Fire Dynamics Simulator (Version 5)
User’s Guide

Kevin McGrattan
Randall McDermott
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 29, 2010
FDS Version 5.5
SVNRepository Revision : 6997
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.
Preface

This Guide describes how to use the Fire Dynamics Simulator (FDS) version 5. Because new features are added periodically, check the current version number on the inside front jacket of this manual.

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

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

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

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

Randall McDermott joined the research staff of the Building and Fire Research Lab in 2008. He received a B.S. degree from the University of Tulsa in Chemical Engineering in 1994 and a doctorate at the University of Utah in 2005. His research interests include subgrid-scale models and numerical methods for large-eddy simulation, adaptive mesh refinement, immersed boundary methods, and Lagrangian particle methods.

Simo Hostikka is a Senior Research Scientist at VTT Technical Research Centre of Finland. He received a master of science (technology) degree in 1997 and a doctorate in 2008 from the Department of Engineering Physics and Mathematics of the Helsinki University of Technology. He is the principal developer of the radiation and solid phase sub-models within FDS.

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 the Fire Research Grants Program. He is the principal developer of the multi-parameter mixture fraction combustion model and control logic within FDS.
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 Vetteri, Dave Stroup, Steve Kerber and Nelson Bryner, who have used FDS and Smokeview as part of several investigations of fire fighter line of duty deaths. As part of these studies, they have provided valuable information on the model’s usability and accuracy when compared to large scale measurements made during fire reconstructions.

Bryan Klein of NIST assisted in adding cross-referencing functionality to this document, making it easier to view electronically.

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.

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

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

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

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

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

Prof. Ofodike Ezekoye and Craig Weinschenk of the University of Texas, Austin, helped to formulate and test the thermocouple model within FDS. This work was based on an earlier analysis by Stephen Welsh and Philip Rubini.

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution/Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nick Agnew</td>
<td>Maunsell, Australia</td>
</tr>
<tr>
<td>Camille Azzi</td>
<td>Universities of Glasgow and Strathclyde, Scotland</td>
</tr>
<tr>
<td>Matthew Bilson</td>
<td>Maunsell, Australia</td>
</tr>
<tr>
<td>George Braga</td>
<td>Federal District Fire Department, Brazil</td>
</tr>
<tr>
<td>Keith Calder</td>
<td>Senez Reed Calder Engineering, Canada</td>
</tr>
<tr>
<td>Steven Chi Heng Lam</td>
<td>Hoare Lea Fire Engineering, UK</td>
</tr>
<tr>
<td>Doo Chan Choi</td>
<td>Rolf Jensen &amp; Associates, Inc., USA</td>
</tr>
<tr>
<td>Marco Cigolini</td>
<td>Italferr spa, Italy</td>
</tr>
<tr>
<td>John Cutonilli</td>
<td>Hughes Associates, Inc., USA</td>
</tr>
<tr>
<td>Sylvain Desanghere</td>
<td>CTICM (Centre Technique Industriel de la Construction Métallique), France</td>
</tr>
<tr>
<td>Montu L. Das</td>
<td>Gage-Babcock &amp; Associates, USA and Canada</td>
</tr>
<tr>
<td>Franck Didieux</td>
<td>Laboratoire National de Métrologie et d’Essais (LNE), France</td>
</tr>
<tr>
<td>Johannes Dimyadi</td>
<td>AstraVision-Solutions, New Zealand</td>
</tr>
<tr>
<td>Bill Ferrante</td>
<td>Roosevelt Fire District, USA</td>
</tr>
<tr>
<td>Paul Fuss</td>
<td>NIST, USA</td>
</tr>
<tr>
<td>Ralf Galster</td>
<td>Ing. Büro für Brandschutz Riesener, Germany</td>
</tr>
<tr>
<td>Andreas Gerndt</td>
<td>University of Louisiana, USA</td>
</tr>
<tr>
<td>Emanuele Gissi</td>
<td>Corpo Nazionale dei Vigili del Fuoco, Comando Prov. di Genova, Italy</td>
</tr>
<tr>
<td>Simon J. Ham</td>
<td>Fire Safety Engineering Consultants Ltd., UK</td>
</tr>
<tr>
<td>Paul Hart</td>
<td>Swiss Re, GAP Services, USA</td>
</tr>
<tr>
<td>Hsiao, Li Kai (Gary)</td>
<td>Fire Bureau, Taipei, Taiwan</td>
</tr>
<tr>
<td>Hu Zhi-Xin</td>
<td>University of Maryland, USA</td>
</tr>
<tr>
<td>Ilya N. Karkin</td>
<td>SITIS Ltd., Russia</td>
</tr>
<tr>
<td>Susanne Kilian</td>
<td>hhberlin, Fire Safety Engineers, Germany</td>
</tr>
<tr>
<td>Sung Chan Kim</td>
<td>School of Mechanical Engineering, Chung Ang University, Korea</td>
</tr>
<tr>
<td>Pierre-Louis Lamballais</td>
<td>Flashover-Backdraft, France</td>
</tr>
<tr>
<td>A. Leonardi</td>
<td>StIIL (Studio di Ingegneria Leonardi), Italy</td>
</tr>
<tr>
<td>Davy Leroy</td>
<td>Arup Fire, UK</td>
</tr>
<tr>
<td>Yunlong Liu</td>
<td>Parsons Brinckerhof, Australia</td>
</tr>
<tr>
<td>Timothy Liu</td>
<td>Locke Carey Fire Consultants, UK</td>
</tr>
<tr>
<td>Dave McGill</td>
<td>Seneca College, Ontario, Canada</td>
</tr>
<tr>
<td>Ken Miller</td>
<td>Las Vegas Fire &amp; Rescue, USA</td>
</tr>
<tr>
<td>Stephan B. Miller</td>
<td>Mr. 3D Computer Graphics and the University of Houston Downtown, USA</td>
</tr>
<tr>
<td>Pete Muir</td>
<td>Safe Consulting, UK</td>
</tr>
<tr>
<td>Stephen Olenick</td>
<td>Combustion Science &amp; Engineering, Inc., USA</td>
</tr>
<tr>
<td>Kristopher Overholt</td>
<td>University of Houston Downtown, USA</td>
</tr>
<tr>
<td>PENG Wei</td>
<td>State Key Laboratory of Fire Science, China</td>
</tr>
<tr>
<td>Andrew Purchase</td>
<td>Maunsell, Australia</td>
</tr>
<tr>
<td>Christian Rogsch</td>
<td>University of Wuppertal, Germany</td>
</tr>
<tr>
<td>Michael Roth</td>
<td>RWDI, Canada</td>
</tr>
<tr>
<td>Ahmed Salem</td>
<td>Alexandria University, Egypt</td>
</tr>
<tr>
<td>Robert Schmidt</td>
<td>Combustion Science &amp; Engineering, Inc., USA</td>
</tr>
<tr>
<td>Joe Skaggs</td>
<td>CASE Forensics, USA</td>
</tr>
<tr>
<td>Piotr Smardz</td>
<td>Ahearne Fire Engineering Consultants, Ireland</td>
</tr>
<tr>
<td>Name</td>
<td>Affiliation</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>Jamie Stern-Gottfried</td>
<td>Arup Fire, UK</td>
</tr>
<tr>
<td>Boris Stock</td>
<td>BFT Cognos GmbH, Germany</td>
</tr>
<tr>
<td>Blair Stratton</td>
<td>Beca, New Zealand</td>
</tr>
<tr>
<td>Csaba Szilagyi</td>
<td>Szolnok Fire Department, Hungary</td>
</tr