF(N) to control the ramp-ups for either the mass fraction or mass flux of species N. The mass fraction of species N at the surface is given by

$$Y_N(t) = Y_N(0) + f(t)\,(Y_N - Y_N(0))$$

where $Y_N(0)$ is the ambient mass fraction of species N (MASS_FRACTION_0 in the Nth SPEC namelist line is used to prescribe $Y_N(0)$), Y_N is the desired mass fraction to which the function $f(t)$ is ramping (MASS_FRACTION(N) specified in the SURF line is used to prescribe Y_N). The function $f(t)$ is either a tanh, t^2, or user-defined function. For a user-defined function, indicate the name of the ramp function with RAMP_MF(N), a character string.

8.5.2 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.

Note that each set of RAMP lines must have a unique ID and that the lines must be listed with monotonically increasing T.

8.5.3 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.

8.6 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 here in this document and on the FDS website.

Table 8.1 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" 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 8.1 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. The use of RGB is preferable, especially to create colors that do not clash with the pastel colors used to show temperatures, concentrations, *etc.* See Table 8.1 for a list of color names and RGB values.

8.6.1 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 (SGI Users: use rgb files instead of jpg). 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 8.1: 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

8.7 Verifying the Solid Phase Properties

As this chapter has demonstrated, real materials 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 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 MISC line. This turns off the convective heat flux from gas to surface and vis verse. The heat flux to the solid is specified via EXTERNAL_FLUX (kW/m^2) 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.

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, 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.

Chapter 9

Combustion and Radiation

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, 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.

The gas phase reaction can be described in two ways. By default, a so-called *mixture fraction* model is used to account for the evolution of the fuel gas from its surface of origin through the combustion process. The alternative is what is referred to as the *finite-rate* approach, where all of the individual gas species involved in the combustion process are defined and tracked individually. This is a costlier and more complicated approach than the *mixture fraction* model. This chapter describes both methods, with an emphasis on the more commonly used mixture fraction model.

9.1 Mixture Fraction Combustion: The `REAC` Namelist Group

There are two ways of designating a fire: the first is to specify a Heat Release Rate Per Unit Area `HRRPUA` on a `SURF` line. The other is to specify a `HEAT_OF_REACTION`, along with other thermal parameters on a `MATL` line, in which case the burning rate of the fuel depends on the net heat feedback to the surface. In both cases, the mixture fraction combustion model is used. In fact, the mere presence of these parameters automatically invokes the mixture fraction model. Do not specify explicitly gas species like oxygen if you have also specified heat release rates or solid phase reaction rates.

A single `REAC` line is used with the mixture fraction model. If the `REAC` line is not found in the input file, propane will be used as the surrogate fuel, and all burning rates will be adjusted accordingly. If you only specify the fire's heat release rate with `HRRPUA`, then the reaction parameters may not require adjusting, and no `REAC` line need be added to the input file. However, if you know something about the predominant fuel gas, you might want to consider specifying, at the very least, the basic stoichiometry via the `REAC` line.

Using the mixture fraction model, each reaction is assumed to be of the form:

$$\mathrm{C_xH_yO_zN_v Other_w} + \nu_{\mathrm{O_2}}\ \mathrm{O_2} \rightarrow \nu_{\mathrm{CO_2}}\ \mathrm{CO_2} + \nu_{\mathrm{H_2O}}\ \mathrm{H_2O} + \nu_{\mathrm{CO}}\ \mathrm{CO} + \nu_{\mathrm{Soot}}\ \mathrm{Soot} + \nu_{\mathrm{N_2}}\ \mathrm{N_2} + \nu_{\mathrm{Other}}\ \mathrm{Other} \tag{9.1}$$

You need only specify the chemical formula of the fuel along with the yields of CO, soot, and H_2, and the amount of hydrogen in the soot, H_{frac}. For completeness you can specify the N_2 content of the fuel and the presence of other species. FDS will use that information internally to determine the amount of combustion

products that are formed:

$$\begin{aligned}
\nu_{O_2} &= \nu_{CO_2} + \frac{\nu_{CO}}{2} + \frac{\nu_{H_2O}}{2} - \frac{\mathrm{z}}{2} \\
\nu_{CO_2} &= \mathrm{x} - \nu_{CO} - (1 - H_{frac})\nu_{soot} \\
\nu_{H_2O} &= \frac{\mathrm{y}}{2} - \frac{H_{frac}}{2}\nu_{soot} - \nu_{H_2} \\
\nu_{CO} &= \frac{W_f}{W_{CO}} y_{CO} \\
\nu_{H_2} &= \frac{W_f}{W_{H_2}} y_{H_2} \\
\nu_{soot} &= \frac{W_f}{W_S} y_s \\
\nu_{N_2} &= \frac{\mathrm{v}}{2} \\
\nu_{other} &= w \\
W_s &= H_{frac} W_H + (1 - H_{frac}) W_C
\end{aligned}$$

The following parameters may be prescribed on the REAC line. 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.

ID A character string naming the reaction.

C, H, O, N, OTHER The fuel chemical formula. All numbers are positive. (Mixture Fraction only, default values are those of propane)

MW_OTHER Average molecular weight for OTHER (g/mol). (Mixture Fraction only, default is the molecular weight of N_2, 28 g/mol)

Y_O2_INFTY Ambient mass fraction of oxygen (Mixture Fraction only, default 0.23)

Y_F_INLET Mass fraction of fuel in fuel stream (Mixture Fraction only, default 1.0)

SOOT_YIELD The fraction of fuel mass converted into smoke particulate, y_s. Note that this parameter does not apply to the processes of soot growth and oxidation, but rather to the net production of the smoke particulate from the fire. (Mixture Fraction only, default 0.01)

SOOT_H_FRACTION The fraction of the atoms in the soot that are hydrogen. (Mixture Fraction only, default 0.1)

CO_YIELD The fraction of fuel mass converted into carbon monoxide, y_{CO}. (Mixture Fraction only, default 0.0)

H2_YIELD The fraction of fuel mass converted into hydrogen, y_{H_2}. (Mixture Fraction only, default 0.0)

HEAT_OF_COMBUSTION ΔH (kJ/kg). 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

$$\Delta H \approx \frac{\nu_{O_2} W_{O_2}}{\nu_f W_f} \ \texttt{EPUMO2} \qquad \text{kJ/kg}$$

EPUMO2 The amount of energy released per unit mass of oxygen consumed. (kJ/kg) 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.

IDEAL Logical value indicating whether or not the EPUMO2 or HEAT_OF_COMBUSTION values represent values for complete combustion (.TRUE.) or for incomplete combustion (.FALSE.), i.e. the values account for the specified y_{CO}, y_{H_2}, and y_s. If IDEAL, then FDS will internally adjust ΔH to account for products of incomplete combustion.

A few sample REAC lines are given here. The values are for demonstration only.

```
&REAC ID          = 'METHANE'
      C           = 1.
      H           = 4. /

&REAC ID                  = 'PROPANE'
      SOOT_YIELD          = 0.01
      C                   = 3.
      H                   = 8.
      HEAT_OF_COMBUSTION  = 46460.
      IDEAL               = .TRUE. /

&REAC ID                  = 'PROPANE'
      SOOT_YIELD          = 0.01
      C                   = 3.
      H                   = 8.
      HEAT_OF_COMBUSTION  = 46124.
      IDEAL               = .FALSE. /

&REAC ID                  = 'ACRYLONITRILE'
      C                   = 3.
      H                   = 3.
      N                   = 1.
      HEAT_OF_COMBUSTION  = 24500.
      IDEAL               = .TRUE. /

&REAC ID                  = 'CARBON DISULFIDE'
      C                   = 1.
      Other               = 2.
      MW_OTHER            = 32.
      HEAT_OF_COMBUSTION  = 13600.
      IDEAL               = .TRUE. /
```

9.1.1 Important Issues Related to the Mixture Fraction Models

This section explains the various approximations that affect both the gas phase parameters (REAC line) and the solid (or liquid) phase parameters (SURF line). These approximations are needed either to compensate for less than desirable mesh resolution or limitations of the mixture fraction combustion model.

Heat of Combustion: By default the EPUMO2 value is combined with the stoichiometric parameters listed on the REAC line to compute the heat of combustion. Specifying the HEAT_OF_COMBUSTION will override that computation. However, if heats of reaction have been specified on the MATL line and the heat of combustion of the material differs from that specified by the governing reaction, then add a HEAT_OF_COMBUSTION (kJ/kg) to the MATL line. With the mixture fraction combustion model, it is assumed that there is only one fuel. However, in a realistic fire scenario, there may be many fuels originating from 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.

Gas Phase Fire Suppression: 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 a flame is viable at the fuel-oxygen interface. The Technical Reference Guide [1] contains more details about how the mechanism works. The only parameters you can control are the Limiting Oxygen Index X_O2_LL, and the CRITICAL_FLAME_TEMPERATURE. Both are set on the REAC line. The default values are 0.15 (volume fraction) and 1427 °C, respectively. To eliminate any gas phase suppression, set X_O2_LL to 0. To turn off suppression completely, set SUPPRESSION=.FALSE. on the MISC line.

CO Production: An algorithm has been implemented that computes the combustion as a two step reaction that predicts the formation and destruction of CO. The Technical Reference Guide [1] contains more details about how the mechanism works. This algorithm is used when CO_PRODUCTION is set to .TRUE. on the MISC line. Even though the algorithm predicts CO formation and its eventual oxidation at elevated temperature, it cannot predict the post-flame yield of CO. For example, within a flashed over compartment, the algorithm predicts the elevated CO levels, but it cannot predict the CO concentration of the exhaust gases that exit the flaming region. Thus, even if using this model, you must specify the CO_YIELD that is expected of a well-ventilated fire.

Note that when active, this algorithm requires the use of three parameters for the mixture fraction vs. the two parameters used when it is disabled and will therefore increase run times and memory usage accordingly. If the simulation you are performing will not result in an under-ventilated fire, then there will be of little if any benefit to enabling the CO production algorithm.

9.2 Extra Gas Species: The SPEC Namelist Group

Normally when you specify a fire via either HRRPUA on the SURF line or reaction parameters on the MATL line, the mixture fraction combustion model is applied. A a set of two or three scalar variables, Z_i, represent the state of the combustion process from pure fuel ($\sum Z_i = 1$) to pure air ($\sum Z_i = 0$). The major reactants and products of combustion – fuel, O_2, CO_2, H_2O, N_2, CO and soot – are all pre-tabulated functions of the mixture fraction, Z. In other words, the values of Z_i in any given mesh cell determines the mass fraction of all the gases listed. The fuel chemistry listed under the REAC namelist group is used to generate the table associating the mass fractions with Z_i. You need not, *and should not*, explicitly list the reactants and products of combustion.

Suppose however that gases are introduced into the domain that are neither reactants nor products of combustion. This gas can be tracked separately from the mixture fraction via an additional scalar transport equation[1]. In fact, there does not need to be any fire at all – FDS can be used to transport a mixture of non-reacting ideal gases.

The namelist group SPEC is used to specify each additional species. Each SPEC line should include at the very least the name of the species via a character string called (ID). Next, if the ambient (initial) mass fraction 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 9.1. The physical properties of these gases are known and do not need to be specified. However, if a desired gas is not included in Table 9.1, its molecular weight MW must be specified in units of g/mol. In addition, if a DNS calculation is being performed, either the Lennard-Jones potential parameters σ (SIGMALJ) and ε/k (EPSILONKLJ) should be specified; or the VISCOSITY (kg/m/s), CONDUCTIVITY (W/m/K), and DIFFUSIVITY (m^2/s) between the given species and the background species should be specified.

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

As an example, the lines:

```
&SPEC ID='ARGON', MASS_FRACTION_0=0.1, MW=40. /
&SPEC ID='HELIUM' /
       .
       .
&SURF ID='INLET', MASS_FRACTION(2)=0.2, VEL=-0.3, TAU_MF(2)=0.5, TAU_V=0.5 /
```

specify that ARGON and HELIUM are to be included in the calculation in addition to the (unlisted) default BACKGROUND_SPECIES='AIR'. At the INLET, a mixture of helium (0.2 by mass), argon (0.1 by mass because nothing different is specified), and air (0.7 by mass making up the rest) flows out at a velocity of 0.3 m/s *into* the flow domain. The mass fraction of helium and the velocity are both ramped up according to the function $\tanh(t/0.5)$.

If the simulation does not involve the mixture fraction model – either because no combustion is desired or if a finite rate reaction(s) is being specified (see Section 9.3) – 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 implicitly defined BACKGROUND_SPECIES, use the MISC line. If this species is not listed in Table 9.1, specify its molecular weight, MW, and (optionally) its VISCOSITY and CONDUCTIVITY. In the absence of any of these parameters, the appropriate values of 'AIR' are assumed.

[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 mixture fraction variable and water vapor generated by evaporating sprinkler droplets is tracked via its own transport equation. In the case of sprinklers, do not specify WATER VAPOR as an extra species – it is done automatically.

Table 9.1: **Optional Gas Species [5]**

Species	Mol. Wgt. (g/mol)	σ (Å)	k/ε (K)
AIR	29	3.711	78.6
CARBON DIOXIDE	44	3.941	195.2
CARBON MONOXIDE	28	3.690	91.7
HELIUM	4	2.551	10.22
HYDROGEN	2	2.827	59.7
METHANE	16	3.758	148.6
NITROGEN	28	3.798	71.4
OXYGEN	32	3.467	106.7
PROPANE	44	5.118	237.1
WATER VAPOR	18	2.641	809.1

Recognized species that are emissive will been 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.

9.3 Finite-Rate Combustion

Usually, FDS uses mixture fraction concepts to describe combustion. However, FDS can also explicitly track gas species and reactions that can occur between them. 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. Note: you may use the finite-rate reaction scheme 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` on the `MISC` line is normally set to be `'NITROGEN'`.

3. The namelist group `SPEC` is used to specify each additional species. Do not enter a `SPEC` line for the background species.

4. Read Section 9.2 for a description of the boundary conditions for the gas species.

5. 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 evalute the reactions in the order they are listed in the input file.

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

   `OXIDIZER` Character string indicating which of the listed optional gas species is the oxidizerl.

   `BOF` Pre-exponential factor in one-step chemical reaction in units of cm^3/mole/s.

   `E` Activation energy for one-step chemical reaction in units of kJ/kmol.

   `NU` Array containing the stoichiometry of the chemical reaction for each `SPEC` where negative values indicate reactants and positive values indicate products. Note that the background species cannot participate in the reaction.

   `N_S` Array containing the exponents for the finite rate equation for each `SPEC`. Note that a `SPEC` can be given an `N_S` but not a `NU`, i.e the rate equation can be dependent on a species that does not participate directly in the reaction. Note that the background species cannot participate in the reaction.

   `HEAT_OF_COMBUSTION` The effective heat of combustion the chemical reaction in units of kJ/kg. (Default 40,000 kJ/kg)

9.4 Radiation Transport: The `RADI` Namelist Group

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. The most important issue involves 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 on the mesh. In calculations in which the mesh cells are on the order of a centimeter and 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. To compensate, if you prescribe a non-zero value of `RADIATIVE_FRACTION` on the `RADI` line, a mesh cell cut by the flame radiates that fraction of the chemical energy being released into it. Some of that energy may be reabsorbed elsewhere, yielding a net radiative loss that is less than `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.

There are several ways to improve the performance of the Finite Volume Method in solving the radiation transport equation (RTE), most of which increase the computation time. The solver has two modes of operation – a gray gas model (default) and a wide band model [1]. Modifications to these models can be made via a namelist group called RADI. If running in gray gas mode (default), 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 reduced from every 3 time steps with integer TIME_STEP_INCREMENT. The increment over which the angles are updated can be reduced from 5 with the integer ANGLE_INCREMENT. Briefly, 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.

A few parameters affecting the absorption of radiation by water droplets are as follows: RADTMP is the assumed radiative source temperature. It is used in the computation of the mean scattering and absorption cross sections of water droplets. 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.

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.. See FDS Technical Reference Guide for more details.

Note also that it is possible to turn off the radiation transport solver (saving roughly 20 % in CPU time) by adding the statement RADIATION=.FALSE. to the MISC line. For isothermal calculations, the radiation is turned off automatically. If burning is taking place and radiation is turned off, then the total heat release rate is reduced by the RADIATIVE_FRACTION, which is input on the RADI line. This radiated energy completely disappears from the calculation. More on this feature can be found in Section 9.1.1.

Chapter 10

Particles and Droplets: The PART Namelist Group

Lagrangian particles[1] are used in FDS as 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.

All Lagrangian particles must be explicitly defined via the PART namelist group. In versions of FDS prior to 5, water droplets and smoke particles were implicitly defined. Shortcuts for defining water droplets and smoke particles are possible, via parameters like WATER=.TRUE. and MASSLESS=.TRUE.

10.1 Basics

Properties of different types of Lagrangian particles are designated via the PART namelist group. Much like SURF lines contain the properties of a solid surface or vent, PART lines contain information about particles and droplets. 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 type 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 Spk-123', QUANTITY='SPRINKLER LINK TEMPERATURE', PART_ID='my water', ... /
```

[1] Throughout this section, the terms "droplets" and "particles" are used interchangeably. From the point of view of FDS, they are all Lagrangian particles; that is, point elements that are not bound by the structure of the underlying grid.

Note that a 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.

10.2 Controlling Particles and Droplets

Depending on how the particles or droplets are introduced into the computational domain, the following are important parameters for controlling them:

DT_INSERT Time increment in seconds between the introduction of a "batch" of particles or droplets. The number per "batch" depends on how they are introduced. If more particles are desired, lower the input value of this parameter. The default value is 0.05 s.

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 10.4.

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.

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 NPPC is 1. As an example, the following set of input lines:

```
&PART ID='drops', QUANTITIES(1:3)='DROPLET_DIAMETER','DROPLET_TEMPERATURE','DROPLET_AGE',
      DIAMETER=750., SAMPLING_FACTOR=1, COLOR='RED', EVAPORATE=.FALSE. /
&SURF ID='HOLE', PART_ID='drops', VEL=-5., PARTICLE_MASS_FLUX=0.1, COLOR='RED' /
&OBST XB=-0.2,0.2,-0.2,0.2,4.0,4.4, SURF_IDS='INERT','HOLE','INERT' /
```

creates an obstruction that ejects non-evaporating, red particles with a mean volumetric diameter of 750 μm out of its sides at a rate of 0.1 kg/m^2/s. FDS will adjust the mass flux if the obstruction or vent dimensions are changed to conform to the numerical grid. Note that the IDs have no meaning other than as identifiers. The particles are colored red in Smokeview, but can also be colored according to their diameter, temperature, or age.

Droplets Introduced at a Sprinkler or Nozzle

DROPLETS_PER_SECOND is the number of droplets inserted every second per active sprinkler or nozzle. Its default value is 1000. 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. Also note that the number of droplets introduced per "batch" is DROPLETS_PER_SECOND times DT_INSERT.

Particles or Droplets Introduced Initially

Sometimes it is convenient to introduce droplets or particles at the start of the simulation. For this purpose, NUMBER_INITIAL_DROPLETS is the number of particles/droplets within the computational domain at the start of the simulation. Its default value is 0, meaning that initially there are no particles or droplets present. If non-zero, also specify MASS_PER_VOLUME (kg/m^3) which specifies the particle/droplet mass per unit volume (Default 1 kg/m^3). Do not confuse this parameter with DENSITY, explained in the next section. For example, water has a DENSITY of 1000 kg/m^3, whereas a liter of water broken up into droplets and spread over a cubic meter has a MASS_PER_VOLUME of 1 kg/m^3. Also, to limit the particles/droplets 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.

Droplets that Strike 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.

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.

10.3 Particle and Droplet Properties

For Lagrangian particles that are not MASSLESS, the following parameters should be included on the PART line.

DENSITY The density of the liquid or solid droplet/particle. (Default 1000 kg/m^3)

SPECIFIC_HEAT Specific heat of liquid or solid droplet/particle. (Default 4.184 kJ/kg/K)

DIAMETER Median volumetric diameter of droplets/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

$$F(d) = \begin{cases} \frac{1}{\sqrt{2\pi}} \int_0^d \frac{1}{\sigma d'} e^{-\frac{[\ln(d'/d_m)]^2}{2\sigma^2}} \, dd' & (d \le d_m) \\ 1 - e^{-0.693\left(\frac{d}{d_m}\right)^\gamma} & (d_m < d) \end{cases} \tag{10.1}$$

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 $d = d_m$. 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, eliminating numerical instabilities that can occur when droplets get very, very small. The default MINIMUM_DIAMETER is 20 μm. 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.

The following parameters pertain to the evaporation of liquid droplets. It is assumed by default that non-massless particles are liquid droplets, but you can specify EVAPORATE=.FALSE. to change this. Details are in the next section.

VAPORIZATION_TEMPERATURE Boiling temperature of liquid droplet. (Default 100 °C)

MELTING_TEMPERATURE Melting (solidification) temperature of liquid droplet. (Default 0 °C)

INITIAL_TEMPERATURE Initial temperature of liquid droplet. (Default TMPA)

HEAT_OF_VAPORIZATION Latent heat of vaporization of liquid droplet. (Default 2259 kJ/kg)

10.4 Special Types of Particles and Droplets

There are several useful attributes that you can assign to particles or droplets, usually via a simple logical parameter. Be aware with each of these parameters that specifying it as .TRUE. may cause other parameters to be functionally useless, or may cause conflicts that FDS may or may not detect. A good rule of thumb is always to ask yourself what is the basic information that *must* be conveyed to the program, and stick to that. For example, if the particles are to be MASSLESS, there is no point in declaring anything else except maybe a COLOR for Smokeview.

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. Unlike early versions of FDS, particles are no longer colored by gas phase quantities, but rather by properties of the particle itself. For example, 'DROPLET_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 silly since MASSLESS particles are for visualization only.

Static Particles or Droplets

STATIC is a logical parameter indicating whether particles move or just serve as obstructions or clutter. Setting STATIC=.TRUE. only makes sense in conjunction with a non-zero value of NUMBER_INITIAL_DROPLETS. The default value of STATIC is .FALSE.

Water Droplets

WATER=.TRUE. declares that the liquid droplets evaporate into WATER VAPOR, a separate gas phase species that is *automatically* added to the calculation by this command. By default, WATER=.FALSE., even though the default properties of droplets are that of water. Setting WATER=.TRUE. instructs FDS to add WATER VAPOR as an explicitly defined species, and it also invokes appropriate constants related to the absorption of thermal radiation by the water droplets. It also causes the droplets to be colored blue in Smokeview.

If the liquid droplets are to evaporate into some other gaseous species, you must explicitly define the species via the SPEC namelist group (see Section 9.2), and then designate the appropriate SPEC_ID on the PART line.

Fuel Droplets

FUEL=.TRUE. indicates that the liquid droplets evaporate into fuel gas and burn. In this case, add the HEAT_OF_COMBUSTION (kJ/kg) of the fuel. Fuel droplets are colored yellow by default in Smokeview. This feature only works for a mixture fraction-based combustion calculation, in which case 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.

Particles that do not Evaporate

Unless you declare MASSLESS=.TRUE. on the PART line, it is assumed that the particle or droplet has mass and thermal properties that dictate its heat up and evaporation. To prevent evaporation, set EVAPORATE=.FALSE. The particles will still heat up due to convection, but they will not shrink and no additional gaseous species need to be declared.

Note that the absorption of thermal radiation by water (WATER=.TRUE.) or fuel droplets (FUEL=.TRUE.) is handled in FDS with fairly well-established physical sub-models, the details of which are contained in the FDS Technical Reference Guide [1]. However, for arbitrary particles or droplets, there is no assumed radiative absorption.

10.5 Coloring Particles and Droplets

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 'DROPLET_TEMPERATURE' (°C), 'DROPLET_DIAMETER' (μm), 'DROPLET_VELOCITY' (m/s), 'DROPLET_MASS' (kg), 'DROPLET_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.

10.6 Special Topic: Droplet Fuel Sprays

The evaporation of water droplets from sprinklers has been generalized so that a liquid fuel spray nozzle can be modeled. Fuel evaporation is triggered by the inclusion of the phrase FUEL=.TRUE. on the appropriate PART line. The spray nozzle characteristics are specified in the same way as those for a sprinkler. Here is an example of a liquid fuel spray nozzle (also see Section 16.6.2):

```
&DEVC ID='nozzle_1', XYZ=4.0,-.3,0.5, PROP_ID='nozzle', QUANTITY='TIME', SETPOINT=0. /

&PART ID='heptane droplets', FUEL=.TRUE., VAPORIZATION_TEMPERATURE=98.,
      HEAT_OF_VAPORIZATION=316., SPECIFIC_HEAT=2.25, DENSITY=688.,
      QUANTITIES(1:2)='DROPLET_DIAMETER','DROPLET_TEMPERATURE',
      DROPLETS_PER_SECOND=2000, DIAMETER=1000., HEAT_OF_COMBUSTION=44500.,
      DT_INSERT=0.02, SAMPLING_FACTOR=1 /

&PROP ID='nozzle', PART_ID='heptane droplets', FLOW_RATE=1.96,
      FLOW_RAMP='fuel', DROPLET_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 /
```

The vaporization (boiling) temperature of the liquid fuel is in degrees Celsius, the heat of vaporization is in units of kJ/kg, the specific heat is in units of kJ/kg/K, and the density is in units of kg/m^3. `FUEL=.TRUE.` automatically invokes a mixture fraction calculation in which fuel from the evaporating fuel droplets is burned according to the overall reaction scheme.

Note that this construct is fragile and subject to mesh dependence. If the mesh cells are too coarse, the evaporating fuel is diluted to such a degree that it never burns. Proper resolution depends on the type of fuel and the amount of fuel being ejected from the nozzle.

Simulations with both fuel and water droplets are possible, unlike versions of FDS prior to 5.

10.7 Special Topic: Suppression by Water (Mixture Fraction Model Only)

Modeling suppression of a fire by a water spray is challenging because the relevant physical mechanisms occur at length scales smaller than a single mesh cell. In the gas phase, flames are extinguished due to lowered temperatures and dilution of the oxygen supply. See Section 9.1.1 for more information about gas phase suppression.

For the solid phase, 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. 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

$$\dot{m}''_f(t) = \dot{m}''_{f,0}(t)\, e^{-\int k(t)\, dt} \tag{10.2}$$

Here $\dot{m}''_{f,0}(t)$ is the user-specified burning rate per unit area when no water is applied and k is a function of the local water mass per unit area, m''_w, expressed in units of kg/m^2.

$$k(t) = \texttt{E_COEFFICIENT}\, m''_w(t) \quad \text{s}^{-1} \tag{10.3}$$

The parameter `E_COEFFICIENT` must be obtained experimentally, and it is expressed in units of m^2/kg/s. Usually, this type of suppression algorithm is invoked when the fuel is complicated, like a cartoned commodity.

Chapter 11

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.

Past versions of FDS 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. Devices are now specified with a new format that streamlines and expands the possibilities of sensor profiles.

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).

11.1 Device Location and Orientation: The DEVC Namelist Group (Table 13.4)

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 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.

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 prescribe,

```
&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.

11.2 Device Output

Each device has a QUANTITY associated with it. The output file for all DEVC quantities is 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. If the number of DEVC lines exceeds 256, the limit of some spreadsheet applications, the output file will be split into appropriately sized smaller files. To prevent the file splitting, specify COLUMN_DUMP_LIMIT=.FALSE. on the DUMP line.

All devices must have a specified QUANTITY. Some special devices (Section /refinfo:PROP) have their QUANTITY specifed on a PROP line.

A QUANTITY specified on a PROP line associated with a DEVC line will override an QUANTITY specified on the DEVC line.

11.3 Special Devices and their Properties: The PROP Namelist Group (Table 13.16)

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.

11.3.1 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,
      DROPLET_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 165.)

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 in units of L/min. An alternative is to provide the K_FACTOR in units of L/min/bar$^{\frac{1}{2}}$ and the OPERATING_PRESSURE in units of atm. 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 gpm/psi$^{\frac{1}{2}}$ is equivalent to 14.41 L/min/bar$^{\frac{1}{2}}$.

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)

DROPLET_VELOCITY Initial droplet velocity. (Default 5 m/s)

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. The droplets are uniformly distributed within this belt.

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 Section 10 for additional details.

Be aware that sprinklers produce many droplets that need to be tracked in the calculation. To limit the burden, 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, is included in 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, VELOCITY, and FLOW_RATE, then a SPRAY_PATTERN_TABLE can be specified using the TABL (Section 8.5) 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_PARTTERN_TABLE and then one or more TABL lines of the format:

[1] Past versions of FDS used a separate file to store properties of a given sprinkler. This file is no longer used.

```
&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 example below, the spray pattern is defined as two jets, each with a velocity of 10 m/s and a flow rate of 20 L/min (the total FLOW_RATE is 40 L/min and the fraction for each jet is 0.5). The jets are centered at points 45° below the "equator," and are separated by 180°.

```
&PROP ID='y-pipe',
      QUANTITY='SPRINKLER LINK TEMPERATURE',
      FLOW_RATE=40.,
      PART_ID='water_drops',
      SPRAY_PATTERN_TABLE='TABLE1' /

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

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. Also note that the TABL lines can be specified in any order.

11.3.2 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, for example, specify a QUANTITY and SETPOINT directly on the DEVC line. An example of a nozzle can be found in the verification case **water_fuel_sprays.fds**

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., DROPLET_VELOCITY=5., SPRAY_ANGLE=0.,45.    /
```

designate two nozzles of the same type, one which activates at zero seconds, the other at 5 s. Note that nozzles must have an associated PROP line. The parameter PART_ID tells FDS to spray liquid droplets.

11.3.3 Heat Detectors

QUANTITY='LINK TEMPERATURE', as in this example, 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).

11.3.4 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='spot 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', QUANTITY='spot obscuration', ALPHA_C=1.8, BETA_C=-1.1,
      ALPHA_E=0.98, BETA_E=-0.77, ACTIVATION_OBSCURATION=3.28 /
```

where the two characteristic filling or "lag" times are of the form:

$$\delta t_e = \alpha_e u^{\beta_e} \quad ; \quad \delta t_c = \alpha_c u^{\beta_c} \tag{11.1}$$

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 11.1. 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 11.1: Suggested Values for Smoke Detector Model. See Ref. [6] 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	—

11.3.5 Beam Detection Systems

A beam detector can be defined by specifying the endpoints (x1,y1,z1),(x2,y2,z2), of the beam, using XB, and the total total % obscuration at which the detector activates. The two endpoints must lie in the same mesh. FDS determines which mesh cells lie along the path specified by the two endpoints. The beam detector response is evaluated as

$$\text{Obscuration} = \left(1 - \exp\left(-K_m \sum_{i=1}^{N} \rho_{soot,i}\, \Delta x_i\right)\right) \times 100 \ \% \tag{11.2}$$

where i is a mesh cell along the path of the beam, $\rho_{soot,i}$ is the soot density of the mesh cell, and Δx_i is the distance within the mesh cell that is traversed by the beam.

```
&DEVC XB=x1,x2,y1,y2,z1,z2, QUANTITY='path obscuration', ID='beam1', SETPOINT=0.33 /
```

11.3.6 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='soot density', ID='soot1', DEVC_ID='asp1', FLOWRATE=0.1, DELAY=20 /
&DEVC XYZ=..., QUANTITY='soot density', ID='soot2', DEVC_ID='asp1', FLOWRATE=0.2, DELAY=10 /
 ...
&DEVC XYZ=..., QUANTITY='soot density', 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

$$\text{Obscuration} = \left(1 - \exp\left(-K_m \frac{\sum_{i=1}^{N} \rho_{soot,i}(t - t_{d,i})\, \dot{m}_i}{\sum_{i=1}^{N} \dot{m}_i}\right)\right) \times 100 \ \ \%/\text{m} \tag{11.3}$$

where $\dot{m}_i$ is the mass FLOWRATE of the ith sampling location, $\rho_{soot,i}(t - t_{d,i})$ is the soot density at the ith 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.

11.4 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 11.5 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 seconds. 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.

11.4.1 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.

Note that while single DEVC can be used to control multiple items, a DEVC that is being used for a HOLE should not be used for anything else other than additional HOLEs.

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:

```
&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 remove an obstruction, then re-create a different one in its place, use 2 lines

```
&OBST XB=..., SURF_ID='whatever', DEVC_ID='device 1' /
&OBST XB=..., SURF_ID='whatever', DEVC_ID='device 2' /
```

since the code simply sees this as two different obstructions.

Until further notice, an obstruction that makes up the boundary of a "pressure zone" (see Section 8.3) should not be created or removed. The reason for this restriction is that abrupt changes in pressure can cause numerical instabilities.

11.4.2 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 seconds, 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 second ramp-up will begin at 30 seconds instead of at 0 seconds.

If in this example INITIAL_STATE=.TRUE., then the fan should "deactivate," or turn off at 30 seconds. 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. The reason for this restriction is that abrupt changes in pressure can cause numerical instabilities.

11.5 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 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.

Table 11.2: Control function types for CTRL

Function Type	Description
ANY	Changes state if any INPUTs are .TRUE.
ALL	Changes state if all INPUTs are .TRUE.
ONLY	Changes state if and only if N INPUTs are .TRUE.
AT_LEAST	Changes state if at least 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.

A control is identified by the ID parameter. The inputs to the control are identified by 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.

11.5.1 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.

11.5.2 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.

11.5.3 Control Function: DEADBAND

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..

11.5.4 Control Function: RESTART and KILL

There are times when one only wishes 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 one observed the goal being met. By using the KILL control function this can be done automatically.

Additionally there are analyses where one wishes to create some baseline condition and the run multiple permutations of that baseline. For example one may wish to run a series of simulations where different mitigation strategies are tried once a detector alarms. By using the RESTART control function, one 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 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 lead to the job being stopped gracefully, with restart files output, when the temperature at a given point rises above 1000 °C; or to just restart files being output when the velocity at a given point exceeds 10 m/s.

11.5.5 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 numberical 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 vaule 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 seconds. It would then stay in a true state until the timer reaches 120 seconds 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.

11.5.6 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' /
```

11.5.7 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'/
```

Chapter 12

Output Data

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 `DEVC`, 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.

12.1 Output Control Parameters: The `DUMP` Namelist Group

The namelist group `DUMP` contains parameters (Table 13.5) 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_DEVC`, `DT_SLCF`, `DT_PART`, `DT_ISOF`, `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_DROPLETS 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 By default, every 10 time steps FDS purges the output file buffers and forces the data to be written out into the respective output files. To stop this from happening, set this parameter to .FALSE.

12.2 Output Options

12.2.1 Point Measurement Devices

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 12.1. 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.

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. 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.

12.2.2 Integrated (non-pointwise) Measurement Devices

In addition to point measurements, the DEVC group can be used to report integrated quantities (See Table 12.1). 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.

12.2.3 Output Statistics

A useful feature of a device (DEVC) is to specify an output quantity along with a desired statistic. For example,

```
&DEVC XYZ=2.3,4.5,6.7, QUANTITY='TEMPERATURE', ID='whatever', STATISTICS='MAX' /
```

causes FDS to write out the maximum gas phase temperature over the entire mesh containing the point XYZ. Note that it does not compute the maximum over the entire computational domain, just that particular mesh. Other STATISTICS include 'MIN' and 'MEAN'. They can be used for both gas and solid phase output quantities. In the case of solids, the specification of a SURF_ID limits the search to only those surfaces. Use the STATISTICS feature with caution because it demands that FDS evaluate the given QUANTITY in all gas or solid phase cells.

12.2.4 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 13.15) 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 BNDF, 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.

12.2.5 Animated Planar Slices: The SLCF Namelist Group

The SLCF ("slice file") namelist group parameters (Table 13.20) 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.

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.

Slice file information is recorded in files (See Section 19.7) 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 12.4 for more details.

12.2.6 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 12.1. 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).

12.2.7 Animated Isosurfaces: The ISOF Namelist Group

The ISOF ("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:

```
&ISOF QUANTITY='TEMPERATURE', VALUE(1)=50., VALUE(2)=200., VALUE(3)=500. /
```

where the values are in degrees C. Note that the isosurface output files **CHID_*n*.iso** can become very large, so experiment with different sampling rates.

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, soot density or oxygen demand substantial amounts of time to compute at each mesh cell.

12.2.8 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.

Note that there can only be one `DUMP` line.

Data stored in Plot3D [7] files (See Section 19.2) 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.

12.3 Special Output Quantities

12.3.1 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 file consists of six columns. The first column contains the time in seconds. The second through fifth columns contain integrated energy gains and losses, all in units of kW. The second column contains the total heat release rate, the third contains the radiative heat loss to all the boundaries (solid and open), the fourth contains the convective and radiative heat loss to the boundaries (*i.e.* the energy flowing out of or into the domain), and the fifth contains the energy conducted into the solid surfaces. The sixth column contains the total burning rate of fuel, in units of kg/s. It is included merely as a check of the total heat release rate.

Let Ω denote the unblocked computational domain, *i.e.* the volume within the bounding rectangle occupied by gas. Let $\partial\Omega$ by the boundary of Ω. The boundary can be divided into two parts $\partial\Omega = \partial\Omega_1 + \partial\Omega_2$. The first part $\partial\Omega_1$ consists of all the solid walls. The second part $\partial\Omega_2$ consists of openings from outside the domain through which gases may flow. This could be an open window to the exterior, or a forced vent.

The total heat release rate is given by

$$\dot{Q} = \int_\Omega \dot{q}''' \, dV \tag{12.1}$$

The radiative loss to the boundaries can be computed with either a volume or boundary integral

$$\dot{Q}_r = \int_\Omega \nabla \cdot \mathbf{q}_r \, dV = \int_{\partial\Omega} \mathbf{q}_r \cdot d\mathbf{S} = \int_{\partial\Omega} \dot{q}''_r \, dA \tag{12.2}$$

It represents the energy radiating away from the fire and hot gases into the solid boundaries or out of the computational domain. The convective/radiative loss to open boundaries is

$$\dot{Q}_{conv} = \int_{\partial\Omega} c_p \, \rho \, (T - T_\infty) \, \mathbf{u} \cdot d\mathbf{S} + \int_{\partial\Omega_2} \dot{q}''_r \, dA \tag{12.3}$$

where the integral is positive if the flow and radiative flux are going out of the domain. The conductive loss to solid surfaces is given by

$$\dot{Q}_{cond} = \int_{\partial\Omega_1} \dot{q}''_r + \dot{q}''_c \, dA \tag{12.4}$$

where the integral is positive if heat is being lost into a wall colder than the gas.

For scenarios in which the fire is the primary source of energy, after the gas temperatures within the computational domain reach a nearly steady state

$$\dot{Q} \approx \dot{Q}_{conv} + \dot{Q}_{cond} \tag{12.5}$$

This is merely a check of the global energy balance, that is, the energy generated within the space heats up the gases and solid surfaces, and then a balance between heat input and output is achieved.

12.3.2 Visibility and Obscuration

If you are performing a fire calculation using the mixture fraction 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 [8]. The intensity of monochromatic light passing a distance L through smoke is attenuated according to

$$I/I_0 = e^{-KL} \tag{12.6}$$

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

$$K = K_m \, \rho \, Y_s \tag{12.7}$$

Devices that output a % obscuration such as a `DEVC` with a `QUANTITY` of `aspiration`, `spot obscuration` (smoke detector), or `path obscuration` (beam detector) are discussed respectively in Section 11.3.6, Section 11.3.4, and Section 11.3.5

Estimates of visibility through smoke can be made by using the equation

$$S = C/K \tag{12.8}$$

where C is a nondimensional 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 [8]. 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. The second parameter is called the `MASS_EXTINCTION_COEFFICIENT`, and it is the K_m in Eq. (12.7). The default value is 8700 m^2/kg, a value suggested for flaming combustion of wood and plastics. The third parameter is called the `VISIBILITY_FACTOR`, the constant C in Eq. (12.8). It is 3 by default.

The gas phase output quantity `extinction coefficient` is K. The visibility S is output via the keyword `visibility`. Note that each is tied to the mixture fraction formulation of combustion.

12.3.3 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 [9] 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}:

$$z_{int} = \frac{T_l (I_1 I_2 - H^2)}{I_1 + I_2 T_l^2 - 2 T_l H} \tag{12.9}$$

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

$$(H - z_{int})\, T_u = \int_{z_{int}}^H T(z)\, dz \tag{12.10}$$

Further discussion of similar procedures can be found in Ref. [10].

The quantities LAYER HEIGHT, UPPER TEMPERATURE and LOWER TEMPERATURE can be designated via "device" (DEVC) lines in the input file[1]. 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.

12.3.4 The True Gas Temperature vs. the Measured Gas Temperature

The output quantity THERMOCOUPLE is the temperature of the thermocouple itself, usually close to the gas temperature, but not always. It is determined by solving the following equation for T_{TC} iteratively [11]

$$\varepsilon_{TC}(\sigma T_{TC}^4 - U/4) + h(T_{TC} - T_g) = 0 \tag{12.11}$$

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, $h = k_a \mathrm{Nu}/\mathrm{Pr}/d_{TC}$. The bead BEAD_DIAMETER and BEAD_EMISSIVITY are given on the associated PROP line. 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.

12.3.5 Heat Fluxes

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 HEAT_FLUX. The individual components, the *net* convective and radiative fluxes, are CONVECTIVE_FLUX and RADIATIVE_FLUX, respectively. If you want to compare predicted heat flux with a measurement, you often need to use GAUGE_HEAT_FLUX. The difference between 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):

$$\dot{q}_r''/\varepsilon + \dot{q}_c'' + h(T_w - T_\infty) + \sigma(T_w^4 - T_\infty^4)$$

If the heat flux gauge used in an experiment has a temperature other than ambient, set GAUGE_TEMPERATURE on the PROP line associated with the device. When comparing against a radiometer measurement, use RADIOMETER:

$$\dot{q}_r''/\varepsilon + \sigma(T_w^4 - T_\infty^4)$$

For diagnostic purposes it is sometimes convenient to output the INCIDENT_HEAT_FLUX:

$$\dot{q}_r''/\varepsilon + \sigma T_w^4 + \dot{q}_c''$$

There is a gas phase output quantity called RADIANT_INTENSITY. This is used mainly for diagnosing problems with the radiation solver. Even though its units are kW/m^2, it should not be interpreted as the heat flux to an object that would occupy that particular point in space. Rather, it is the integral over all directions of the radiation intensity, $I(\mathbf{x},\mathbf{s})$, a function of both space and direction. It is denoted by U, as in Eq. (12.11).

[1] Note that in FDS 5 and beyond, these quantities are no longer available as slice files.

12.3.6 Droplet Output Quantities

It is possible to record various properties of evaporating droplets. Some of the output quantities are associated with solid boundaries. For example, `[PART_ID]_MPUA` is the Mass Per Unit Area of the droplets named `PART_ID`. Likewise, `[PART_ID]_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. The accumulated total 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. The cooling of a solid surface by droplets of a given type is given by `[PART_ID]_CPUA`, the Cooling Per Unit Area in units of kW/m^2.

Be aware of the fact that the default behavior for droplets hitting the "floor," that is, the plane $z =$ `ZMIN`, is to disappear (`POROUS_FLOOR=.TRUE.` on the `MISC` line). In this case, `[PART_ID]_MPUA` will be zero, but `[PART_ID]_AMPUA` will not. FDS stores the droplet mass just before removing the droplet from the simulation for the purpose of saving CPU time.

Away from solid surfaces, `[PART_ID]_MPUV` is the Mass Per Unit Volume of the droplets as they fly through the air, in units of kg/m^3. `[PART_ID]_FLUX_X`, `[PART_ID]_FLUX_Y`, and `[PART_ID]_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 kg/m^2/s.

12.3.7 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 [12], the following equation can be solved via a simple iterative technique to determine an effective gas temperature, $T_{\rm AST}$:

$$\dot{q}''_r + \dot{q}''_c = \varepsilon\sigma\left(T_{\rm AST}^4 - T_w^4\right) + h(T_{\rm AST} - T_w) \tag{12.12}$$

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, $T_{\rm AST}$, 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.

12.3.8 Integrated Mass and Energy Fluxes through Openings

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.

12.3.9 Wind and the Pressure Coefficient

In the field of wind engineering, a commonly used quantity is known as the `PRESSURE_COEFFICIENT`:

$$C_p = \frac{p-p_\infty}{\frac{1}{2}\rho_\infty U^2} \tag{12.13}$$

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.

12.4 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**.

12.5 Summary of Output Quantities

Table 12.1, spread over the following pages, summarizes the various Output Quantities. Note that lower case quantities are appropriate only for calculations involving the mixture fraction, Z. If individual species are listed via `SPEC` namelist lines, the quantity for mass and volume fractions are `[SPEC_ID]` and `[SPEC_ID]_VF`, respectively. For example, the quantities `water vapor` and `WATER VAPOR` denote the volume fraction of water vapor generated by combustion and the *mass* fraction of water vapor from evaporated sprinkler droplets, respectively.

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.

Table 12.1: Summary of all Output Quantities

Output `QUANTITY`	Symbol	Units	File Type
`ABSORPTION_COEFFICIENT`	κ	1/m	D,I,P,S
`ADIABATIC_SURFACE_TEMPERATURE`	See Section 12.3.7	°C	B,D
`aspiration`	See Section 11.3.6	%	D
`BURNING_RATE`	$\dot{m}''_f$	kg/m^2/s	B,D
`carbon dioxide`	$X_{CO_2}(Z)$	mol/mol	D,I,P,S
`carbon monoxide`	$X_{CO}(Z)$	mol/mol	D,I,P,S
`CONTROL`	See Section 11.5		D
`CONVECTIVE_FLUX`	$\dot{q}''_c$ (Section 12.3.5)	kW/m^2	B,D
`DENSITY`	ρ	kg/m^3	D,I,P,S
`DIVERGENCE`	$\nabla \cdot \mathbf{u}$	s^{-1}	D,I,P,S
`DROPLET_DIAMETER`	$2r_d$	μm	PA
`DROPLET_VELOCITY`	$\lvert\mathbf{u}_d\rvert$	m/s	PA
`DROPLET_TEMPERATURE`	T_d	°C	PA
`DROPLET_MASS`	m_d	kg	PA
`DROPLET_AGE`	t_d	s	PA
`extinction coefficient`	K (Section 12.3.2)	1/m	D,I,P,S
`fuel`	$X_F(Z)$	mol/mol	D,I,P,S
`GAUGE_HEAT_FLUX`	See Section 12.3.5	kW/m^2	B,D
`H`	$H = \lvert\mathbf{u}\rvert^2/2 + \tilde{p}/\rho_0$	(m/s)^2	D,I,P,S
`HEAT FLOW`	See Section 12.3.8	kW	D
`HEAT_FLUX`	See Section 12.3.5	kW/m^2	B,D
`HRR`	$\int \dot{q}'''\, dV$	kW	D
`HRRPUV`	$\dot{q}'''$	kW/m^3	D,I,P,S
`INCIDENT_HEAT_FLUX`	See Section 12.3.5	kW/m^2	B,D
`INSIDE_WALL_TEMPERATURE`	See Section 12.2.4	°C	D
`LAYER HEIGHT`	See Section 12.3.3	m	D
`LINK TEMPERATURE`	See Section 11.3.3	°C	D
`LOWER TEMPERATURE`	See Section 12.3.3	°C	D

Output QUANTITY	Symbol	Units	File Type
MASS FLOW	See Section 12.3.8	kg/s	D
MIXTURE_FRACTION	Z	kg/kg	D,I,P,S
nitrogen	$X_{N_2}(Z)$	mol/mol	D,I,P,S
oxygen	$X_{O_2}(Z)$	mol/mol	D,I,P,S
oxygen mass fraction	$Y_{O_2}(Z)$	kg/kg	D,I,P,S
[PART_ID]_AMPUA	See Section 12.3.6	kg/m^2	B,D
[PART_ID]_CPUA	See Section 12.3.6	kW/m^2	B,D
[PART_ID]_FLUX_X	See Section 12.3.6	kg/m^2/s	P,S
[PART_ID]_FLUX_Y	See Section 12.3.6	kg/m^2/s	P,S
[PART_ID]_FLUX_Z	See Section 12.3.6	kg/m^2/s	P,S
[PART_ID]_MPUA	See Section 12.3.6	kg/m^2	B,D
[PART_ID]_MPUV	See Section 12.3.6	kg/m^3	D,I,P,S
path obscuration	See Section 11.3.5	%	D
PRESSURE	$\tilde{p}$	Pa	D,I,P,S
PRESSURE_COEFFICIENT	C_p (Section 12.3.9)		B,D
RADIANT_INTENSITY	$\int I(\mathbf{x},\mathbf{s})\,d\Omega$	kW/m^2	D,I,P,S
RADIATIVE_FLUX	See Section 12.3.5	kW/m^2	B,D
RADIOMETER	See Section 12.3.5	kW/m^2	B,D
soot volume fraction	$\rho Y_s(Z)/\rho_s$	mol/mol	D,I,P,S
soot density	$\rho Y_s(Z)$	mg/m^3	D,I,P,S
[SPEC_ID]	Y	kg/kg	D,I,P,S
[SPEC_ID]_VF	X	mol/mol	D,I,P,S
[SPEC_ID]_FLUX_X	$\rho u Y_\alpha$	kg/m^2/s	D,I,P,S
[SPEC_ID]_FLUX_Y	$\rho v Y_\alpha$	kg/m^2/s	D,I,P,S
[SPEC_ID]_FLUX_Z	$\rho w Y_\alpha$	kg/m^2/s	D,I,P,S
spot obscuration	See Section 11.3.4	%/m	D
SPRINKLER LINK TEMPERATURE	See Section 11.3.1	°C	D
TEMPERATURE	T (Section 12.3.4)	°C	D,I,P,S
THERMOCOUPLE	T_{TC} (Section 12.3.4)	°C	D,I,P,S
TIME	t (Section 11.1)	s	D
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	See Section 12.3.3	°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 12.3.2)	m	D,I,P,S
VOLUME FLOW	See Section 12.3.8	m^3/s	D
WALL_TEMPERATURE	T_w	°C	B,D
water vapor	$X_{H_2O}(Z)$	mol/mol	D,I,P,S
WATER VAPOR	X_{H_2O}	mol/mol	D,I,P,S

Chapter 13

Alphabetical List of Input Parameters

This Appendix lists all of the input parameters for FDS in seperate 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.

13.1 BNDF (Boundary File Parameters)

Table 13.1: For more information see Section 12.2.6.

BNDF (Boundary File Parameters)				
FYI	Character	Comment String (has no effect)		
PROP_ID	Character	Source of specific property info		
QUANTITY	Character	Quantity to visualize		

13.2 CLIP (MIN/MAX Clipping Parameters)

Table 13.2: For more information see Section 6.6.

CLIP (Specified Upper and Lower Limits)				
FYI	Character	Comment String (has no effect)		
MAXIMUM_DENSITY	Real	Maximum Gas Density	kg/m^3	
MAXIMUM_MASS_FRACTION	Real Array	Maximum Gas Mass Fraction	kg/kg	
MAXIMUM_TEMPERATURE	Real	Maximum Gas Temperature	°C	
MINIMUM_DENSITY	Real	Minimum Gas Density	kg/m^3	
MINIMUM_MASS_FRACTION	Real Array	Minimum Gas Mass Fraction	kg/kg	
MINIMUM_TEMPERATURE	Real	Maximum Gas Temperature	°C	

13.3 CTRL (Control Function Parameters)

Table 13.3: For more information see Section 11.5.

CTRL (Control Function Parameters)				
DELAY	Real	Time delay	s	0.
FUNCTION_TYPE	Character	Type of control function		
ID	Character	IDentifier		
INITIAL_STATE	Logical	Initial state of control function		.FALSE.
INPUT_ID	Char. Array	DEVC and/or CTRL input IDs		
LATCH	Logical	Control function changes state only once		.TRUE.
N	Integer	Number of .TRUE. INPUTs		1
ON_BOUND	Character	Active edge of a DEADBAND		LOWER
RAMP_ID	Character	ID for a CUSTOM ramp controller		
SETPOINT(2)	Real	Lower and upper bound of a DEADBAND		

13.4 DEVC (Device Parameters)

Table 13.4: For more information see Section 11.1.

DEVC (Device Parameters)				
BYPASS_FLOWRATE	Real	Aspiration smoke detector parameter	kg/s	0
CTRL_ID	Character	Associated CTRL line		
DELAY	Real	Transport time for an aspiration detector	s	0
DEVC_ID	Character	Associated DEVC line for aspiration detector		
DEPTH	Real	Depth into surface for internal wall temp	m	0
FLOWRATE	Real	Suction flowrate for an aspiration detector	kg/s	0
FYI	Character	Comment String (has no effect)		
IOR	Integer	Index of Orientation ($\pm$1,$\pm$2,$\pm$3)		
ID	Character	Identifying label for output		
INITIAL_STATE	Logical	Initial state of device		.FALSE.
LATCH	Logical	Device cannot change state multiple times		.TRUE.
ORIENTATION	Real Triplet	Direction vector		0,0,-1
PROP_ID	Character	Associated PROPerty line		
QUANTITY	Character	Name of Quantity to output		
ROTATION	Real Triplet	Rotation Angle	deg	0
SETPOINT	Real	Value at which device changes state		
STATISTICS	Character	See Section 12.2.3		
SURF_ID	Character	See Section 12.2.3		
TRIP_DIRECTION	Integer	Sign of derivative at first state change		1
XB(6)	Real Sextuplet	Coordinates of non-point measurement	m	
XYZ	Real Triplet	Physical coordinates	m	

13.5 DUMP (Output Parameters)

Table 13.5: For more information see Section 12.1.

DUMP (Output Parameters)				
COLUMN_DUMP_LIMIT	Logical	Limit text output to 255 columns		.TRUE.
DT_BNDF	Real	Boundary dump interval	s	$2\Delta t$/NFRAMES
DT_CTRL	Real	Control status dump interval	s	Δt/NFRAMES
DT_DEVC	Real	Device output dump interval	s	Δt/NFRAMES
DT_HRR	Real	Heat release dump interval	s	Δt/NFRAMES
DT_ISOF	Real	Iso-surface dump interval	s	Δt/NFRAMES
DT_MASS	Real	Mass diagnostic dump interval	s	Δt/NFRAMES
DT_PART	Real	Particle dump interval	s	Δt/NFRAMES
DT_PL3D	Real	PLOT3D dump interval	s	Δt/5
DT_PROF	Real	Profile dump interval	s	Δt/NFRAMES
DT_RESTART	Real	Restart core dump interval	s	1000000.
DT_SLCF	Real	Slice dump interval	s	Δt/NFRAMES
FLUSH_FILE_BUFFERS	Logical	Periodically empty file buffers		.TRUE.
MASS_FILE	Logical	Flag for species MASS file		.FALSE.
MAXIMUM_DROPLETS	Integer	Max particles per mesh		500000
NFRAMES	Integer	Number of Frames of output data		1000
PLOT3D_QUANTITY(5)	Char. Quint	Names of PLOT3D Quantities		See Section 12.2.8
SMOKE3D	Logical	Flag for 3D Smoke Visualization		.TRUE.
WRITE_XYZ	Logical	Flag for writing PLOT3D .xyz file		.FALSE.

Δt=T_END-T_BEGIN

13.6 HEAD (Header Parameters)

Table 13.6: For more information see Section 6.1.

HEAD (Header Parameters)				
CHID	Character	Job Identification String		'output'
FYI	Character	Comment String (has no effect)		
TITLE	Character	Title for job		

13.7 HOLE (Obstruction Cutout Parameters)

Table 13.7: For more information see Section 7.2.

HOLE (Obstruction Cutout Parameters)				
COLOR	Character	Color name of obstruction color		
CTRL_ID	Character	ID of ConTRoL to control hole's existence		
DEVC_ID	Character	ID of DEViCe to control hole's existence		
FYI	Character	Comment String (has no effect)		
RGB(3)	Integer Triplet	Color indices (0 - 255) for resulting obstruction(s)		
TRANSPARENCY	Real	Transparency of obstruction		
XB(6)	Real Sextuplet	Physical coordinates	m	

13.8 INIT (Initial Conditions)

Table 13.8: For more information see Section 6.5.

INIT (Initial Conditions)				
DENSITY	Real	Initial value of density	kg/m^3	Ambient
MASS_FRACTION(II)	Real Array	Initial value of species II	kg/kg	Ambient
TEMPERATURE	Real	Initial value of temperature	°C	TMPA
XB(6)	Real Sextuplet	Coordinates	m	

13.9 ISOF (Isosurface Parameters)

Table 13.9: For more information see Section 12.2.7.

ISOF (Isosurface Parameters)				
FYI	Character	Comment String (has no effect)		
QUANTITY	Character	Quantity to visualize		
VALUE(I)	Real Array	Contour value(s)		

13.10 MATL (Material Properties)

Table 13.10: For more information see Section 8.4.

MATL (Material Properties)				
A	Real	Pre-exponential factor	1/s	1E13
ABSORPTION_COEFFICIENT	Real	Absorption Coefficient	1/m	50000.
BOILING_TEMPERATURE	Real	Boiling temperature	°C	5000.
CONDUCTIVITY	Real	Thermal conductivity	W/m/K	0.1
CONDUCTIVITY_RAMP	Character	Ramp ID for conductivity		
DENSITY	Real	Solid mass per unit volume	kg/m^3	500.
E	Real	Activation energy	kJ/kmol	
EMISSIVITY	Real	Emissivity		0.9
FYI	Character	Comment String (has no effect)		
HEAT_OF_COMBUSTION	Real	Heat of combustion	kJ/kg	
HEAT_OF_REACTION	Real	Heat of reaction	kJ/kg	0.
ID	Character	IDentifier		
THRESHOLD_TEMPERATURE	Real	Threshold temperature	°C	-273.15
N_REACTIONS	Character	Number of Reactions		0
N_S	Real	Exponent of mass fraction		1.
N_T	Real	Exponent of temperature		0.
NU_FUEL	Real	Fuel Yield	kg/kg	0.
NU_RESIDUE	Real	Residue Yield	kg/kg	0.
NU_WATER	Real	Steam Yield	kg/kg	0.
REFERENCE_RATE	Real	Reaction rate at ref. temp.	s^{-1}	0.10
REFERENCE_TEMPERATURE	Real	Reference temperature	°C	
RESIDUE	Character	ID of residue MATL		
SPECIFIC_HEAT	Real	Specific heat	kJ/kg/K	1.0
SPECIFIC_HEAT_RAMP	Character	Ramp ID for specific heat		

13.11 MESH (Mesh Parameters)

Table 13.11: For more information see Section 6.3.

MESH (Mesh Parameters)				
COLOR	Character	Mesh Line Color		'BLACK'
CYLINDRICAL	Logical	2-D Axi-symmetric calculation		.FALSE.
ID	Character	MESH IDentifier		
IJK	Integer Triplet	No. cells in x, y, and z directions		10
FYI	Character	Comment String (has no effect)		
RGB	Integer Triplet	Color indices (0-255)		0,0,0
SYNCHRONIZE	Logical	Sync. time steps of multiple meshes		.TRUE.
XB	Real Sextuplet	Min/Max Coordinates of the MESH	m	0,1,0,1,0,1

13.12 MISC (Miscellaneous Parameters)

Table 13.12: For more information see Section 6.4.

MISC (Miscellaneous Parameters)				
ASSUMED_GAS_TEMPERATURE	Real	See Section 8.7		
BACKGROUND_SPECIES	Character	See Section 9.2		'AIR'
BAROCLINIC	Logical	Baroclinic torque correction		.FALSE.
BNDF_DEFAULT	Logical	See Section 12.2.6		.TRUE.
CFL_MAX	Real	See Section 6.4.6		1.0
CFL_MIN	Real	See Section 6.4.6		0.8
CSMAG	Real	Smagorinsky constant		0.20
CONDUCTIVITY	Real	See Section 9.2	W/m/K	
CO_PRODUCTION	Logical	See Section 9.1.1		.FALSE.
DNS	Logical	Direct Numerical Simulation		.FALSE.
FYI	Character	Comment String (has no effect)		
GVEC	Real triplet	Gravity vector	m/s^2	0,0,-9.81
H_FIXED	Real	See Section 8.7		
HUMIDITY	Real	Relative Humidity	%	40.
ISOTHERMAL	Logical	Isothermal calculation		.FALSE.
LAPSE_RATE	Real	See Section 8.2.5	°C/m	0
LES	Logical	Large Eddy Simulation		.TRUE.
MW	Real	Molecular Weight (Section 9.2)	g/mol	
NOISE	Logical	Toggle initial noise on and off		.TRUE.
PR	Real	Prandtl number (LES only)		0.5
P_INF	Real	Ambient pressure	Pa	101325
POROUS_FLOOR	Logical	See Section 11.3.1		.TRUE.
RADIATION	Logical	Radiation calculation flag		.TRUE.
RAMP_GX	Character	Time function, x comp. of gravity		
RAMP_GY	Character	Time function, y comp. of gravity		
RAMP_GZ	Character	Time function, z comp. of gravity		
RESTART	Logical	Restart previous calculation		.FALSE.
RESTART_CHID	Character	Restart file CHID		CHID
SC	Real	Schmidt number (LES only)		0.5
SOLID_PHASE_ONLY	Logical	See Section 8.7		.FALSE.
STRATIFICATION	Logical	See Section 8.2.5		.TRUE.
SUPPRESSION	Logical	See Section 9.1.1		.TRUE.
SURF_DEFAULT	Character	Default SURFace type		'INERT'
TEXTURE_ORIGIN(3)	Char. Triplet	See Section 8.6.1	m	(0.,0.,0.)
THICKEN_OBSTRUCTIONS	Logical	See Section 7.1		.FALSE.
TMPA	Real	Ambient Temperature	°C	20.
U0,V0,W0	Reals	Prevailing velocity field	m/s	0.
VISCOSITY	Real	See Section 9.2	kg/m/s	
VN_MAX	Real	See Section 6.4.6		1.0
VN_MIN	Real	See Section 6.4.6		0.8

13.13 OBST (Obstruction Parameters)

Table 13.13: For more information see Section 7.1.

OBST (Obstruction Parameters)				
ALLOW_VENT	Logical	Allow vents on obstruction		.TRUE.
BNDF_FACE(-3:3)	Logical Array	See Section 12.2.6		.TRUE.
BNDF_OBST	Logical	See Section 12.2.6		.TRUE.
COLOR	Character	Color name of obstruction color		
CTRL_ID	Character	ID of Controlling ConTRoL		
DEVC_ID	Character	ID of Controlling DEViCe		
FYI	Character	Comment String (has no effect)		
OUTLINE	Logical	Draw as Outline		.FALSE.
PERMIT_HOLE	Logical	Allow a Hole		.TRUE.
REMOVABLE	Logical	Allow obstruction to be removed		.TRUE.
RGB(3)	Integer Triplet	Color indices (0 - 255)		
SAWTOOTH	Logical	See Section 7.1.1		.TRUE.
SURF_ID	Character	Associated Surface		
SURF_IDS(3)	Character Triplet	Associated Surfaces (top,side,bot.)		
SURF_ID6(6)	Character Sextuplet	Associated Surfaces (like XB)		
THICKEN	Logical	Force at least one cell thick		.FALSE.
TEXTURE_ORIGIN(3)	Real Triplet	See Section 8.6.1	m	(0.,0.,0.)
TRANSPARENCY	Real	Transparency indicator		1
XB(6)	Real Sextuplet	Min/Max Physical coordinates	m	

13.14 PART (Lagrangian Particles/Droplets)

Table 13.14: For more information see Section 10.

PART (Lagrangian Particles/Droplets)				
AGE	Real	Droplet lifetime	s	100000.
COLOR	Character	Default color of droplets		'BLACK'
DENSITY	Real	Droplet density	kg/m^3	1000.
DIAMETER	Real	Median Volumetric Diameter	μm	500.
DROPLETS_PER_SECOND	Integer	Drops per second per head		1000
DT_INSERT	Real	Time between particle insertions	s	0.05
EVAPORATE	Logical	Assume liquid evaporation		.TRUE.
FYI	Character	Comment String (has no effect)		
FUEL	Logical	Liquid Fuel		.FALSE.
GAMMA_D	Real	Parameter for size distribution		2.4
HEAT_OF_COMBUSTION	Real	Heat of Combustion	kJ/kg	
HEAT_OF_VAPORIZATION	Real	Latent Heat of Vaporization	kJ/kg	2259.
HORIZONTAL_VELOCITY	Real	Droplet speed, horizontal	m/s	0.2
ID	Character	Identifier		
INITIAL_TEMPERATURE	Real	Initial Temperature	°C	TMPA
MASSLESS	Logical	Massless tracers		.FALSE.
MASS_PER_VOLUME	Real	Droplet mass per unit volume	kg/m^3	1.
MAXIMUM_DIAMETER	Real	Above which droplet breaks up	μm	∞
MINIMUM_DIAMETER	Real	Below which droplet evaporates	μm	20.
MELTING_TEMPERATURE	Real	Melting Temperature	°C	0.
MONODISPERSE	Logical	Uniform droplet size		.FALSE.
NUMBER_INITIAL_DROPLETS	Integer	Number of droplets at start		0
QUANTITIES(10)	Character	Quantities for coloring		
RGB(3)	Integers	Color indices (0-255)		
SAMPLING_FACTOR	Integer	Filter for output file		1
SPEC_ID	Character	Name of gas species		
SPECIFIC_HEAT	Real	Droplet specific heat	kJ/kg/K	4.184
STATIC	Logical	Stationary Particles		.FALSE.
VAPORIZATION_TEMPERATURE	Real	Liquid Droplet Boiling Temp	°C	100.
VERTICAL_VELOCITY	Real	Droplet speed, vertical	m/s	0.5
XB(6)	Real	Initial particle placement region	m	
WATER	Logical	Water Droplet		.FALSE.

13.15 PROF (Wall Profile Parameters)

Table 13.15: For more information see Section 12.2.4.

PROF (Wall Profile Parameters)				
IOR	Real	Orientation of wall surface		
ID	Character	Identifier		
FYI	Character	Comment String (has no effect)		
QUANTITY	Character	Name of output quantity		
XYZ	Real Triplet	Coordinates of wall surface	m	

13.16 PROP (Device Properties)

Table 13.16: For more information see Section 11.3.

PROP (Device Properties)				
ACTIVATION_TEMPERATURE	Real	Threshold link temperature	°C	74
ACTIVATION_OBSCURATION	Real	Threshold value of obscuration	%/m	3.28
ALPHA_C	Real	Smoke detector parameter		1.8
ALPHA_E	Real	Smoke detector parameter		0.0
BETA_C	Real	Smoke detector parameter		1.0
BETA_E	Real	Smoke detector parameter		1.0
BEAD_DIAMETER	Real	Diameter of TC bead	m	0.001
BEAD_EMISSIVITY	Real	Emissivity of TC bead		0.85
C_FACTOR	Real	Sprinkler activation parameter		0.
CHARACTERISTIC_VELOCITY	Real	See Section 12.3.9	m/s	1.0
DROPLET_VELOCITY	Real	Initial droplet velocity	m/s	5.0
FLOW_RATE	Real	Sprinkler or nozzle flow rate	L/min	
FLOW_RAMP	Character	Time RAMP for flow		
FLOW_TAU	Real	Time constant for flow		0.0
GAUGE_TEMPERATURE	Real	See Section 12.3.5	°C	TMPA
ID	Character	IDentifier		
INITIAL_TEMPERATURE	Real	Initial link temperature	°C	TMPA
K_FACTOR	Real	Flow parameter	L/min/atm$^{1/2}$	1.
LENGTH	Real	Smoke detector parameter		1.8
OFFSET	Real	Droplet offset distance	m	0.05
OPERATING_PRESSURE	Real	Sprinkler pipe pressure	atm	1.
PART_ID	Character	Name of associated PART line		
QUANTITY	Character	Name of associated output		
RTI	Real	Response Time Index	$\sqrt{\text{m s}}$	100.
SMOKEVIEW_ID	Character	Name of drawn object		
SPRAY_ANGLE(2)	Real	Cone angles for water spray	deg	60.,75.
SPRAY_PATTERN_TABLE	Character	TABL for spray pattern		

13.17 RADI (Radiation Parameters)

Table 13.17: For more information see Section 9.4.

RADI (Radiation Parameters)				
ANGLE_INCREMENT	Integer	Number of angles skipped per update		5
CH4_BANDS	Logical	Include extra fuel bands		.FALSE.
KAPPA0	Real	Constant absorption coefficient	1/m	0
NMIEANG	Integer	Number of polar angles		15
NUMBER_RADIATION_ANGLES	Integer	Number of solid angles		104
PATH_LENGTH	Real	Path length for radiation calc.	m	
RADIATIVE_FRACTION	Real	Radiative Loss Fraction		0.35
RADTMP	Real	Assumed radiative source temp.	°C	900
TIME_STEP_INCREMENT	Integer	Number time steps skipped		3
WIDE_BAND_MODEL	Logical	Non-gray gas assumption		.FALSE.

13.18 RAMP (Ramp Function Parameters)

Table 13.18: For more information see Section 8.5.

RAMP (Ramp Function Parameters)				
FYI	Character	Comment String (has no effect)		
F	Real	Function value		
FYI	Character	Comment String (has no effect)		
ID	Character	Identifier		
T	Real	Time (or Temperature)	s (or °C)	

13.19 REAC (Reaction Parameters)

Table 13.19: For more information see Section 9.1.

REAC (Reaction Parameters)				
BOF	Real	Pre-exponential Factor (Finite Rate)	$cm^3/mol/s$	
C	Real	Carbon atoms in fuel		3
CO_YIELD	Real	Fraction of CO from the fuel	kg/kg	0
CRITICAL_FLAME_TEMPERATURE	Real	Suppression criterion	°C	1427
E	Real	Activation Energy (Finite Rate)	kJ/kmol	
EPUMO2	Real	Energy per Unit Mass Oxygen	kJ/kg	13100
FUEL	Character	Name of Fuel (Finite Rate)		
FYI	Character	Comment String (has no effect)		
H	Real	Hydrogen atoms in fuel		8
H2_YIELD	Real	Fraction of H_2 from the fuel	kg/kg	0
HEAT_OF_COMBUSTION	Real	Energy per Unit Mass Fuel	kJ/kg	
HRRPUA_SHEET	Real	Upper limit on flame HRR	kW/m^2	200.
ID	Character	Identifier		
IDEAL	Logical	Adjust for minor product yields		.FALSE.
MASS_EXTINCTION_COEFFICIENT	Real	Visibility parameter	m^2/kg	8700.
MAXIMUM_VISIBILITY	Real	Visibility parameter	m	30
MW_OTHER	Real	Molecular Weight of OTHER	g/mol	28
N	Real	Nitrogen atoms in the fuel		0
N_S(N)	Real	Arrhenius Exponents (Finite Rate)		
NU(N)	Real	Reaction stoichiometry (Finite Rate)		
O	Real	Oxygen atoms in the fuel		0
OTHER	Real	Other atoms in the fuel		0
OXIDIZER	Character	Name of Oxidizer (Finite Rate)		
SOOT_YIELD	Real	Fraction of soot from the fuel	kg/kg	0.01
SOOT_H_FRACTION	Real	Atom fraction of hydrogen in soot		0.1
VISIBILITY_FACTOR	Real	Visibility parameter		3
X_O2_LL	Real	Lower Oxygen Limit	mol/mol	0.15
Y_F_INLET	Real	Mass Frac. of Fuel in Burner	kg/kg	1.0
Y_F_LFL	Real	Lower Fuel limit (mass fraction)	kg/kg	0.0
Y_O2_INFTY	Real	Ambient Oxygen Mass Frac.	kg/kg	0.23

13.20 SLCF (Slice File Parameters)

Table 13.20: For more information see Section 12.2.5.

SLCF (Slice File Parameters)				
FYI	Character	Comment String (has no effect)		
MESH_NUMBER	Integer	Save only slices in this mesh		
PBX	Real	x-plane to save slice file		
PBY	Real	y-plane to save slice file		
PBZ	Real	z-plane to save slice file		
QUANTITY	Character	Name of Quantity to display		
VECTOR	Logical	Include flow vectors		.FALSE.
XB(6)	Real Sextuplet	Min/Max coordinates of region to save	m	

13.21 SPEC (Species Parameters)

Table 13.21: For more information see Section 9.2.

SPEC (Species Parameters)				
ABSORBING	Logical	Gas species abosrbs radiation		.FALSE.
CONDUCTIVITY	Real	Conductivity k	W/m/K	
DIFFUSIVITY	Real	Diffusivity D	m^2/s	
EPSILONKLJ	Real	Leonard-Jones Parameter		0
FYI	Character	Comment String (has no effect)		
ID	Character	Name of species		
MASS_FRACTION_0	Real	Initial mass fraction		0
MW	Real	Molecular Weight	g/mol	29.
SIGMALJ	Real	Leonard-Jones Parameter		0
VISCOSITY	Real	Dynamic Viscosity *mu*	kg/m/s	

13.22 SURF (Surface Properties)

Table 13.22: For more information see Section 8.2.

SURF (Surface Properties)				
ADIABATIC	Logical	Adiabatic thermal BC		.FALSE.
BACKING	Character	Back boundary condition		'VOID'
BURN_AWAY	Logical	Object can vanish		.FALSE.
CELL_SIZE_FACTOR	Real	See Section 8.4.5		1.0
COLOR	Character	Surface Color		
CONVECTIVE_HEAT_FLUX	Real	Heat flux at surface	kW/m^2	0.
DUCT_PATH	Integer Pair	Pressure Zones for fans		0,0
E_COEFFICIENT	Real	Extinguishing coefficient	1/s	0.
EMISSIVITY	Real	Emissivity		0.9
EXTERNAL_FLUX	Real	Heat flux to surface	kW/m^2	0.
FYI	Character	Comment String		
GEOMETRY	Character	Geometry type		'CARTESIAN'
HEAT_OF_VAPORIZATION	Real	For specified HRR only	kJ/kg	0.
HRRPUA	Real	HRR Per Unit Area	kW/m^2	0.
ID	Character	IDentifier		
IGNITION_TEMPERATURE	Real	Ignition temperature	°C	5000.
LEAK_PATH	Integer Pair	Pressure Zones for leakage		
MASS_FLUX(I)	Real Array	For species I		0.
MASS_FLUX_TOTAL	Real	Total Mass Flux		
MASS_FRACTION(I)	Real Array	For species I		
MATL_ID	Char. 2D Array	(Layer,Component)		
MATL_MASS_FRACTION	Real 2D Array	(Layer,Component)		
MAX_PRESSURE	Real	Max over-pressure for fan	Pa	1E12
MLRPUA	Real	Mass loss rate per unit area	kg/m^{2}s	0.
NPPC	Integer	Number of particles per cell		1
PARTICLE_MASS_FLUX	Real	See Section 10.2	kg/m^2 s	0.
PART_ID	Character	Lagrangian Particle ID		
POROUS	Logical	Non-solid boundary		.FALSE.
PLE	Real	Atmospheric profile exponent		0.3
PROFILE	Character	Name of velocity profile		
RAMP_MF(I)	Character	Ramp ID for species I		
RAMP_Q	Character	Ramp ID for HRR		
RAMP_T	Character	Ramp ID for temp.		
RAMP_V	Character	Ramp ID for velocity		
RGB(3)	Integer Triplet	Color indices (0-255)		255,204,102
SHRINK	Logical	Shrinking material		.TRUE.
SLIP_FACTOR	Real	Velocity Slip Condition		0.5
STRETCH_FACTOR	Real	See Section 8.4.5		2.0
TAU_MF(I)	Real Array	Ramp time for species I	s	1.
TAU_Q	Real	Ramp time for HRR	s	1.
TAU_T	Real	Ramp time for temp.	s	1.

TAU_V	Real	Ramp time for velocity	s	1.
TEXTURE_HEIGHT	Real	Height of texture image	m	1.
TEXTURE_MAP	Character	Name of texture map file		
TEXTURE_WIDTH	Real	Width of texture image	m	1.
THICKNESS(IL)	Real Array	Thickness of Layer IL	m	0.
TMP_BACK	Real	Back surface temperature BC	°C	20.
TMP_FRONT	Real	Front surface temperature	°C	20.
TMP_INNER	Real	Initial solid temperature	°C	20.
TRANSPARENCY	Real	Transparency of obstruction	1	
VEL	Real	Normal velocity	m/s	0.
VEL_T	Real Pair	Tangential velocity comps.	m/s	0.
VOLUME_FLUX	Real	Normal velocity x vent area	m^3/s	0.
Z0	Real	Atmospheric profile origin	m	10.

13.23 TABL (Table Parameters)

Table 13.23: For more information see Section 8.5.

TABL (Table Parameters)				
ID	Character	IDentifier		
FYI	Character	Comment String (has no effect)		
TABLE_DATA	Real Array	Data for one row of the table		

13.24 TIME (Time Parameters)

Table 13.24: For more information see Section 6.2.

TIME (Time Parameters)				
DT	Real	Initial time step	s	
FYI	Character	Comment String (has no effect)		
SYNCHRONIZE	Logical	Sync time step of multiple meshes		.TRUE.
T_BEGIN	Real	Starting time for calculation	s	0.
T_ENDorTWFIN	Real	Ending time for calculation	s	1
WALL_INCREMENT	Integer	Time steps between 1D wall solution updates		2

13.25 TRNX, TRNY, TRNZ (MESH Transformations)

Table 13.25: For more information see Section 6.3.3.

TRNX, TRNY, TRNZ (MESH Transformations)				
CC	Real	Computational coordinate	m	
FYI	Character	Comment String (has no effect)		
IDERIV	Integer	Order of polynomial transformation		
MESH_NUMBER	Integer	Number of mesh to transform		
PC	Real	Physical coordinate or derivative		

13.26 VENT (Vent Parameters)

Table 13.26: For more information see Section 7.3.

VENT (Vent Parameters)				
COLOR	Character	See Section 8.6		
CTRL_ID	Character	ID of Control Function		
DEVC_ID	Character	ID of Controlling Device		
FYI	Character	Comment String (has no effect)		
IOR	Integer	Orientation Index		
MASS_FRACTION(N)	Real Array	Mass Fraction of species N at OPEN vent	kg/kg	
MB	Character	Mesh Boundary		
OUTLINE	Logical	Draw vent as outline		.FALSE.
PBX, PBY, PBZ	Real	Coordinate Plane		
RGB(3)	Integer Triplet	See Section 8.6		
SPREAD_RATE	Real	See Section 8.2.6	m/s	0.0
SURF_ID	Character	Associated Surface		'INERT'
TEXTURE_ORIGIN(3)	Real Triplet	See Section 8.6.1	m	(0.,0.,0.)
TMP_EXTERIOR	Real	Temperature at OPEN vent	°C	
TRANSPARENCY	Real	Transparency indicator		1.0
XB(6)	Real Sextuplet	Min/Max physical coordinates	m	
XYZ(3)	Real Triplet	See Section 8.2.6	m	

13.27 ZONE (Pressure Zone Parameters)

Table 13.27: For more information see Section 8.3.

ZONE (Pressure Zone Parameters)				
ID	Character	IDentifier		
LEAK_AREA(N)	Real	Leakage area to pressure zone N	m^2	0
XB(6)	Real Sextuplet	Coordinates of Zone	m	

Chapter 14

Conversion of Old Input Files to FDS 5

Many changes and improvements have been made in the latest release FDS 5. To make an FDS 4 input data file compatible with the new FDS 5 application, a few changes must be made to the file. This appendix will point out all the changes that need to be made to convert an FDS 4.x input file to the new FDS 5.x format.

14.1 Numerical Domain Parameters: GRID and PDIM

In previous versions, the computational domain and numerical mesh were specified via lines of the form:

```
&GRID IBAR=30, JBAR=20, KBAR=10 /
&PDIM XBAR0=0.0, XBAR=3.0, YBAR0=0.0, YBAR=2.0, ZBAR0=0.0, ZBAR=1.0 /
```

In FDS 5, these two lines are now written via the single line:

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

Rules for multiple meshes and mesh transformations still apply.

14.2 Obstructions, Vents, and Holes: OBST, VENT, and HOLE

The syntax for these lines is fairly similar to past versions, with the following exceptions:

- For a VENT that spans an entire mesh boundary, CB='XBAR0' is now MB='XMIN'. The character string 'XBAR' is now 'XMAX'. The same applies for the y and z coordinate parameters.
- Control parameters like T_ACTIVATE, HEAT_REMOVE, *etc.*, are now consolidated into DEVC_ID and CTRL_ID. In brief, any change to an obstruction, vent, or hole is tied to a specific device or control function. See Sections 11.1 and 11.5 for details.

14.3 Surface Parameters: SURF

The most significant change to the input file format has been splitting of the SURF line. In past versions, the SURF namelist group contained all the information about a particular boundary type – its material properties, color, thickness, and so on. However, in FDS 5, solid boundaries can now consist of multiple layers of materials, making the old SURF line too cumbersome to specify. Instead, there is a new namelist group called MATL that just contains the properties of a given material. What used to be

```
&SURF ID        = 'BRICK WALL'
      RGB       = 0.6,0.2,0.2
      KS        = 0.69
      C_P       = 0.84
      DENSITY   = 1600.
      BACKING   = 'EXPOSED'
      THICKNESS = 0.20 /
```

is now given by two input lines:

```
&MATL ID             = 'BRICK'
      CONDUCTIVITY   = 0.69
      SPECIFIC_HEAT  = 0.84
      DENSITY        = 1600. /

&SURF ID        = 'BRICK WALL'
      MATL_ID   = 'BRICK'
      RGB       = 166,41,41
      BACKING   = 'EXPOSED'
      THICKNESS = 0.20 /
```

The surface is still specified the same way as before, for example:

```
&OBST XB=0.1, 5.0, 1.0, 1.2, 0.0, 1.0, SURF_ID='BRICK WALL' /
```

Notice the change in the names of the thermal properties `KS` and `C_P` to `CONDUCTIVITY` and `SPECIFIC_HEAT`, respectively. Notice that the color `RGB` is now specified via integers between 0 and 255, instead of real numbers between 0.0 and 1.0. Better yet, just use the `COLOR` Table 8.1.

14.4 Reaction Parameters: `REAC`

For most applications, the specification of the combustion reaction has become easier. In past versions, you needed to specify the fuel, its molecular weight, soot and/or CO yields, and the ideal stoichiometry of the reaction:

```
&REAC ID                = 'PROPANE'
      FYI               = 'C_3 H_8'
      MW                = 44.
      SOOT_YIELD        = 0.01
      NU_O2             = 5.
      NU_CO2            = 3.
      NU_H2O            = 4. /
```

Now, you just need to describe the composition of the fuel molecule and any non-ideal product yield. FDS 5 computes what it needs based on this information.

```
&REAC ID                = 'PROPANE'
      SOOT_YIELD        = 0.01
      C                 = 3.
      H                 = 8. /
```

14.5 Device Parameters: SPRK, SMOD, HEAT, THCP

Past versions of FDS had a variety of ways to specify devices. For example, a sprinkler was specified via a line of the form:

```
&SPRK XYZ=4.5,6.7,3.6, MAKE='Acme_K-17', LABEL='spk_34' /
```

which located the sprinkler at XYZ and indicated that the sprinkler's properties were listed in a file called **Acme_K-17.spk**. Smoke and heat detectors were specified via lines of the form:

```
&SMOD XYZ=4.5,6.7,3.6, LENGTH=2.6, ACTIVATION_OBSCURATION=1.4, LABEL='sd_34' /
&HEAT XYZ=4.5,6.7,3.6, RTI=45., ACTIVATION_TEMPERATURE=74., LABEL='hd_39' /
```

In FDS 5, these devices are all specified in the same way:

```
&PROP ID='Acme_K-17', QUANTITY='SPRINKLER LINK TEMPERATURE', RTI=148., C_FACTOR=0.7,
      ACTIVATION_TEMPERATURE=74., PART_ID='water drops', FLOW_RATE=189.3,
      DROPLET_VELOCITY=10., SPRAY_ANGLE=30.,80.   /

&DEVC ID='spk_34', XYZ=4.5,6.7,3.6, PROP_ID='Acme_K-17' /
```

Point output via "thermocouples" (THCPs) are now given by "devices" (DEVCs):

```
&DEVC XYZ=0.7,0.9,2.1, QUANTITY='WALL_TEMPERATURE', IOR=-2, ID='probe_2' /
```

The syntax of the old THCP namelist group is almost the same. Just swap DEVC for THCP, and change LABEL to ID. In FDS 5, any input record is identified via its ID.

Part III

Sample Cases and Verification

Chapter 15

Forms of Verification

ASTM E 1355 [13] outlines methods to evaluate the mathematical and numerical robustness of deterministic fire models. This process, often referred to as *model verification*, ensures the accuracy of the numerical solution of the governing equations. The methods include comparison with analytical solutions, code checking, and numerical tests.

15.1 Comparison with Analytical Solutions

Most complex combustion processes, including fire, are turbulent and time-dependent. There are no closed-form mathematical solutions for the fully-turbulent, time-dependent Navier-Stokes equations. CFD provides an approximate solution for the non-linear partial differential equations by replacing them with discretized algebraic equations that can be solved using a powerful computer. While there is no general analytical solution for fully-turbulent flows, certain sub-models address phenomenon that do have analytical solutions, for example, one-dimensional heat conduction through a solid. These analytical solutions can be used to test sub-models within a complex code such as FDS. The developers of FDS routinely use such practices to verify the correctness of the coding of the model [14, 15]. Such verification efforts are relatively simple and routine and the results may not always be published nor included in the documentation. Examples of routine analytical testing include:

- The radiation solver has been verified with scenarios where simple objects, like cubes or flat plates, are positioned in simple, sealed compartments. All convective motion is turned off, the object is given a fixed surface temperature and emissivity of one (making it a black body radiator). The heat flux to the cold surrounding walls is recorded and compared to analytical solutions. These studies help determine the appropriate number of solid angles to be set as the default.

- Solid objects are heated with a fixed heat flux, and the interior and surface temperatures as a function of time are compared to analytical solutions of the one-dimensional heat transfer equation. These studies help determine the number of nodes to use in the solid phase heat transfer model. Similar studies are performed to check the pyrolysis models for thermoplastic and charring solids.

- Early in its development, the hydrodynamic solver that evolved to form the core of FDS was checked against analytical solutions of simplified fluid flow phenomena. These studies were conducted at the National Bureau of Standards (NBS)[1] by Rehm, Baum and co-workers [16, 17, 18, 19]. The emphasis of this early work was to test the stability and consistency of the basic hydrodynamic solver, especially

[1] The National Institute of Standards and Technology (NIST) was formerly known as the National Bureau of Standards.

the velocity-pressure coupling that is vitally important in low Mach number applications. Many numerical algorithms developed up to that point in time were intended for use in high-speed flow applications, like aerospace. Many of the techniques adopted by FDS were originally developed for meteorological models, and as such needed to be tested to assess whether they would be appropriate to describe relatively low-speed flow within enclosures.

- A fundamental decision made by Rehm and Baum early in the FDS development was to use a direct (rather than iterative) solver for the pressure. In the low Mach number formulation of the Navier-Stokes equations, an elliptic partial differential equation for the pressure emerges, often referred to as the Poisson equation. Most CFD methods use iterative techniques to solve the governing conservation equations to avoid the necessity of directly solving the Poisson equation. The reason for this is that the equation is time-consuming to solve numerically on anything but a rectilinear grid. Because FDS is designed specifically for rectilinear grids, it can exploit fast, direct solvers of the Poisson equation, obtaining the pressure field with one pass through the solver to machine accuracy. FDS employs double-precision (8 byte) arithmetic, meaning that the relative difference between the computed and the exact solution of the discretized Poisson equation is on the order of 10^{-12}. The fidelity of the numerical solution of the entire system of equations is tied to the pressure/velocity coupling because often simulations can involve hundreds of thousands of time steps, with each time step consisting of two solutions of the Poisson equation to preserve second-order accuracy. Without the use of the direct Poisson solver, build-up of numerical error over the course of a simulation could produce spurious results. Indeed, an attempt to use single-precision (4 byte) arithmetic to conserve machine memory led to spurious results simply because the error per time step built up to an intolerable level.

15.2 Code Checking

An examination of the structure of the computer program can be used to detect potential errors in the numerical solution of the governing equations. The coding can be verified by a third party either manually or automatically with profiling programs to detect irregularities and inconsistencies [13].

At NIST and elsewhere, FDS has been compiled and run on computers manufactured by IBM, Hewlett-Packard, Sun Microsystems, Digital Equipment Corporation, Apple, Silicon Graphics, Dell, Compaq, and various other personal computer vendors. The operating systems on these platforms include Unix, Linux, Microsoft Windows, and Mac OSX. Compilers used include Lahey Fortran, Digital Visual Fortran, Intel Fortran, IBM XL Fortran, HPUX Fortran, Forte Fortran for SunOS, the Portland Group Fortran, and several others. Each combination of hardware, operating system and compiler involves a slightly different set of compiler and run-time options and a rigorous evaluation of the source code to test its compliance with the Fortran 90 ISO/ANSI standard [20]. Through this process, out-dated and potentially harmful code is updated or eliminated, and often the code is streamlined to improve its optimization on the various machines. However, simply because the FDS source code can be compiled and run on a wide variety of platforms does not guarantee that the numerics are correct. It is only the starting point in the process because it at least rules out the possibility that erratic or spurious results are due to the platform on which the code is running.

Beyond hardware issues, there are several useful techniques for checking the FDS source code that have been developed over the years. One of the most best ways is to exploit symmetry. FDS is filled with thousands of lines of code in which the partial derivatives in the conservation equations are approximated as finite differences. It is very easy in this process to make a mistake. Consider, for example, the finite difference approximation of the thermal diffusion term in the ijkth cell of the three-dimensional grid:

$$(\nabla \cdot k\nabla T)_{ijk} \approx \frac{1}{\delta x}\left[k_{i+\frac{1}{2},jk}\frac{T_{i+1,jk}-T_{ijk}}{\delta x} - k_{i-\frac{1}{2},jk}\frac{T_{ijk}-T_{i-1,jk}}{\delta x}\right] +$$

$$\frac{1}{\delta y}\left[k_{i,j+\frac{1}{2},k}\frac{T_{i,j+1,k}-T_{ijk}}{\delta y}-k_{i,j-\frac{1}{2},k}\frac{T_{ijk}-T_{i,j-1,k}}{\delta y}\right]+$$
$$\frac{1}{\delta z}\left[k_{ij,k+\frac{1}{2}}\frac{T_{ij,k+1}-T_{ijk}}{\delta z}-k_{ij,k-\frac{1}{2}}\frac{T_{ijk}-T_{ij,k-1}}{\delta z}\right]$$

which is written as follows in the Fortran source code:

```
DTDX = (TMP(I+1,J,K)-TMP(I,J,K))*RDXN(I)
KDTDX(I,J,K) = .5*(KP(I+1,J,K)+KP(I,J,K))*DTDX
DTDY = (TMP(I,J+1,K)-TMP(I,J,K))*RDYN(J)
KDTDY(I,J,K) = .5*(KP(I,J+1,K)+KP(I,J,K))*DTDY
DTDZ = (TMP(I,J,K+1)-TMP(I,J,K))*RDZN(K)
KDTDZ(I,J,K) = .5*(KP(I,J,K+1)+KP(I,J,K))*DTDZ

DELKDELT = (KDTDX(I,J,K)-KDTDX(I-1,J,K))*RDX(I) +
.          (KDTDY(I,J,K)-KDTDY(I,J-1,K))*RDY(J) +
.          (KDTDZ(I,J,K)-KDTDZ(I,J,K-1))*RDZ(K)
```

This is one of the simpler constructs because the pattern that emerges within the lines of code make it fairly easy to check. However, a mis-typing of an `I` or a `J`, a plus or a minus sign, or any of a hundred different mistakes can cause the code to fail, or worse produce the wrong answer. A simple way to eliminate many of these mistakes is to run simple scenarios that have perfectly symmetric initial and boundary conditions. For example, put a hot cube in the exact center of a larger cold compartment, turn off gravity, and watch the heat diffuse from the hot cube into the cold gas. Any simple error in the coding of the energy equation will show up almost immediately. Then, turn on gravity, and in the absence of any coding error, a perfectly symmetric plume will rise from the hot cube. This checks both the coding of the energy and the momentum equations. Similar checks can be made for all of the three dimensional finite difference routines. So extensive are these types of checks that the release version of FDS has a routine that generates a tiny amount of random noise in the initial flow field so as to eliminate any false symmetries that might arise in the numerical solution.

The process of adding new routines to FDS is as follows: typically the routine is written by one person (not necessarily a NIST staffer) who takes the latest version of the source code, adds the new routine, and writes a theoretical and numerical description for the FDS Technical Reference Guide, plus a description of the input parameters for the FDS User's Guide. The new version of FDS is then tested at NIST with a number of benchmark scenarios that exercise the range of the new parameters. Provisional acceptance of the new routine is based on several factors: (1) it produces more accurate results when compared to experimental measurement, (2) the theoretical description is sound, and (3) any empirical parameters are obtainable from the open literature or standard bench-scale apparatus. If the new routine is accepted, it is added to a test version of the software and evaluated by external users and/or NIST grantees whose research is related to the subject. Assuming that there are no intractable issues that arise during the testing period, the new routine eventually becomes part of the release version of FDS.

Even with all the code checking performed at NIST, it is still possible for errors to go unnoticed. One remedy is the fact that the source code for FDS is publicly released. Although it consists of on the order of 10,000 lines of Fortran statements, various researchers outside of NIST have been able to work with it, add enhancements needed for very specific applications or for research purposes, and report back to the developers bugs that have been detected. The source code is organized into 14 separate files, each containing subroutines related to a particular feature of the model, like the mass, momentum, and energy conservation equations, sprinkler activation and sprays, the pressure solver, *etc.* The lengthiest routines are devoted to input, output and initialization. Most of those working with the source code do not concern themselves with these lengthy routines but rather focus on the finite-difference algorithm contained in a few of the more

important files. Most serious errors are found in these files, for they contain the core of the algorithm. The external researchers provide feedback on the organization of the code and its internal documentation, that is, comments within the source code itself. Plus, they must compile the code on their own computers, adding to its portability. Some of the work performed by researchers who have modified the source code is discussed in Volume 2. However, most of the routine error reports are via electronic mail and are undocumented. Most of the current error reports involve routines that are not frequently used by the FDS developers. For example, the opening of compartment doors or the breaking of windows, especially upon activation of a heat detector, is a feature of FDS commonly used in the fire protection engineering community but less so at NIST. As a result, numerous reports have been made over the years in which a complicated sequence of events prescribed by the user is not carried out by the program. The errors are easy to fix, but the number of possible permutations of events make it difficult to check them all. Another problem reported by users are scenarios that extend the parameter range beyond which the model was originally conceived. Walls made of foam, fires in refrigerators, gas leaks, fuel spills, *etc.*, are just some of the phenomena that users have attempted to model but have run into difficulty because the model parameters either have never been exercised (*e.g.* very low thermal conductivities) or are not allowed (*e.g.* temperatures below ambient). These reports by the users help to improve and extend the use of the model.

15.3 Numerical Tests

Numerical techniques used to solve the governing equations within a model can be a source of error in the predicted results. The hydrodynamic model within FDS is second-order accurate in space and time. This means that the error terms associated with the approximation of the spatial partial derivatives by finite differences is of the order of the square of the grid cell size, and likewise the error in the approximation of the temporal derivatives is of the order of the square of the time step. As the numerical grid is refined, the "discretization error" decreases, and a more faithful rendering of the flow field emerges. The issue of grid sensitivity is extremely important to the proper use of the model and will be taken up in the next chapter.

A common technique of testing flow solvers is to systematically refine the numerical grid until the computed solution does not change, at which point the calculation is referred to as a Direct Numerical Solution (DNS) of the governing equations. For most practical fire scenarios, DNS is not possible on conventional computers. However, FDS does have the option of running in DNS mode, where the Navier-Stokes equations are solved without the use of sub-grid scale turbulence models of any kind. Because the basic numerical method is the same for LES and DNS, DNS calculations are a very effective way to test the basic solver, especially in cases where the solution is steady-state. Throughout its development, FDS has been used in DNS mode for special applications. For example, FDS (or its core algorithms) have been used at a grid resolution of roughly 1 mm to look at flames spreading over paper in a microgravity environment [21, 22, 23, 24, 25, 26], as well as "g-jitter" effects aboard spacecraft [27]. Simulations have been compared to experiments performed aboard the US Space Shuttle. The flames are laminar and relatively simple in structure, and the comparisons are a qualitative assessment of the model solution. Similar studies have been performed comparing DNS simulations of a simple burner flame to laboratory experiments [28]. Another study compared FDS simulations of a counterflow diffusion flames to experimental measurements and the results of a one-dimensional multi-step kinetics model [29].

Early work with the hydrodynamic solver compared two-dimensional simulations of gravity currents with salt-water experiments [30]. In these tests, the numerical grid was systematically refined until almost perfect agreement with experiment was obtained. Such convergence would not be possible if there were a fundamental flaw in the hydrodynamic solver.

Chapter 16

Verification Test Suite

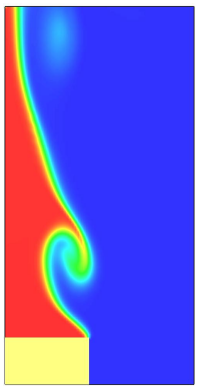
This Chapter contains a set of relatively simple calculations that are used to *verify* the major physical algorithms within FDS. That is, these samples confirm that the equations have been properly coded. They do not imply that the equations actually represent some physical phenomena. That is part of the Validation process. Note that the names in parentheses in each section header correspond to the names of the input files for the cases.

16.1 Hydrodynamics

There are no analytical solutions of the fully-turbulent Navier-Stokes equations, but it is possible to simulate well known fluid flows to determine if the basic fluid flow solver in FDS is working properly.

16.1.1 Axially-Symmetric Helium Plume (helium_2d)

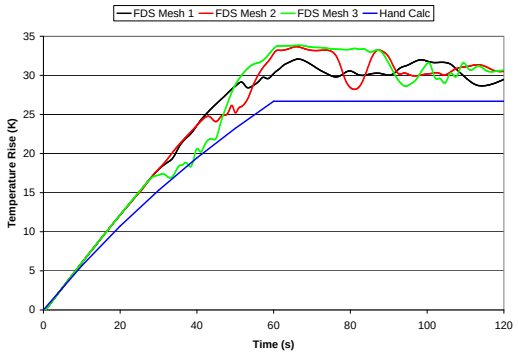
The governing equations solved in FDS are written in terms of a three dimensional Cartesian coordinate system. However, a two dimensional Cartesian or two dimensional cylindrical (axially-symmetric) calculation can be performed by setting the number of cells in the y direction to 1. An example of an axially-symmetric helium plume is shown here, along with the input file:

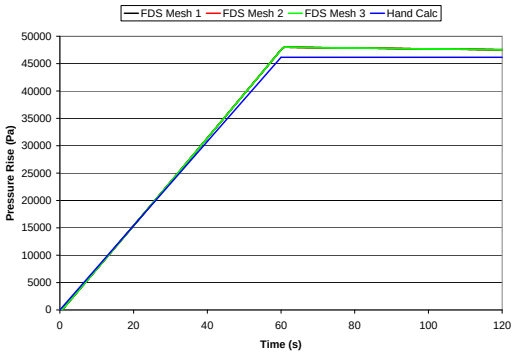


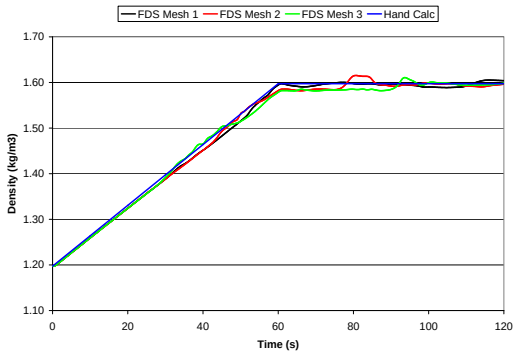
```
&HEAD CHID='helium_2d',TITLE='Axisymmetric Helium Plume' /
&MESH IJK=72,1,144 XB=0.00,0.08,-0.001,0.001,0.00,0.16, CYLINDRICAL=.TRUE. /
&TIME TWFIN=5.0 /
&MISC DNS=.TRUE., ISOTHERMAL=.TRUE. /
&SPEC ID='HELIUM'  /
&SURF ID='HELIUM', VEL=-0.673, MASS_FRACTION(1)=1.0, TAU_MF(1)=0.3 /
&VENT MB='XMAX' ,SURF_ID='OPEN' /
&VENT MB='ZMAX' ,SURF_ID='OPEN' /
&OBST XB= 0.0,0.036,-0.001,0.001,0.00,0.02, SURF_IDS='HELIUM','INERT','INERT' /
&DUMP PLOT3D_QUANTITY(1)='PRESSURE',PLOT3D_QUANTITY(5)='HELIUM' /
&SLCF PBY=0.000,QUANTITY='DENSITY', VECTOR=.TRUE. /
&SLCF PBY=0.000,QUANTITY='HELIUM' /
&TAIL /
```

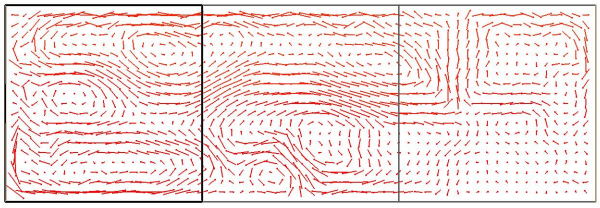
16.1.2 Pressure Rise in a Sealed Enclosure (pressure_rise)

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 kg/m^2/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.`. The domain is divided into three meshes each 1 m long with identicle 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. A hand computation is performed at 10 second intervals using the First Law of Thermodynamics and the equation of state. The figures below show the results of this verification case. As is seen denisty matches exactly showing that FDS is injecting the appropriate quantity of mass and is properly initializing the domain. Pressure rise and temperature rise; however, are overpredicted by 3 % and 12 % respectively. Also a slight drop in pressure is seen from 60 s to 120 s, indicating that the current implementation adiabatic boundary condition has a slight error in it.









16.1.3 Leaks and Fans in a Sealed Enclosure (leak_test and leak_test_2)

A new feature of FDS 5 is the idea of a "pressure zone." Unlike traditional compartment or "zone" fire models, FDS was not designed under the assumption that there exist rooms connected by doors or ducts. Rather, the geometry in FDS is completely specified by the user. However, there are features of simpler models that we want to retain. For example, a leak through a small crack, or the transport of air through a ventilation duct. In the following example, a simple compartment (3.6 m by 2.4 m by 2.4 m) has a small fan at one end and one leak under the door at the other end. It is assumed for this example that the compartment is contained within a larger compartment that is perfectly sealed. The fan draws air into the compartment from the plenum space, increasing the pressure inside and decreasing it outside. A steady state is achieved when the volume flow into and out of the compartment falls into balance.

The volume flow rate of the fan is given by the "fan curve"

$$\dot{V}_{\rm fan} = A_{\rm duct} U_{\rm max}\ {\rm sign}(\Delta p_{\rm max} - \Delta p)\sqrt{\frac{|\Delta p - \Delta p_{\rm max}|}{\Delta p_{\rm max}}} \tag{16.1}$$

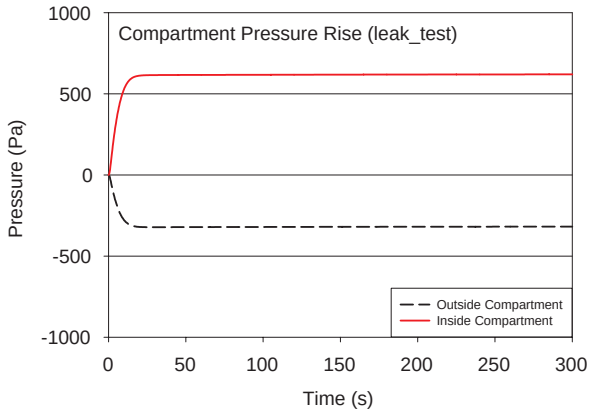
where Δp is the difference in pressure and $A_{\rm duct} = 0.16$ m^2, $U_{\rm max} = 0.1$ m/s, and $\Delta p_{\rm max} = 1000$ Pa. The volume flow due to the leak is given by:

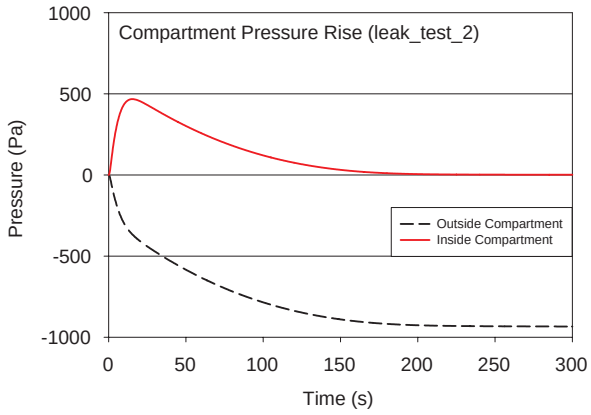
$$\dot{V}_{\rm leak} = A_{\rm leak}\sqrt{\frac{2\Delta p}{\rho_\infty}} \tag{16.2}$$

where $A_{\rm leak} = 0.0001$ m^2 and $\rho_\infty = 1.2$ kg/m^3. After 5 min the pressure difference is 938.2 Pa. The theoretical value, obtained by equating the fan and leak volume flow rates and solving for Δ_p, is 938.9 Pa. The slight difference is due to the fact that the solid boundaries within the interior of the computational domain admit a slight volume flux related to details of the numerical solver.

Just for fun, we add another leak to the compartment, only this time the leak is to the exterior of the entire computational domain, an infinite void at ambient pressure. Now the fan flow rate ought to balance the sum of the flow rates from the two leaks. After 5 min, the pressure difference is 935.2 Pa.

The two cases are summarized in the following plots:





16.1.4 Two Fans in a Wall (fan_test)

Consider two simple compartments divided by a wall with two fans installed, blowing in opposite directions.

16.1.5 Stack Effect (stack_effect)

If the interior temperature of a building is at a different temperature than the surrounding atmosphere, upward or downward air flows within shafts or stairwells connected to the ambient via leakage paths will occur. This phenomena is known as the stack effect. The **stack_effect** test case is a 2D simulation of a 304 m tall building initialized to a temperature of 20 $^\circ$ with the surround ambient temperature initialized to 10 $^\circ$. Two small openings in the building are defined 2.5 m above the ground floor of the building and 2.5 m below the roof of the building.

The initial density stratification is defined by assuming a lapse rate of 0 $^\circ$C/m.

$$\rho_0(z) = \rho_\infty e^{-\frac{gW}{\tilde{R}_0 T_0} z} \tag{16.3}$$

Applying this to the external and internal locations at the lower and upper vents results in densities of 1.2392, 1.1969, 1.1954, and 1.1546 kg/m^3, respectively. FDS computes the same values to within machine precision. Since the openings in the building are equally spaced over its height, the neutral plane of the building will be close to its midpoint. The pressure gradient across the building's wall can be computed as

$$\delta P = \frac{W P_0 g}{R_0} \left(\frac{1}{T_{ambient}} - \frac{1}{T_{building}} \right) h \tag{16.4}$$

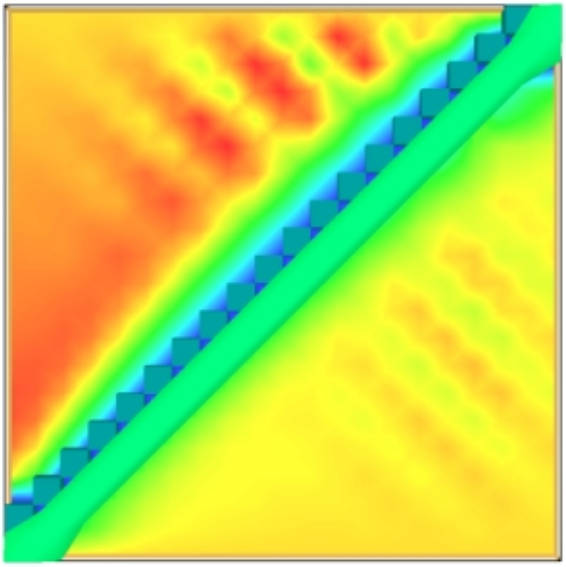
where h is the distance from the neutral plane. Using this pressure gradient in Bernoulli's equation (and assuming it remains constant) results in a velocity of 10.09 m/s through the vent. FDS computes a peak velocity of 10.13 m/s or an error of 0.5 %.

16.1.6 Sawtooth (sawtooth)

Sometimes it is desired to have stair-stepped obstructions representing curved or sloped geometry. A concern is that this may change the flow pattern near the wall. To lessen the impact of stair-stepped boundaries near the edges of the obstructions, one may specify the option `SAWTOOTH=.FALSE.` If `SAWTOOTH` is set to `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 the figure below, 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.



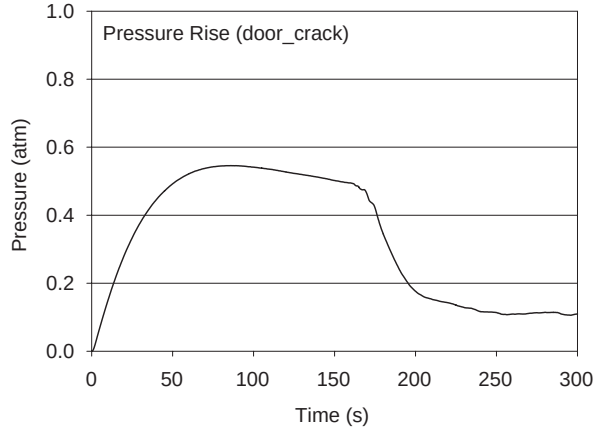
16.2 Combustion

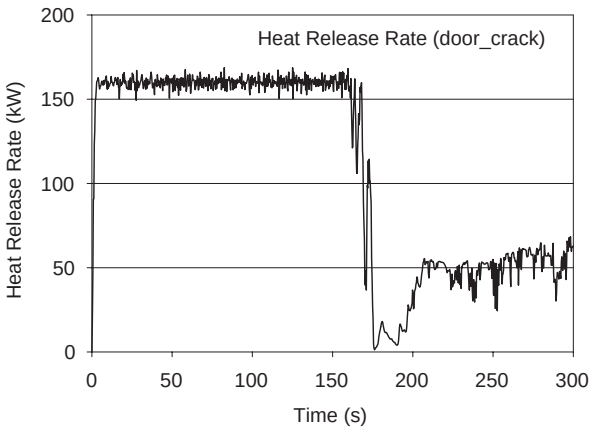
16.2.1 A Simple Under-Ventilated Compartment Fire (door_crack)

This example uses the same simple compartment that was used to test leakage and fan curves in the previous section. Now, we add a small (160 kW) fire, with the same fan and leak under the door. The compartment now opens to the atmosphere, not a sealed plenum. We expect a rapid pressure rise in the compartment due to the effect of the fire and the fan. Initially, the pressure rise is approximately:

$$\frac{d\overline{p}_1}{dt} \approx (\gamma - 1)\,\frac{\dot{Q}}{V} + \gamma \overline{p}\,\frac{\dot{V}}{V} \approx 3200 \text{ Pa/s} \approx 0.03 \text{ atm/s} \tag{16.5}$$

where $\gamma \approx 1.4$, $\dot{Q} = 160{,}000$ W, $V = 20.7$ m^3, and $\dot{V} = 0.016$ m^3/s. In roughly 150 s, the pressure rises about 0.6 atm, at which point the fire dies due to lack of oxygen. Then the pressure decreases, and the fan starts up again (it had stalled due to high pressure in the compartment). The fan, and the leak under the door, increase the oxygen concentration, at least near these openings, and the fuel-rich gases in the compartment continue to burn.





While this case has a number of interesting physical effects, and it *verifies* several features of FDS, it is very important to note the following:

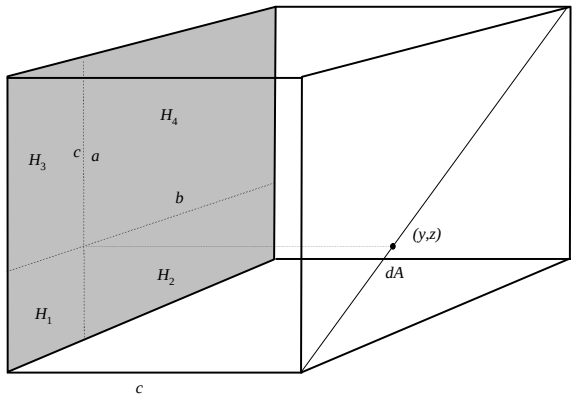
- Although there is smoke seen flowing backwards out the fan duct, in reality there would have been much more. Most conventionally built structures will not withstand over-pressures of 0.6 atm without some sort of relief. The fan and the crack under the door obey simple formulae based on pressure differences, but these assumptions have limits.

- 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 opening 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 the local ratio of fuel and oxygen are appropriate.

16.3 Radiation

16.3.1 Radiation inside a box (radiation_in_a_box)

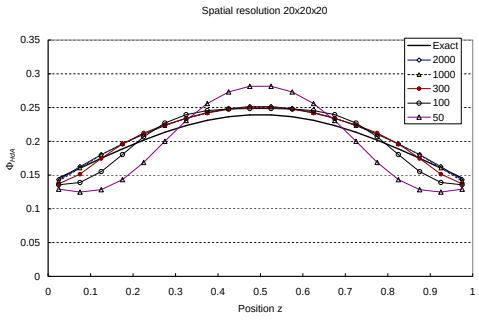
This verification case tests the computation of three-dimensional configuration factor Φ inside a cube box with one hot wall and five cold (0 K) walls. An overview of the test geometry is shown here:

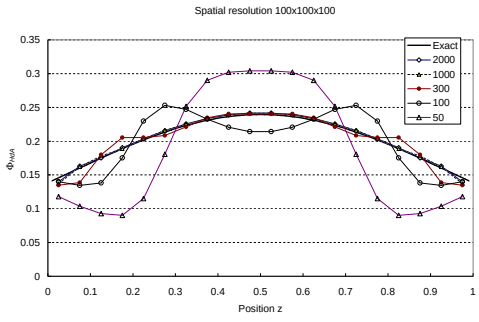


The configuration factors are calculated at the diagonal of the cold wall opposite to the hot wall. The exact values of the configuration factor from plane element dA to parallel rectangle H are calculated using the analytical solution [31]

(y,z)	Φ_{HdA}	(y,z)	Φ_{HdA}
0.025	0.1457	0.275	0.2135
0.075	0.1603	0.325	0.2233
0.125	0.1748	0.375	0.2311
0.175	0.1888	0.425	0.2364
0.225	0.2018	0.475	0.2391

Different variations of the case include the mesh resolution (20^3 and 100^3 cells) and the number of radiation angles (50, 100, 300, 1000, 2000). The exact and FDS results are shown here:





16.3.2 Radiation from a plane layer (radiation_plane_layer)

This case tests the computation of three-dimensional radiation from a homogenous, infinitely wide layer of radiating material. The temperature of the layer is 1273.15 K and absorption coefficients are varied. The thickness of the layer is fixed at 1.0 m, and the optical depth is 1.0κ. Wall temperatures are set to 0 K. The results are compared against the exact solution $S(\tau)$ presented in [32]

$$S(\tau) = S_b\,[1 - 2E_3(\tau)] \tag{16.6}$$

where $S_b = \sigma T^4$ is the black-body heat flux from the radiating plane and $E_3(\tau)$ is the exponential integral function (order 3) of optical depth τ.

The FDS results are computed at two mesh resolutions in the x-direction (I = 20 and I = 150). For I=20, both one-band and six-band versions are included to test the correct integration of heat fluxes over multiple bands. For I=20, 2D-versions are also computed (J=1). A special case with `KAPPA0` = 0 and an opposite wall temperature of 1273.15 K is computed to test the wall heat flux computation. The exact values and FDS predictions of the wall heat fluxes are given here:

τ	$S(\tau)$	FDS (I=20,J=20)		FDS (I=20,J=1)		FDS (I=150)
		1 band	6 bands	1 band	6 bands	1 band
0	148.9709	148.9709	148.4037	147.9426	147.3793	148.9709
0.01	2.8970	2.9180	2.9069	2.8364	2.8256	2.9258
0.1	24.9403	25.5501	25.4529	25.1078	25.0122	25.7045
0.5	82.9457	83.1309	82.8144	84.3719	84.0506	84.0264
1.0	116.2891	115.4051	114.9656	117.801	117.353	116.7751
10.	148.9698	148.9616	148.3947	148.9677	148.4005	148.9695

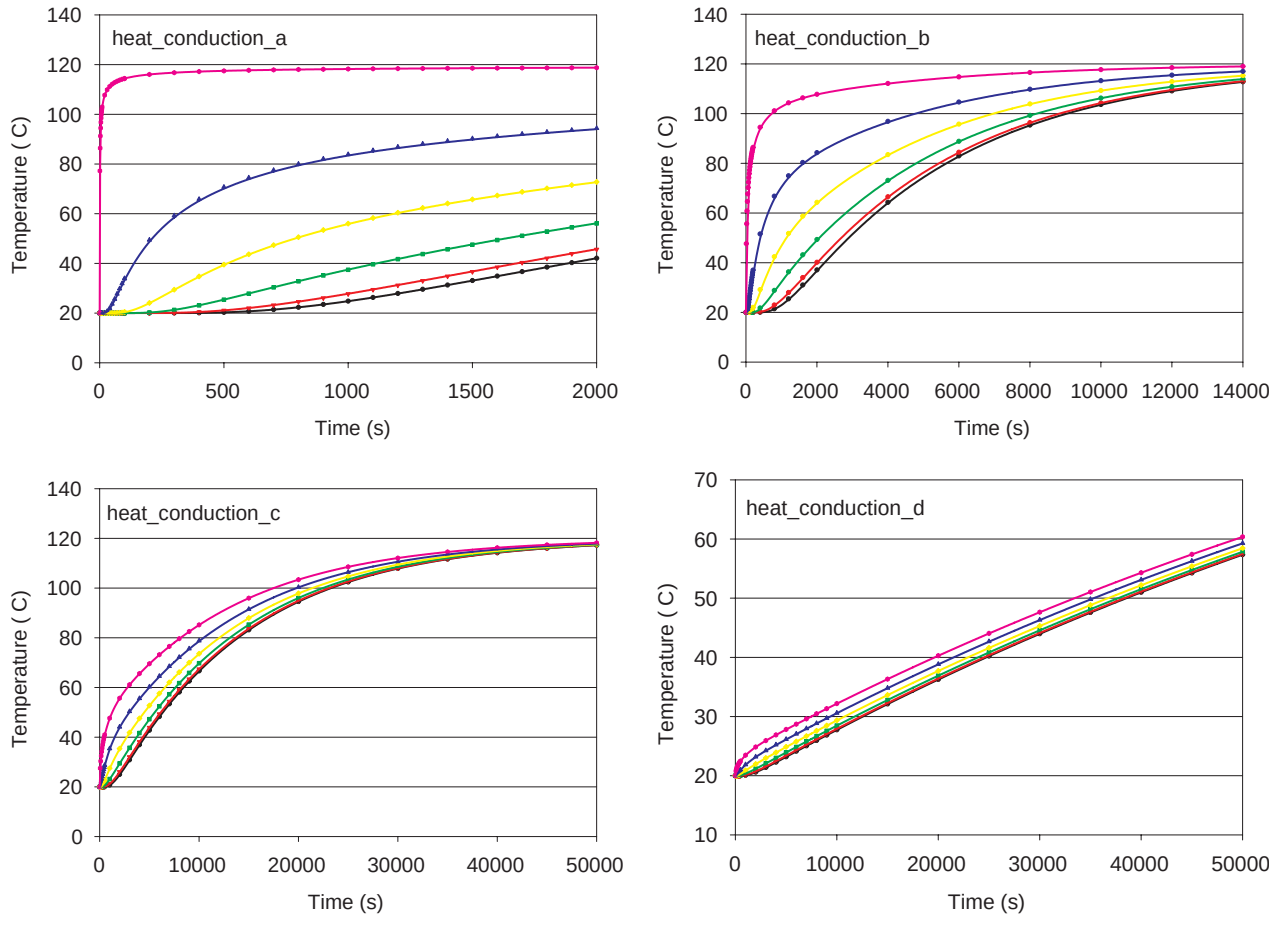
16.4 Solid Phase Phenomena

This section contains examples that test the one-dimensional heat conduction solver in FDS, along with those that include pyrolysis.

16.4.1 Simple Heat Conduction Through a Solid Slab (heat_conduction)

Analytical solutions of transient, one-dimensional heat conduction through a slab can be found in Refs. [33] and [34]. Four cases are examined here. In each, a slab of thickness $L = 0.1$ m is exposed on one face to an air temperature of $T_g = 120$ °C. The other face is insulated (adiabatic). The convective heat transfer from the gas to the slab is $\dot{q}''_c = h(T_g - T_s)$, where h is constant, and T_s is the slab face temperature. No thermal radiation is included.

Case	k (W/m/K)	ρ (kg/m^3)	c (kJ/kg/K)	h (W/m^2/K)	Bi hL/k
A	0.1	100	1	100	100
B	0.1	100	1	10	10
C	1.0	1000	1	10	1
D	10.0	10000	1	10	0.1



16.4.2 Temperature-Dependent Thermal Properties (heat_conduction_kc)

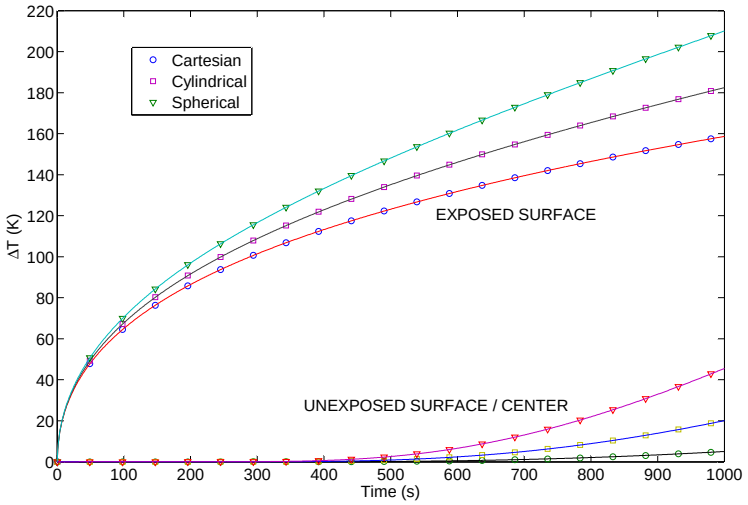
This example demonstrates the 1-D heat conduction in cartesian, cylindrical and spherical geometries with temperature-dependent thermal properties. The reference results were computed using HEATING (version 7.3), a multi-dimensional, finite-difference, general purpose heat transfer model [35]. In cartesian and cylindrical cases, the results have also been verified using a commercial finite-element solver, ABAQUS.

The sample of homogenous material is initially at 0 °C and at $t > 0$ exposed to a gas at 700 °C. A fixed heat transfer coefficient of 10 W/Km^2 is assumed. The density of the material is 10000 kg/m^3. The conductivity and specific heat are functions of temperature with the following values: $k(0) = 0.10$ W/m/K, $k(200) = 0.20$ W/m/K, $c(0) = 1.0$ kJ/kg/K, $c(100) = 1.2$ kJ/kg/K, $c(200) = 1.0$ kJ/kg/K. The thickness (radius) of the sample is 0.01 m. In the cartesian case, the back surface of the material is exposed to a gas at 0 °C. In the Figure below, the solid lines are FDS results and the open symbols are the HEATING results. An example input with cylindrical geometry looks like:

```
&MATL ID='MAT_1'
      EMISSIVITY = 0.0
      CONDUCTIVITY_RAMP='K_RAMP'
      SPECIFIC_HEAT_RAMP = 'C_RAMP'
      DENSITY=10000. /

&RAMP ID = 'K_RAMP' T=0,   F= 0.10 /
&RAMP ID = 'K_RAMP' T=100, F= 0.15 /
&RAMP ID = 'K_RAMP' T=200, F= 0.20 /
&RAMP ID = 'C_RAMP' T=0,   F= 1.00 /
&RAMP ID = 'C_RAMP' T=100, F= 1.20 /
&RAMP ID = 'C_RAMP' T=200, F= 1.00 /

&SURF ID='SLAB'
      STRETCH_FACTOR = 1.0
      GEOMETRY = 'CYLINDRICAL'
      MATL_ID='MAT_1'
      THICKNESS=0.01 /
```



16.4.3 A Simple Two-Step Pyrolysis Example (two_step_solid_reaction)

Consider the set of ordinary differential equations describing the mass fraction of three components of a solid material undergoing thermal degradation:

$$\begin{aligned} \frac{dY_a}{dt} &= -K_{ab}Y_a \\ \frac{dY_b}{dt} &= K_{ab}Y_a - K_{bc}Y_b \\ \frac{dY_c}{dt} &= K_{bc}Y_a \end{aligned} \tag{16.7}$$

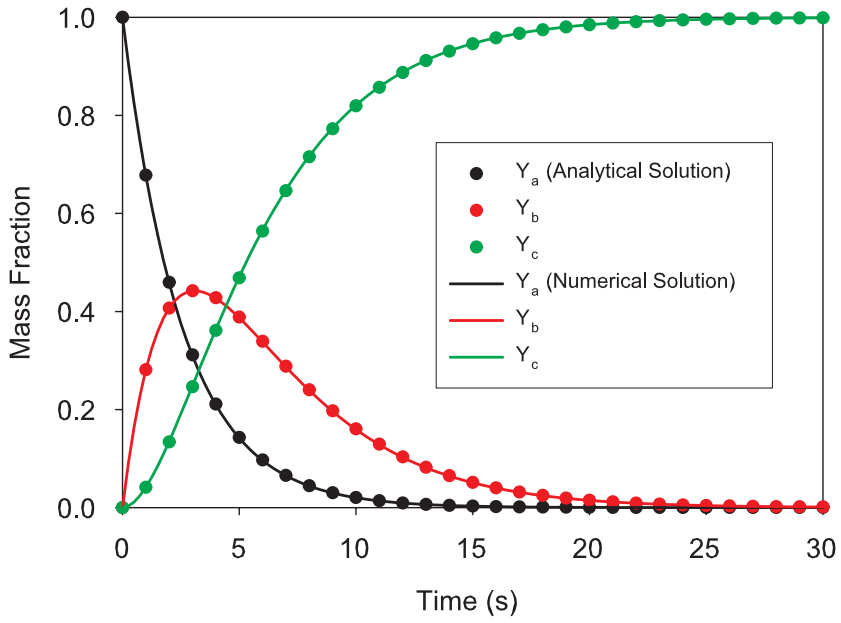
where the mass fraction of component a is 1 initially. The analytical solution is [**?**]

$$Y_a(t) = \exp(-K_{ab}t)$$

$$Y_b(t) = \frac{K_{ab}}{K_{bc}-K_{ab}}\exp(-K_{ab}t) - \exp(-K_{bc}t) \tag{16.8}$$

$$Y_c(t) = \left[K_{ab}(1-\exp(-K_{bc}t)) + K_{bc} * (\exp(-K_{ab}t)-1)\right]/(K_{ab}-K_{bc}) \tag{16.9}$$

The analytical and numerical solution for the parameters $K_{ab} = 0.389$ and $K_{bc} = 0.262$ are shown here:



16.4.4 Wall Internal Radiation (wall_internal_radiation)

The radiative flux inside the walls is computed using a two-flux model. In this case, the accuracy of the two-flux model is tested in the computation of emissive flux from a 0.10 m thick, homogenous layer of material at temperature of 1273.15 K and at ambient temperature of 10 K. The absorption coefficient is varied to cover a range [0.01, 10] of optical depths.

The exact solutions for radiative flux are the analytical solutions of plane layer emission [32]

$$S(\tau) = S_b\,[1 - 2E_3(\tau)] \tag{16.10}$$

where $S_b = \sigma T^4$ is the black-body heat flux from the radiating plane and $E_3(\tau)$ is the exponential integral function (order 3) of optical depth τ. The exact solutions and FDS results are shown below

τ	$S(\tau)(\mathrm{kW/m^2})$	FDS $(\mathrm{kW/m^2})$
0.01	2.897	2.950
0.1	24.94	26.98
0.5	82.95	93.90
1.0	116.3	128.4
10.	149.0	149.0

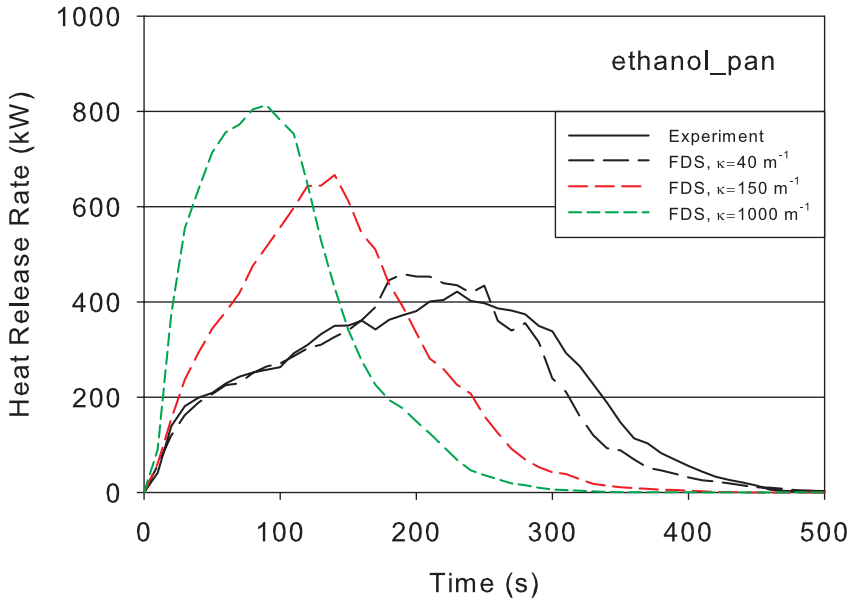
16.4.5 A Liquid Pool Fire (ethanol_pan)

In this example, a steel pan (0.7 m by 0.8 m) is filled with a thin layer (about 5 L, 9 mm) of ethanol, which burns out within about 10 min. This case tests a number of features – burning liquids, multiple layers of solids/liquids, and, most importantly, the absorption coefficient of the liquid. The pyrolysis models in FDS prior to version 5 assumed that radiative feedback from the fire and hot gases within a compartment were absorbed at the surface. In reality, this energy is absorbed in depth; the extent of which is characterized by the absorption coefficient, κ. This is a property of the liquid, as well as the gaseous vapors. FDS now uses an absorption coefficient for both the gas and solid/liquid phases. Here are the input lines that describe the properties of ethanol, and the pan in which it lies:

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

&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. /
```

The results of three calculations are shown below, each identical except for the value of the ABSORPTION_COEFFICIENT. The results of a single experiment are also shown, courtesy of Ian Thomas, Victoria University, Australia [36].



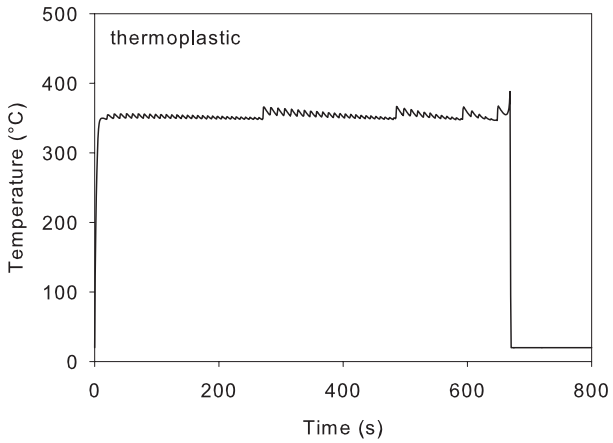
16.4.6 A Thermoplastic (thermoplastic)

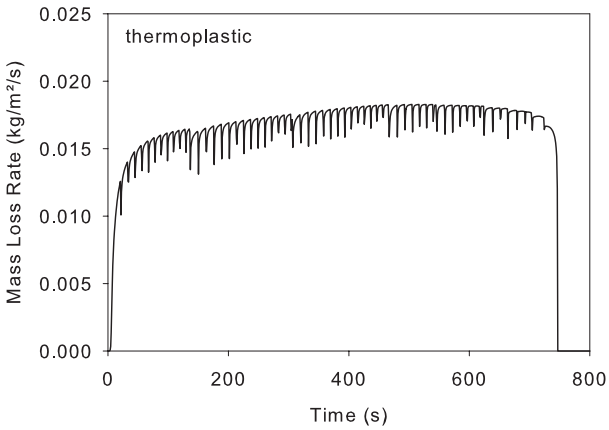
This example just exercises the solid phase algorithm in FDS. Essentially, the gas phase is shut off except for the imposition of a 50 kW/m^2 "virtual" heat flux. The solid in this example is a slab of plastic, similar in composition to PMMA.

```
&MATL ID='PMMA'
      CONDUCTIVITY=0.25
      SPECIFIC_HEAT=1.0
      DENSITY=500.
      N_REACTIONS=1
      NU_FUEL=1.
      HEAT_OF_REACTION=1578.
      HEAT_OF_COMBUSTION=25200.
      REFERENCE_TEMPERATURE=330. /

&SURF ID='PMMA SLAB'
      COLOR='BLACK'
      MATL_ID='PMMA'
      THICKNESS=0.025
      EXTERNAL_FLUX=50. /  External Flux is ONLY for this simple demo exercise
```

The material undergoes only one reaction – the conversion of solid to gaseous fuel vapor. In this case, the HEAT_OF_REACTION is essentially the latent heat of vaporization. The REFERENCE_TEMPERATURE indicates that the reaction is to occur at a (default) rate of 0.1 s^{-1} at 330 °C. The HEAT_OF_COMBUSTION refers to the combustion of the gaseous fuel vapor, which does not occur in this example.





The figures above show that both the temperature and burning rate of the thermoplastic are more or less constant over the burning phase. The slab "burns away" after about 10 min.

16.4.7 A Charring Solid (charring_solid)

This example just exercises the solid phase algorithm in FDS. Essentially, the gas phase is shut off except for the imposition of a 50 kW/m^2 "virtual" heat flux. The reaction mechanism is fairly complicated, as it includes various solids like cellulose, char, and ash. Each are input via `MATL` lines as follows:

```
&SURF ID='SPRUCE'
      STRETCH_FACTOR = 1.
      CELL_SIZE_FACTOR = 0.5
      MATL_ID(1,1:3) = 'CELLULOSE','WATER','LIGNIN'
      MATL_MASS_FRACTION(1,1:3) = 0.70,0.1,0.20
      MATL_ID(2,1) = 'CASI'
      THICKNESS(1:2)  =  0.01,0.01
      EXTERNAL_FLUX = 50. /

&MATL ID               = 'CELLULOSE'
      CONDUCTIVITY_RAMP = 'k_cell'
      SPECIFIC_HEAT    = 2.3
      DENSITY          = 400.
      N_REACTIONS      = 1
      A                = 2.8E19
      E                = 2.424E5
      HEAT_OF_REACTION = 0.
      NU_RESIDUE       = 1.0
      RESIDUE          = 'ACTIVE'/

&MATL ID                    = 'ACTIVE'
      EMISSIVITY            = 1.0
      CONDUCTIVITY_RAMP     = 'k_cell'
      SPECIFIC_HEAT         = 2.3
      DENSITY               = 400.
      N_REACTIONS           = 2
      A(1:2)                = 1.3E10,  3.23E14
      E(1:2)                = 1.505E5, 1.965E5
      HEAT_OF_REACTION(1:2) = 418.,    418.
      NU_RESIDUE(1:2)       = 0.35,    0.0
      NU_FUEL(1:2)          = 0.65,    1.0
      RESIDUE(1)            = 'CHAR' /

&MATL ID               = 'WATER'
      EMISSIVITY       = 1.0
      DENSITY          = 1000.
      CONDUCTIVITY     = 0.6
      SPECIFIC_HEAT    = 4.19
      N_REACTIONS      = 1
      A                = 1E20
      E                = 1.62E+05
      NU_WATER         = 1.0
      HEAT_OF_REACTION = 2260. /

&MATL ID                = 'CASI'
      CONDUCTIVITY_RAMP = 'k_CASI'
      DENSITY           = 200.
      SPECIFIC_HEAT     = 1.0 /

&MATL ID               = 'LIGNIN'
      EMISSIVITY       = 1.0
      DENSITY          = 550.
      CONDUCTIVITY     = 0.1
      SPECIFIC_HEAT    = 1.1 /

&MATL ID               = 'CHAR'
      EMISSIVITY       = 1.0
      DENSITY          = 140.
      CONDUCTIVITY_RAMP = 'k_char'
      SPECIFIC_HEAT    = 1.1 /
```

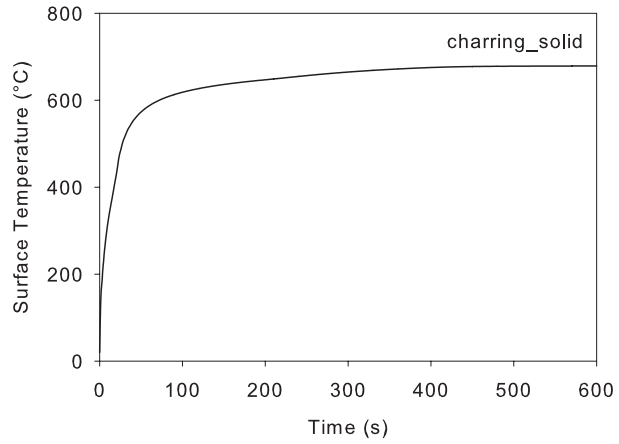
```
&RAMP ID='k_cell', T= 20., F=0.15 /
&RAMP ID='k_cell', T=500., F=0.29 /

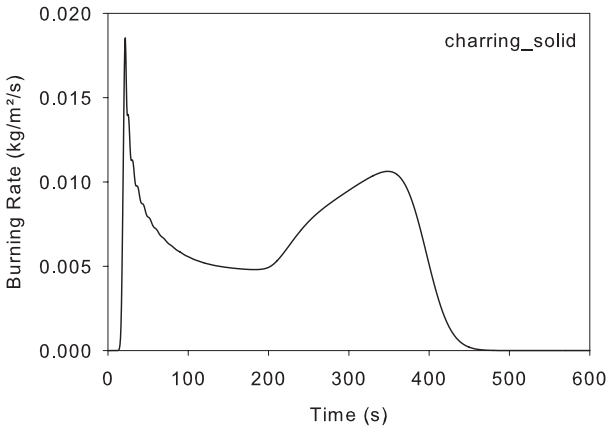
&RAMP ID='k_char', T= 20., F=0.08 /
&RAMP ID='k_char', T=900., F=0.25 /

&RAMP ID='k_CASI', T= 20., F=0.06 /
&RAMP ID='k_CASI', T=400., F=0.25 /
```

Note the edition of the parameter `EXTERNAL_FLUX` on the `SURF` line. This produces a 50 kW/m^2 flux on the sample, without any additional input lines in the file. It is just to test the solid phase model and should not be copied into an actual fire simulation.

The figures below show the surface temperature and burning rate of the wood under the 50 kW/m^2 external heat flux. The burning rate peaks at the start of the simulation, decreases throughout the burning phase, and then peaks again at the end due to presence of an external backing material. The initial peak is typical of char-forming solids.





16.4.8 Testing the "Burn-Away" Feature (box_burn_away)

This is a silly example of a solid block of "foam" that is ignited and burns until it is completely consumed. 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 integrated heat release 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. Its heat of combustion is 20000 kJ/kg. The ignitor is a 10 kW burner placed beside the block. The integrated heat release rate for a 120 s calculation ought to be:

$$(0.4)^3 \text{ m}^3 \times 20 \text{ kg/m}^3 \times 20000 \text{ kJ/kg} + 10 \text{ kW} \times 120 \text{ s} = 26.8 \text{ MJ} \tag{16.11}$$

16.4.9 A Couch Fire (couch)

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
      NU_FUEL               = 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
      NU_FUEL               = 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.

16.4.10 Flame Spread along a Cable Tray (cable_tray)

A common combustible in industrial settings are power, control, and instrument cables. The cables may be bundled in a variety of conduits; the most common of which is a ladder-like "tray." From the point of view of FDS, the pile of cables in a tray is a composite of a variety of plastics, insulators, and metal, usually copper. Here is one way to describe a tray of cables:

```
&MATL ID                    = 'PLASTIC'
      CONDUCTIVITY          = 0.2
      SPECIFIC_HEAT         = 1.5
      DENSITY               = 1500.
      N_REACTIONS           = 1
      HEAT_OF_REACTION      = 3000.
      HEAT_OF_COMBUSTION    = 25000.
      REFERENCE_TEMPERATURE = 400.
      NU_FUEL               = 1.0 /

&MATL ID            = 'COPPER'
      SPECIFIC_HEAT = 0.38
      CONDUCTIVITY  = 387.
      DENSITY       = 8940.   /

&SURF ID                        = 'Loose Cable'
      COLOR                     = 'IVORY BLACK'
      MATL_ID(1,1:2)            = 'PLASTIC','COPPER'
      MATL_MASS_FRACTION(1,1:2) = 0.4,0.6
      BACKING                   = 'EXPOSED'
      THICKNESS                 = 0.02 /

&OBST XB=-2.00, 2.00,-0.14, 0.14, 0.51, 0.55, SURF_ID='Loose Cable' /

&OBST XB=-2.00, 2.00,-0.15,-0.15, 0.50, 0.60, SURF_ID='SHEET METAL' / Tray Side
&OBST XB=-2.00, 2.00, 0.15, 0.15, 0.50, 0.60, SURF_ID='SHEET METAL' / Tray Side
&OBST XB=-1.95,-1.90,-0.15, 0.15, 0.50, 0.50, SURF_ID='SHEET METAL' / Rung
&OBST XB=-1.60,-1.55,-0.15, 0.15, 0.50, 0.50, SURF_ID='SHEET METAL' / Rung
...
&OBST XB= 1.90, 1.95,-0.15, 0.15, 0.50, 0.50, SURF_ID='SHEET METAL' / Rung
```

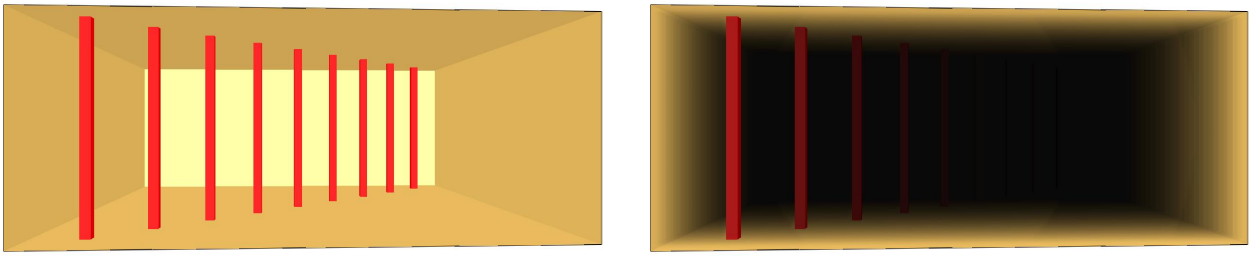
The pile of cables is assumed to be a solid slab, 28 cm wide and 2 cm deep. The tray is slightly wider and deeper, and because it is listed second in the input file, its surface properties take precedence wherever the cable slab and tray coincide. The mesh cells in this example are 5 cm on a side, but the heat transfer within the cable slab are governed by the 2 cm THICKNESS. The slab is 60 % copper, by mass. Note that we are not assuming multiple layers in this example – the slab is a single layer composite of plastic and copper. The plastic burns at about 400 °, but the copper remains. Thus, the cable does not "burn away."

The point of this test case is merely to propose a simple model of flame spread along a tray of assorted cable. Detailed thermo-physical property data for industrial-grade cable is usually not available, and even if it were, it would probably not improve upon the given model. The properties given in this example are almost completely fabricated. What is important here are the HEAT_OF_REACTION and REFERENCE_TEMPERATURE, obtained in most cases by a bench-scale measurement device like the cone calorimeter.

16.5 Detectors

16.5.1 Aspiration Detector (beam_detector)

A 10 m x 10 m x 4 m compartment is filled with 0.006 kg/kg of `MIXTURE_FRACTION_2` with the default soot yield 0.01 kg/kg. This results in an initial soot density of 71.9 mg/m^3 which using the default extinction coefficient of 8700 m^2/kg results in an optical depth of 0.626 m^{-1}. The compartment has a series of obstructions placed at varying depths that are multiples of 1 m. Using the correlation for the output quantity `visibility`, one obtains a visibility distance of 4.8 m. When viewing the smoke levels with Smokeview, one can just barely see the fifth obstacle which is at a distance of 5 m. Smokeview, therefore, is properly displaying the obscuration of the initial soot density. 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.



16.5.2 Aspiration Detector (aspiration_detector)

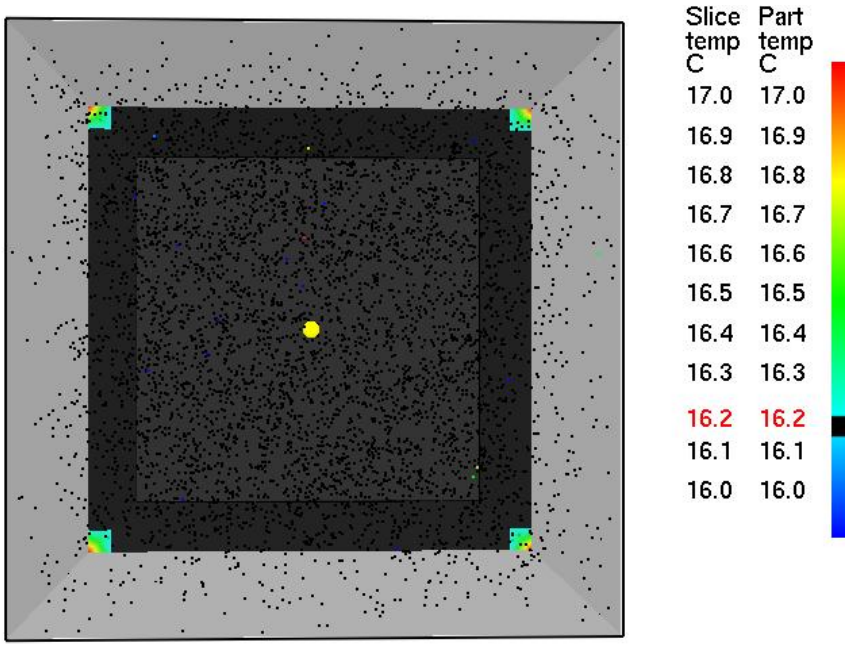
A cubical compartment, 2 m on a side with a fire has a three sampling location aspiration system. The three locations have flow rates of 0.1, 0.5, and 0.8 kg/s, respectively, and transport times of 0.2, 0.1, and 0.3 s, respectively. No bypass flow rate is specified for the aspiration detector. The input file fixes the initial time step to 0.01 s so that the initial output times in the **aspiration_detector_devc.csv** file will line up exactly with the transport times. At 0.75 s, when FDS begins reducing the time step below 0.01 s, the time delayed soot densities at the three sampling locations are $7.4x10^{-6}$ kg/m^3, $9.5x10^{-4}$ kg/m^3, and $1.6x10^{-18}$ kg/m^3, respectively. Using these values along with the respective flow rates results in a detector obscuration of 0.000823 %/m which is the same obscuration as predicted by FDS.

16.6 Droplets and Sprays

This section considers cases involving evaporating droplets, both water and fuel.

16.6.1 Water Droplet Evaporation (water_evaporation)

The case called **water_evaporation** is nothing more than stationary water droplets in an adiabatic box with dimensions of 1 m on a side. The air within the box is stirred to maintain uniform conditions, and there are no leaks or heat losses. The initial air temperature is 40 °C. Initially, the droplets have a median volumetric diameter of 100 μm, a temperature of 90 °C, and a total mass of 0.02 kg. It is expected that a steady-state will be achieved after several minutes. The initial energy content, sum of the air and water enthalpies, of the box is 360,000 kJ. After a short period of time, 0.0141 kg of water evaporate and the box reaches an equilibrium temperature of 16.2 °C, see the figure below. At this point the energy content of the box is 372,000 kJ or a 3 % error. At 16.2 °C, the expected evaporation is 0.0142 kg.



16.6.2 A Liquid Fuel Spray Burner (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.data**), 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:

```
&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. /

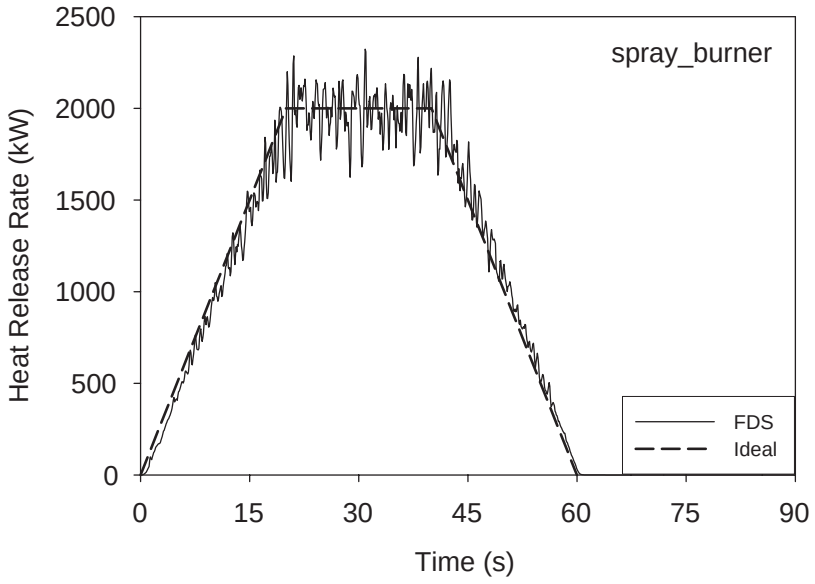
&PART ID='heptane droplets', FUEL=.TRUE., VAPORIZATION_TEMPERATURE=98.,
      HEAT_OF_VAPORIZATION=316., SPECIFIC_HEAT=2.25, DENSITY=688.,
      QUANTITIES(1:2)='DIAMETER','DROPLET_TEMPERATURE',
      DROPLETS_PER_SECOND=2000, DIAMETER=1000., HEAT_OF_COMBUSTION=44500.,
      DT_INSERT=0.02, SAMPLING_FACTOR=1 /

&PROP ID='nozzle', CLASS='NOZZLE', PART_ID='heptane droplets',
      FLOW_RATE=1.96, FLOW_RAMP='fuel', DROPLET_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 and the units are given in the User's Guide [37]. Note that a 2 MW fire is achieved via 2 nozzles flowing heptane at 1.96 L/min each:

$$2 \times 1.96 \, \frac{\mathrm{L}}{\mathrm{min}} \times \frac{1}{60} \, \frac{\mathrm{min}}{\mathrm{s}} \times 688 \, \frac{\mathrm{kg}}{\mathrm{m}^3} \times \frac{1}{1000} \, \frac{\mathrm{m}^3}{\mathrm{L}} \times 44500 \, \frac{\mathrm{kJ}}{\mathrm{kg}} = 2000 \, \mathrm{kW} \tag{16.12}$$

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 here:



16.6.3 Measuring Water Flux (bucket_test)

A common way of measuring the spray distribution for any fire sprinkler is called a "bucket test." Usually, one or several sprinklers is mounted a specified distance above an array of catch bins, the water flows for a given period of time, and the water flux distribution is calculated from the accumulated water mass in each bin. In the test case **bucket_test**, a single sprinkler is mounted 10 cm below a 5 m ceiling. Water flows for 5 s at a constant rate of 60 L/min. The simulation continues for another 5 s to allow water drops time to reach the floor. The total mass of water discharged is

$$60 \frac{\mathrm{L}}{\mathrm{min}} \times 1 \frac{\mathrm{kg}}{\mathrm{L}} \times \frac{1}{60} \frac{\mathrm{min}}{\mathrm{s}} \times 5 \,\mathrm{s} = 5 \,\mathrm{kg} \tag{16.13}$$

In the simulation, the boundary quantity `water_drops_AMPUA` (Accumulated Mass Per Unit Area) records the total water mass per unit area (kg/m^2), analogous to actual buckets the size of a grid cell. Summing the values of `water_drops_AMPUA` over the entire floor yields 4.96 kg. Where is the missing water? Some droplets evaporate, and some droplets fly beyond the computational domain. Also, there remain a small number of suspended drops at the end of the simulation. Note that there are no actual "buckets" in the simulation.

The accumulated water mass at the floor is extracted from the boundary (`BNDF`) file via the command line program **fds2ascii**. Here is a transcript of the session used to convert the binary FDS output file into ASCII format:

```
>> 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, water_drops_AMPUA
  Enter starting and ending time for averaging (s)
9 10
  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
```

16.6.4 Complex Spray Patterns (bucket_test_2)

The test case from the prior section is modified to create two jets of water.

```
&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' /

&TABL ID='TABLE1',TABLE_DATA=30,31,0,1,5,0.2/
&TABL ID='TABLE1',TABLE_DATA=30,31,179,180,5,0.8/
```

The jets are separated by 180 degrees. The jet in the `-x` direction is given one quarter the flow rate of the jet in the `+x` direction.

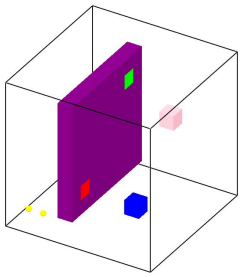
Viewing the particles in Smokeview shows two distinct jets of droplets in opposite directions. Following the post processing instructions above, you can observe that the `+x` jet does have four times the flow of the `-x` jet.

16.7 General Functionality

This section contains a variety of simple tests to check the functionality of the code. These examples are good demonstrations of how to make things happen in FDS.

16.7.1 Creating and Removing HOLEs and OBSTructions (create_remove)

It is often convenient to create or remove solid obstructions, or conversely, remove or create empty holes. They are essentially the same thing as far as the logic in FDS is concerned, but the input can be tricky. To avoid confusion, here is an example of holes and obstructions being created and removed.



```
&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=0.1,0.1,0.1, ID='timer 1', SETPOINT= 1.0, QUANTITY='TIME', INITIAL_STATE=.FALSE./
&DEVC XYZ=0.2,0.1,0.1, ID='timer 2', SETPOINT= 2.0, QUANTITY='TIME', INITIAL_STATE=.TRUE. /
&DEVC XYZ=0.1,0.1,0.1, ID='timer 3', SETPOINT= 3.0, QUANTITY='TIME', INITIAL_STATE=.FALSE./
&DEVC XYZ=0.2,0.1,0.1, ID='timer 4', SETPOINT= 4.0, QUANTITY='TIME', INITIAL_STATE=.TRUE./
```

Chapter 17

Sensitivity Analysis

A sensitivity analysis considers the extent to which uncertainty in model inputs influences model output. Model parameters can be the physical properties of solids and gases, boundary conditions, initial conditions, *etc.* The parameters can also be purely numerical, like the size of the numerical grid. FDS typically requires the user to provide several dozen different types of input parameters that describe the geometry, materials, combustion phenomena, *etc.* By design, the user is not expected to provide numerical parameters besides the grid size, although the optional numerical parameters are described in both the Technical Reference Guide and the User's Guide.

FDS does not limit the range of most of the input parameters because applications often push beyond the range for which the model has been validated. FDS is still used for research at NIST and elsewhere, and the developers do not presume to know in all cases what the acceptable range of any parameter is. Plus, FDS solves the fundamental conservation equations and is much less susceptible to errors resulting from input parameters that stray beyond the limits of simpler empirical models. However, the user is warned that he/she is responsible for the prescription of all parameters. The FDS manuals can only provide guidance.

The grid size is the most important numerical parameter in the model, as it dictates the spatial and temporal accuracy of the discretized partial differential equations. The heat release rate is the most important physical parameter, as it is the source term in the energy equation. Property data, like the thermal conductivity, density, heat of vaporization, heat capacity, *etc.*, ought to be assessed in terms of their influence on the heat release rate. Validation studies have shown that FDS predicts well the transport of heat and smoke when the HRR is prescribed. In such cases, minor changes in the properties of bounding surfaces do not have a significant impact on the results. However, when the HRR is not prescribed, but rather predicted by the model using the thermophysical properties of the fuels, the model output is sensitive to even minor changes in these properties.

The sensitivity analyses described in this chapter are all performed in basically the same way. For a given scenario, best estimates of all the relevant physical and numerical parameters are made, and a "baseline" simulation is performed. Then, one by one, parameters are varied by a given percentage, and the changes in predicted results are recorded. This is the simplest form of sensitivity analysis. More sophisticated techniques that involve the simultaneous variation of several parameters are impractical with a CFD model because the computation time is too long and the number of parameters too large to perform the necessary number of calculations to generate decent statistics.

17.1 Grid Sensitivity

The most important decision made by a model user is the size of the numerical grid. In general, the finer the numerical grid, the better the numerical solution of the equations. FDS is second-order accurate in space and

time, meaning that halving the grid cell size will decrease the discretization error in the governing equations by a factor of 4. Because of the non-linearity of the equations, the decrease in discretization error does not necessarily translate into a comparable decrease in the error of a given FDS output quantity. To find out what effect a finer grid has on the solution, model users usually perform some form of grid sensitivity study in which the numerical grid is systematically refined until the output quantities do not change appreciably with each refinement. Of course, with each halving of the grid cell size, the time required for the simulation increases by a factor of $2^4 = 16$ (a factor of two for each spatial coordinate, plus time). In the end, a compromise is struck between model accuracy and computer capacity.

Some grid sensitivity studies have been documented and published. Since FDS was first publicly released in 2000, significant changes in the combustion and radiation routines have been incorporated into the model. However, the basic transport algorithm is the same, as is the critical importance of grid sensitivity. In compiling sensitivity studies, only those that examined the sensitivity of routines no longer used have been excluded.

As part of a project to evaluate the use of FDS version 1 for large scale mechanically ventilated enclosures, Friday [38] performed a sensitivity analysis to find the approximate calculation time based on varying grid sizes. A propylene fire with a nominal heat release rate was modeled in FDS. There was no mechanical ventilation and the fire was assumed to grow as a function of the time from ignition squared. The compartment was a 3 m by 3 m by 6.1 m space. Temperatures were sampled 12 cm below the ceiling. Four grid sizes were chosen for the analysis: 30 cm, 15 cm, 10 cm, 7.5 cm. Temperature estimates were not found to change dramatically with different grid dimensions.

Using FDS version 1, Bounagui *et al.* [39] studied the effect of grid size on simulation results to determine the nominal grid size for future work. A propane burner 0.1 m by 0.1 m was modeled with a heat release rate of 1500 kW. A similar analysis was performed using Alpert's ceiling jet correlation [40] that also showed better predictions with smaller grid sizes. In a related study, Bounagui *et al.* [41] used FDS to evaluate the emergency ventilation strategies in the Louis-Hippolyte-La Fontaine Tunnel in Montreal, Canada.

Xin [42] used FDS to model a methane fueled square burner (1 m by 1 m) in the open. Engineering correlations for plume centerline temperature and velocity profiles were compared with model predictions to assess the influence of the numerical grid and the size of the computational domain. The results showed that FDS is sensitive to grid size effects, especially in the region near the fuel surface, and domain size effects when the domain width is less than twice the plume width. FDS uses a constant pressure assumption at open boundaries. This assumption will affect the plume behavior if the boundary of the computational domain is too close to the plume.

Ierardi and Barnett [43] used FDS version 3 to model a 0.3 m square methane diffusion burner with heat release rate values in the range of 14.4 kW to 57.5 kW. The physical domain used was 0.6 m by 0.6 m with uniform grid spacings of 15, 10, 7.5, 5, 3, 1.5 cm for all three coordinate directions. For both fire sizes, a grid spacing of 1.5 cm was found to provide the best agreement when compared to McCaffrey's centerline plume temperature and velocity correlations [44]. Two similar scenarios that form the basis for Alpert's ceiling jet correlation were also modeled with FDS. The first scenario was a 1 m by 1 m, 670 kW ethanol fire under a 7 m high unconfined ceiling. The planar dimensions of the computational domain were 14 m by 14 m. Four uniform grid spacings of 50, 33.3, 25, and 20 cm were used in the modeling. The best agreement for maximum ceiling jet temperature was with the 33.3 cm grid spacing. The best agreement for maximum ceiling jet velocity was for the 50 cm grid spacing. The second scenario was a 0.6 m by 0.6 m 1000 kW ethanol fire under a 7.2 m high unconfined ceiling. The planar dimensions of the computational domain were 14.4 m by 14.4 m. Three uniform grid spacings of 60, 30, and 20 cm were used in the modeling. The results show that the 60 cm grid spacing exhibits the best agreement with the correlations for both maximum ceiling jet temperature and velocity on a qualitative basis.

Petterson [45] also completed work assessing the optimal grid size for FDS version 2. The FDS model

predictions of varying grid sizes were compared to two separate fire experiments: The University of Canterbury McLeans Island Tests and the US Navy Hangar Tests in Hawaii. The first set of tests utilized a room with approximate dimensions of 2.4 m by 3.6 m by 2.4 m and fire sizes of 55 kW and 110 kW. The Navy Hangar tests were performed in a hangar measuring 98 m by 74 m by 15 m in height and had fires in the range of 5.5 MW to 6.6 MW. The results of this study indicate that FDS simulations with grids of 0.15 m had temperature predictions as accurate as models with grids as small as 0.10 m. Each of these grid sizes produced results within 15 % of the University of Canterbury temperature measurements. The 0.30 m grid produced less accurate results. For the comparison of the Navy Hangar tests, grid sizes ranging from 0.60 m to 1.80 m yielded results of comparable accuracy.

Musser *et al.* [46] investigated the use of FDS for course grid modeling of non-fire and fire scenarios. Determining the appropriate grid size was found to be especially important with respect to heat transfer at heated surfaces. The convective heat transfer from the heated surfaces was most accurate when the near surface grid cells were smaller than the depth of the thermal boundary layer. However, a finer grid size produced better results at the expense of computational time. Accurate contaminant dispersal modeling required a significantly finer grid. The results of her study indicate that non-fire simulations can be completed more quickly than fire simulations because the time step is not limited by the large flow speeds in a fire plume.

17.2 Sensitivity of Large Eddy Simulation Parameters

FDS uses the Smagorinsky form of the Large Eddy Simulation (LES) technique. This means that instead of using the actual fluid viscosity, the model uses a viscosity of the form

$$\mu_{\rm LES} = \rho \, (C_s \Delta)^2 \, |S|^{\frac{1}{2}} \tag{17.1}$$

where C_s is an empirical constant, Δ is a length on the order of the size of a grid cell, and the deformation term $|S|$ is related to the Dissipation Function, given by Eq. (**??**). Related to the "turbulent viscosity" are comparable expressions for the thermal conductivity and material diffusivity:

$$k_{\rm LES} = \frac{\mu_{\rm LES} \, c_p}{\rm Pr} \quad ; \quad (\rho D)_{\rm LES} = \frac{\mu_{\rm LES}}{\rm Sc} \tag{17.2}$$

The Prandtl number Pr and the Schmidt number Sc are likewise considered to be "turbulent" values. Thus, C_s, Pr and Sc are a set of empirical constants. Most FDS users simply use the default values of (0.2,0.5,0.5), but some have explored their effect on the solution of the equations.

In an effort to validate FDS with some simple room temperature data, Zhang *et al.* [47] tried different combinations of the Smagorinsky parameters, and suggested the current default values. Of the three parameters, the Smagorinsky constant C_s is the most sensitive. Smagorinsky [48] originally proposed a value of 0.23, but researchers over the past three decades have used values ranging from 0.1 to 0.23. There are also refinements of the original Smagorinsky model [49, 50, 51] that do not require the user to prescribe the constants, but rather generate them automatically as part of the numerical scheme.

17.3 Sensitivity of Radiation Parameters

Radiative heat transfer is included in FDS via the solution of the radiation transport equation for a non-scattering 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). There are several limitations of the model. First, the absorption coefficient for

the smoke-laden gas is a complex function of its composition and temperature. Because of the simplified combustion model, the chemical composition of the smokey gases, especially the soot content, can effect both the absorption and emission of thermal radiation. Second, the radiation transport is discretized via approximately 100 solid angles. For targets far away from a localized source of radiation, like a growing fire, the discretization can lead to a non-uniform distribution of the radiant energy. This can be seen in the visualization of surface temperatures, where "hot spots" show the effect of the finite number of solid angles. The problem can be lessened by the inclusion of more solid angles, but at a price of longer computing times. In most cases, the radiative flux to far-field targets is not as important as those in the near-field, where coverage by the default number of angles is much better.

Hostikka *et al.* examined the sensitivity of the radiation solver to changes in the assumed soot production, number of spectral bands, number of control angles, and flame temperature. Some of the more interesting findings were:

- Changing the soot yield from 1 % to 2 % increased the radiative flux from a simulated methane burner about 15 %
- Lowering the soot yield to zero decreased the radiative flux about 20 %.
- Increasing the number of control angles by a factor of 3 was necessary to ensure the accuracy of the model at the discrete measurement locations.
- Changing the number of spectral bands from 6 to 10 did not have a strong effect on the results.
- Errors of 100 % in heat flux were caused by errors of 20 % in absolute temperature.

The sensitivity to flame temperature and soot composition are consistent with combustion theory, which states that the source term of the radiative transport equation is a function of the absorption coefficient multiplied by the absolute temperature raised to the fourth power. The number of control angles and spectral bands are user-controlled numerical parameters whose sensitivities ought to be checked for each new scenario. The default values in FDS are appropriate for most large scale fire scenarios, but may need to be refined for more detailed simulations such as a low-sooting methane burner.

17.4 Sensitivity of Thermophysical Properties of Solid Fuels

An extensive amount of verification and validation work with FDS version 4 has been performed by Hietaniemi, Hostikka, and Vaari at VTT, Finland [52]. The case studies are comprised of fire experiments ranging in scale from the cone calorimeter (ISO 5660-1) to full-scale fire tests such as the room corner test (ISO 9705). Comparisons are also made between FDS results and data obtained in the SBI (Single Burning Item) Euro-classification test apparatus (EN 13823) as well as data obtained in two *ad hoc* experimental configurations: one is similar to the room corner test but has only partial linings and the other is a space to study fires in building cavities.

All of the case studies involve real materials whose properties must be prescribed so as to conform to the assumption in FDS that solids are of uniform composition backed by a material that is either cold or totally insulating. Sensitivity of the various physical properties and the boundary conditions were tested. Some of the findings were:

- The measured burning rates of various materials often fell between two FDS predictions in which cold or insulated backings were assumed for the solid surfaces. FDS lacks a multi-layer solid model.
- The ignition time of upholstery is sensitive to the thermal properties of the fabric covering, but the steady burning rate is sensitive to the properties of the underlying foam.

- Moisture content of wooden fuels is very important and difficult to measure.
- Flame spread over complicated objects, like cables laid out in trays, can be modeled if the surface area of the simplified object is comparable to that of the real object. This suggests sensitivity not only to physical properties, but also geometry. It is difficult to quantify the extent of the geometrical sensitivity.

There is little quantification of the observed sensitivities in the study. Fire growth curves can be linear to exponential in form, and small changes in fuel properties can lead to order of magnitude changes in heat release rate for unconfined fires. The subject is discussed in the FDS Validation Guide [**?**] where it is noted in many of the studies that predicting fire growth is difficult.

Recently, Lautenberger, Rein and Fernandez-Pello [53] developed a method to automate the process of estimating material properties to input into FDS. The methodology involves simulating a bench-scale test with the model and iterating via a "genetic" algorithm to obtain an optimal set of material properties for that particular item. Such techniques are necessary because most bench-scale apparatus do not provide a complete set of thermal properties.

17.5 Summary

The basis of large eddy simulation is that accuracy increases as the numerical mesh is refined. For fire applications, the grid sensitivity studies have shown that the accuracy of the model is a function of the characteristic fire diameter D^* divided by the grid cell size. It is not enough to describe the resolution of the calculation solely in terms of the grid cell size, but rather the grid cell size relative to the heat release rate. For non-fire applications, there are no simple means to evaluate “good resolution.”

As a rule of thumb, in simulations of limited resolution FDS predictions are more reliable in the far-field because the substantial numerical diffusion mimics the unresolved sub-grid scale mixing. This is hard to quantify other than through comparisons with experiment. In some of the sensitivity studies discussed above, the authors conclude that the model works best with a cell size of a given value, and often this cell is not the smallest one tested. In these cases, the authors have found a flow scenario where the unresolved convective mixing is almost exactly offset by numerical diffusion. This is fortuitous, but the conclusion does not necessarily extend to other scenarios. The disadvantage of any turbulence model, large eddy simulation included, is that good results are not guaranteed on grids of limited resolution. The advantage of LES over other turbulence models is that the solution of the actual governing equations, not a temporal or spatial average, is obtained as the mesh is refined.

The same can be said for phenomena closer in to the fire. However, grid resolution is more critical for near-field phenomena because numerical diffusion near the fire on coarse grids does not have the same fortuitous effect as it does on far-field results. In general, coarse resolution will decrease temperatures and velocities by smearing the values over the large grid cells. This can affect the radiative flux, convection to surrounding solids, and ultimately flame spread and fire growth.

Part IV

Working with the FDS Source Code

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 the web site (http://fds-sv.sourceforge.net).

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 **fds5.exe** is the compiled single processor program for a Windows-based PC; thus PC users do not need a Fortran compiler and do not need to compile the source code. For machines for which an executable has not been compiled, you must compile the code. Fortran 90/95 and C 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 mainly of Fortran 90 statements organized into about 25 files, plus an extra file containing some additional C routines needed for output to Smokeview. All of the C code is contained within the file called **isob.c**.

- Be aware that different compilers handle the names of C subroutines differently. Some compilers append an underscore to the names of the C routines called by the Fortran code. If the compiler produces an error involving the names of routines that are not recognized, invoke the C compiler pre-processing directive `pp_noappend` to stop the compiler from appending the underscore to the names of the C routines.

- 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 **main.f90**

- The parallel version of FDS uses **main_mpi.f90** instead of **main.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
isob.c	C Routine for computing isosurfaces and 3D smoke
prec.f90	Specification of numerical precision
smvv.f90	Interfaces for C routines used for Smokeview output
devc.f90	Derived type definitions and constants for devices'
type.f90	Derived type definitions
mesh.f90	Arrays and constants associated with each mesh
cons.f90	Global arrays and constants
func.f90	Global functions and subroutines
irad.f90	Functions needed for radiation solver, including RadCal
ieva.f90	Support routines for evac.f90
evac.f90	Egress computations (future capability)
pois.f90	Poisson (pressure) solver
radi.f90	Radiation solver
part.f90	Lagrangian particle transport and sprinkler activation
ctrl.f90	Definitions and routines for control functions
dump.f90	Output data dumps into files
read.f90	Read input parameters
mass.f90	Mass equation(s) and thermal boundary conditions
wall.f90	Wall boundary conditions
fire.f90	Combustion routines
pres.f90	Spatial discretization of pressure (Poisson) equation
divg.f90	Compute the flow divergence
init.f90	Initialize variables and Poisson solver
velo.f90	Momentum equations
main.f90 **or** main_mpi.f90	Main programs, serial and parallel versions

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 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 (°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.

Smokeview can display the Plot3D data. In addition, the Plot3D data sets can be read into some other graphics programs that accept the data format. This particular format is very convenient, and recognized by a number of graphics packages, including AVS, IRIS Explorer and Tecplot [1].

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

[1] With the exception of Smokeview, the graphics packages referred to in this document are not included with the source code, but are commercially available.

```
N_DEVC
FDS Time , ID(1)     , ID(2)     , ... , ID(N_DEVC)
TIME , QUANTITY(1) , QUANTITY(2) , ... , QUANTITY(N_DEVC)
s    , UNITS(1)    , UNITS(2)    , ... , UNITS(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 `I`th device, `QUANTITY(I)` is the physical quantity represented, `UNITS(I)` the units, `T(J)` the time of the `J`th dump, and `VAL(I,J)` the value at the `I`th device at the `J`th 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

```
N_CTRL
FDS Time , ID(1)     , ID(2)     , ... , ID(N_CTRL)
TIME ,   ,   , ... ,
s    , status    , status    , ... , status(N_CTRL)
T(1) , VAL(1,1)    , VAL(2,1)    , ... , VAL(N_CTRL,1)
T(2) , VAL(1,2)    , VAL(2,2)    , ... , VAL(N_CTRL,2)
                       .
                       .
                       .
```

where `N_CTRL` is the number of control functions, `ID(I)` is the user-defined ID of the `I`th control function, and `VAL(I,J)` the state, `-1 = .FALSE. and +1 = .TRUE.`, of the `I`th control function at the `J`th 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.

19.6 Mixture Fraction State Relations

The functional dependence of the mass fraction of the reactants and products of combustion on the mixture fraction is reported in the comma delimited file **CHID_state_II.csv**. The file consists of nominally 10 columns, the first column containing the mixture fraction, the last column the average molecular weight, and the rest the mass fractions of the various gases. Where `II` represents the chemical reaction for which the state relationships represent. For the two parameter model these are `01` for the complete reaction (formation

of combustion products) and `02` for the null reaction (extinction). For the three parameter model these are `01`, `02`, and `03` for the incomplete (CO production) reaction, complete (CO_2 production) reaction, and the null reaction respectively.

19.7 Slice Files

The slice files defined under the namelist group `SLCF` are named **CHID_*n*.sf** (*n*=01,02...), 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=11,I2),J=J1,J2),K=K1,K2)
        .
        .
        .
WRITE(LUSF) TIME
WRITE(LUSF) (((QQ(I,J,K),I=11,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 12.4 for more details about the program **fds2ascii**.

19.8 Boundary Files

The boundary files defined under the namelist group `BNDF` are named **CHID_*n*.bf** (*n*=01,02...), 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
WRITE(LUBF) I1,I2,J1,J2,K1,K2,IOR
        .
        .
        .
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 length 30. 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. 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

The tracer particles and sprinkler droplets coordinates and related quantities are stored in 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          ! The number ONE as a 4 byte real
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
```

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
WRITE(LUPF) NPLIM       ! Number of particles to write out for this time step
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)
```

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...), 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.

Bibliography

[1] K.B. McGrattan, S. Hostikka, J.E. Floyd, H.R. Baum, and R.G. Rehm. Fire Dynamics Simulator (Version 5), Technical Reference Guide. NIST Special Publication 1018-5, National Institute of Standards and Technology, Gaithersburg, Maryland, October 2007. i, 3, 33, 68, 72, 77, 81

[2] G.P. Forney. User's Guide for Smokeview Version 5 - A Tool for Visualizing Fire Dynamics Simulation Data. NIST Special Publication 1017-1, National Institute of Standards and Technology, Gaithersburg, Maryland, August 2007. i, 3, 7

[3] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI – Portable Parallel Programming with the Message-Passing Interface*. MIT Press, Cambridge, Massachusetts, 2 edition, 1999. 10

[4] Verification and Validation of Selected Fire Models for Nuclear Power Plant Applications. NUREG 1824, United States Nuclear Regulatory Commission, Washington, DC, 2007. 30

[5] R.C. Reid, J.M. Prausnitz, and B.E. Poling. *Properties of Gases and Liquids*. McGraw-Hill, New York, 4th edition, 1987. 70

[6] P.J. DiNenno, editor. *SFPE Handbook of Fire Protection Engineering*. National Fire Protection Association, Quincy, Massachusetts, 3rd edition, 2002. 83

[7] Pamela P. Walatka and Pieter G. Buning. PLOT3D User's Manual, version 3.5. NASA Technical Memorandum 101067, NASA, 1989. 97

[8] G.W. Mulholland. *SFPE Handbook of Fire Protection Engineering*, chapter Smoke Production and Properties. National Fire Protection Association, Quincy, Massachusetts, 3rd edition, 2002. 98, 99

[9] M.L. Janssens and H.C. Tran. Data Reduction of Room Tests for Zone Model Validation. *Journal of Fire Science*, 10:528–555, 1992. 99

[10] 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. 100

[11] 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. 100

[12] U. Wickstrom, 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. 101

[13] American Society for Testing and Materials, West Conshohocken, Pennsylvania. *ASTM E 1355-04, Standard Guide for Evaluating the Predictive Capabilities of Deterministic Fire Models*, 2004. 131, 132

[14] W. Mell, K.B. McGrattan, and H. Baum. Numerical Simulation of Combustion in Fire Plumes. In *Twenty-Sixth Symposium (International) on Combustion*, pages 1523–1530. Combustion Institute, Pittsburgh, Pennsylvania, 1996. 131

[15] K.B. McGrattan, H.R. Baum, and R.G. Rehm. Large Eddy Simulations of Smoke Movement. *Fire Safety Journal*, 30:161–178, 1998. 131

[16] H.R. Baum, R.G. Rehm, P.D. Barnett, and D.M. Corley. Finite Difference Calculations of Buoyant Convection in an Enclosure, Part I: The Basic Algorithm. *SIAM Journal of Scientific and Statistical Computing*, 4(1):117–135, March 1983. 131

[17] H.R. Baum and R.G. Rehm. Finite Difference Solutions for Internal Waves in Enclosures. *SIAM Journal of Scientific and Statistical Computing*, 5(4):958–977, December 1984. 131

[18] H.R. Baum and R.G. Rehm. Calculations of Three Dimensional Buoyant Plumes in Enclosures. *Combustion Science and Technology*, 40:55–77, 1984. 131

[19] R.G. Rehm, P.D. Barnett, H.R. Baum, and D.M. Corley. Finite Difference Calculations of Buoyant Convection in an Enclosure: Verification of the Nonlinear Algorithm. *Applied Numerical Mathematics*, 1:515–529, 1985. 131

[20] J.C. Adams, W.S. Brainerd, J.T. Martin, B.T. Smith, and J.L. Wagener. *Fortran 95 Handbook: Complete ISO/ANSI Reference*. MIT Press, Cambridge, Massachusetts, 1997. 132

[21] K.B. McGrattan, T. Kashiwagi, H.R. Baum, and S.L. Olson. Effects of Ignition and Wind on the Transition to Flame Spread in a Microgravity Environment. *Combustion and Flame*, 106:377–391, 1996. 134

[22] T. Kashiwagi, K.B. McGrattan, S.L. Olson, O. Fujita, M. Kikuchi, and K. Ito. Effects of Slow Wind on Localized Radiative Ignition and Transition to Flame Spread in Microgravity. In *Twenty-Sixth Symposium (International) on Combustion*, pages 1345–1352. Combustion Institute, Pittsburgh, Pennsylvania, 1996. 134

[23] W. Mell and T. Kashiwagi. Dimensional Effects on the Transition from Ignition to Flame Spread in Microgravity. In *Twenty-Seventh Symposium (International) on Combustion*, pages 2635–2641. Combustion Institute, Pittsburgh, Pennsylvania, 1998. 134

[24] W. Mell, S.L. Olson, and T. Kashiwagi. Flame Spread Along Free Edges of Thermally-Thin Samples in Microgravity. In *Twenty-Eighth Symposium (International) on Combustion*, pages 2843–2849. Combustion Institute, Pittsburgh, Pennsylvania, 2000. 134

[25] K. Prasad, Y. Nakamura, S.L. Olson, O. Fujita, K. Nishizawa, K. Ito, and T. Kashiwagi. Effect of Wind Velocity on Flame Spread in Microgravity. In *Twenty-Ninth Symposium (International) on Combustion*, pages 2553–2560. Combustion Institute, Pittsburgh, Pennsylvania, 2002. 134

[26] Y. Nakamura, T. Kashiwagi, K.B. McGrattan, and H.R. Baum. Enclosure Effects on Flame Spread over Solid Fuels in Microgravity. *Combustion and Flame*, 130:307–321, 2002. 134

[27] W.E. Mell, K.B. McGrattan, and H.R. Baum. g-Jitter Effects on Spherical Diffusion Flames. *Microgravity Science and Technology*, 15(4):12–30, 2004. 134

[28] A. Mukhopadhyay and I.K. Puri. An Assessment of Stretch Effects on Flame Tip Using the Thin Flame and Thick Formulations. *Combustion and Flame*, 133:499–502, 2003. 134

[29] A. Hamins, M. Bundy, I.K. Puri, K.B. McGrattan, and W.C. Park. Suppression of Low Strain Rate Non-Premixed Flames by an Agent. In *Proceedings of the 6th International Microgravity Combustion Workshop, NASA/CP-2001-210826*, pages 101–104. National Aeronautics and Space Administration, Lewis Research Center, Cleveland, Ohio, May 2001. 134

[30] K.B. McGrattan, R.G. Rehm, and H.R. Baum. Fire-Driven Flows in Enclosures. *Journal of Computational Physics*, 110(2):285–291, 1994. 134

[31] R. Siegel and J. R. Howell. *Thermal Radiation Heat Transfer*. Taylor & Francis, New York, 4th edition, 2002. 142

[32] Y.B. Zel'dovich and Y.P. Raizer. *Physics of shock waves and high-temperature hydrodynamic phenomena*. Dover Publications, New York, 2002. Translated from the Russian and then edited by W.D.Hayes and R.F. Probstein. 143, 147

[33] D. Drysdale. *An Introduction to Fire Dynamics*. John Wiley and Sons, New York, 2nd edition, 2002. 144

[34] H.S. Carslaw and J.C. Jaegar. *Conduction of Heat in Solids*. Oxford University Press, 2nd edition, 1959. 144

[35] K.W. Childs. HEATING 7: Multidimensional, Finite-Difference Heat Conduction Analysis Code System. Technical Report PSR-199, Oak Ridge National Laboratory, Oak Ridge, TN, 1998. 145

[36] I.R. Thomas, K.A.M. Moinuddin, and I.D. Bennetts. The Effect of Fuel Quantity and Location on Small Enclosure Fires. *Journal of Fire Protection Engineering*, 17(2):85–102, May 2007. 148

[37] K.B. McGrattan, B.W. Klein, S. Hostikka, and J.E. Floyd. Fire Dynamics Simulator (Version 5), User's Guide. NIST Special Publication 1019-5, National Institute of Standards and Technology, Gaithersburg, Maryland, October 2007. 158

[38] P. Friday and F. W. Mowrer. Comparison of FDS Model Predictions with FM/SNL Fire Test Data. NIST GCR 01-810, National Institute of Standards and Technology, Gaithersburg, Maryland, April 2001. 164

[39] A. Bounagui, N. Benichou, C. McCartney, and A. Kashef. Optimizing the Grid Size Used in CFD Simulations to Evaluate Fire Safety in Houses. In *3rd NRC Symposium on Computational Fluid Dynamics, High Performance Computing and Virtual Reality*, pages 1–8, Ottawa, Ontario, Canada, December 2003. National Research Council, Canada. 164

[40] R.L. Alpert. *SFPE Handbook of Fire Protection Engineering*, chapter Ceiling Jet Flows. National Fire Protection Association, Quincy, Massachusetts, 3rd edition, 2003. 164

[41] A. Bounagui, A. Kashef, and N. Benichou. Simulation of the Dynamics of the Fire for a Section of the L.H.-La Fontaine Tunnel. IRC-RR- 140, National Research Council Canada, Ottawa, Canada, K1A0R, September 2003. 164

[42] Y. Xin. Assessment of Fire Dynamics Simulation for Engineering Applications: Grid and Domain Size Effects. In *Proceedings of the Fire Suppression and Detection Research Application Symposium, Orlando, Florida*. National Fire Protection Association, Quincy, Massachusetts, 2004. 164

[43] J.A. Ierardi and J.R. Barnett. A Quantititive Method for Calibrating CFD Model Calculations. In *Proceedings of the CIB-CTBUH International Conference on Tall Buildings*, pages 507–514. International Council for Research and Innovation in Building and Construction (CIB), 2003. 164

[44] G. Heskestad. *SFPE Handbook of Fire Protection Engineering*, chapter Fire Plumes, Flame Height and Air Entrainment. National Fire Protection Association, Quincy, Massachusetts, 3rd edition, 2002. 164

[45] N.M. Petterson. Assessing the feasibility of reducing the grid resolution in fds field modeling. Fire Engineering Research Report 2002/6, University of Canterbury, Christchurch, New Zealand, March 2002. 164

[46] A. Musser, K. B. McGrattan, and J. Palmer. Evaluation of a Fast, Simplified Computational Fluid Dynamics Model for Solving Room Airflow Problems. NISTIR 6760, National Institute of Standards and Technology, Gaithersburg, Maryland, June 2001. 165

[47] W. Zhang, A. Hamer, M. Klassen, D. Carpenter, and R. Roby. Turbulence Statistics in a Fire Room Model by Large Eddy Simulation. *Fire Safety Journal*, 37:721–752, 2002. 165

[48] J. Smagorinsky. General Circulation Experiments with the Primitive Equations. I. The Basic Experiment. *Monthly Weather Review*, 91(3):99–164, March 1963. 165

[49] J.W. Deardorff. Numerical Investigation of Neutral and Unstable Planetary Boundary Layers. *Journal of Atmospheric Sciences*, 29:91–115, 1972. 165

[50] M. Germano, U. Piomelli, P. Moin, and W.H. Cabot. A Dynamic Subgrid-Scale Eddy Viscosity Model. *Physics of Fluids A*, 3(7):1760–1765, 1991. 165

[51] D.K. Lilly. A Proposed Modification of the Germano Subgrid-Scale Closure Method. *Physics of Fluids A*, 4(3):633–635, 1992. 165

[52] J. Hietaniemi, S. Hostikka, and J. Vaari. FDS Simulation of Fire Spread – Comparison of Model Results with Experimental Data. VTT Working Paper 4, VTT Building and Transport, Espoo, Finland, 2004. 166

[53] C. Lautenberger, G. Rein, and C. Fernandez-Pello. The application of a genetic algorithm to estimate the material properties for fire modeling from bench-scale fire test data. *Fire Safety Journal*, 41:204–214, 2006. 167

Index

2D Calculations, 25

atmospheric stratification, 47
Axially-Symmetric Calculations, 25

baroclinic torque, 32
BNDF, 96
  QUANTITY, 96
boundary conditions, 43, 44
boundary layer, 46

CLIP, 35
CO Production, 68
Colors, 62
Controls, 88
  Create and Remove Obstructions, 85
  Function List, 88
  Using DEVC, 85
CTRL, 88
  DELAY, 89
  FUNCTION_TYPE, 88
    ALL, 89
    ANY, 89
    AT_LEAST, 89
    CUSTOM, 90
    DEADBAND, 89
    KILL, 90
    ONLY, 89
    RESTART, 90
    TIME_DELAY, 89
  ID, 88
  INITIAL_STATE, 89
  INPUT_ID, 88
  LATCH, 88
  N, 89
  ON_BOUND, 89
  RAMP_ID, 90
  SETPOINT, 89

DEVC, 79
  BYPASS_FLOWRATE, 84
  DELAY, 84
  FLOWRATE, 84
  ID, 79
  IOR, 79
  ORIENTATION, 79
  PROP_ID, 79
  QUANTITY, 80
  XYZ, 79
Device
  Aspiration Detector, 84
  Beam Detector, 83
  Heat Detector, 82
  Smoke Detector, 83
  Spray Nozzle, 82
  Sprinkler, 80
Download, 7
dry-pipe sprinkler system, 91
DUMP, 93
  COLUMN_DUMP_LIMIT, 80

fan, 89
fds2ascii, 102
Features, 3
  Recent Changes, 4
Fire from Heat Release Rate, 44
fixed surface temperature, 45
Flame Extinction, 68

gravity, 32

HEAD, 23
  CHID, 23
  TITLE, 23
HOLE, 39
  COLOR, 39
  CTRL_ID, 39, 125
  DEVC_ID, 39
  RGB, 39

TRANSPARENCY, 39
XB, 39
HRRPUA, 44
HVAC, 33, 88–90

INIT, 35
initial solid temperature, 58
Input File
Overview, 19
ISOF, 96

Large Eddy Simulation, 33
Liquid Fuels, 56
louver, 46

Mass Flux, 45
Material layers, 52
MATL, 52
A, 53
ABSORPTION_COEFFICIENT, 53
BOILING_TEMPERATURE, 57
CONDUCTIVITY, 53
DENSITY, 53
E, 53
EMISSIVITY, 53
HEAT_OF_COMBUSTION, 55
HEAT_OF_REACTION, 55
N_S, 53
N_T, 54
NU_FUEL, 53
NU_RESIDUE, 53
NU_WATER, 53
REFERENCE_RATE, 54
REFERENCE_TEMPERATURE, 54
RESIDUE, 53
SPECIFIC_HEAT, 53
THRESHOLD_TEMPERATURE, 54
MB - Mesh Boundary, 40
MESH, 25
IJK, 25
SYNCHRONIZE, 27
XB, 25
mesh dimensions, 29
MISC, 31
RADIATION, 72
MLRPUA, 44
multi-mesh efficiency, 27
Multiple Meshes, 26

Namelist Group, 19
Namlist Groups Table, 22
Navier-Stokes, 3
non-planar geometries, 48

OBST, 37
CTRL_ID, 125
PERMIT_HOLE, 40
outdoor fires, 47
Output, 93
Adiabatic Surface Temperature, 101
Boundary File, 96
fds2ascii, 102
Integrated Measurements, 94
Isosurface File, 96
Plot3D, 96
Point Measurements, 94
Slice File, 96
Statistics, 95
Summary of Quantities, 103
Output Files
.out, 12
.stop, 12
_devc.csv, 80
_hrr.csv, 98

PART, 73
Poisson initialization, 15
POROUS_FLOOR
output issues, 101
pre-action sprinkler system, 91
pressure leakage, 49
PROF, 95
IOR, 95
QUANTITY, 95
XYZ, 95
PROP, 80
Pyrolysis models, 53

RADI, 71
ANGLE_INCRMENT, 72
CH4_BANDS, 72
NMIEANG, 72
NUMBER_RADIATION_ANGLES, 72
RADIATIVE_FRACTION, 71
RADTMP, 72
TIME_STEP_INCREMENT, 72
WIDE_BAND_MODEL, 72
radial fire spread, 47

RAMP, 59
REAC, 65
Restart, 31
Running FDS, 9
  Parallel, 10
    LAM-MPI, 11
    MPICH2, 11
  Serial, 9

Sawtooth, 38
SLCF, 95
Sloped Ceilings, 38
Solid fuels, 53
SPEC, 69
spinkler
  Spray Pattern, 81
Spray Pattern, 81
sprinkler, 80, 89
Sprinkler Suppression, 78
stack effect, 33, 35
Support, 13
  Error Statements, 14
    Inadequate Resources, 14
    Input File, 14
    Numerical Instability, 14
    Poisson, 15
    Run-Time, 15
  Issue Reporting, 15
  Version Number, 13
SURF, 44
  ADIABATIC, 44, 45
  BACKING, 45
  BURN_AWAY, 57
  CELL_SIZE_FACTOR, 58
  COLOR, 62
  CONVECTIVE_HEAT_FLUX, 45
  DUCT_PATH, 51
  E_COEFFICIENT, 78
  EMISSIVITY, 45
  EXTERNAL_FLUX, 64
  GEOMETRY, 48
  HEAT_OF_VAPORIZATION, 55
  HRRPUA, 47, 55
  IGNITION_TEMPERATURE, 55
  LEAK_PATH, 49
  MASS_FLUX, 46
  MASS_FLUX_TOTAL, 46
  MASS_FRACTION, 46, 60
  MATL_ID, 52
  MATL_MASS_FRACTION, 52
  MAX_PRESSURE, 50
  NPPC, 74
  PART_ID, 74
  PARTICLE_MASS_FLUX, 74
  PLE, 47
  POROUS, 51
  PROFILE, 47
  RAMP_MF, 60
  RAMP_Q, 47, 55, 59
  RAMP_T, 59
  RAMP_V, 59
  RGB, 62
  SHRINK, 58
  SLIP_FACTOR, 46
  STRETCH_FACTOR, 58
  TAU_MF, 60
  TEXTURE_HEIGHT, 62
  TEXTURE_MAP, 62
  TEXTURE_WIDTH, 62
  THICKNESS, 52
  THICNKNESS, 44
  TMP_BACK, 58
  TMP_FRONT, 45
  TMP_INNER, 58
  VEL, 45
  VOLUME_FLUX, 45
  Z0, 47
Surface Texture Maps, 62
System Requirements
  Hardware, 7
  MPI, 8
  Operating System, 8

TABL, 59
TAIL, 20
tangential velocity, 46
thermostat, 89
TIME, 23
  DT, 24
  SYNCHRONIZE, 24, 27
  T_BEGIN, 23
  T_END, 23
  WALL_INCREMENT, 58, 64
TRNX,TRNY,TRNZ, 25, 28
  CC, 29
  PC, 29

Troubleshooting
VENT Orientation, 41

VENT, 40
CTRL_ID, 125
Volume Flux, 45

ZONE, 49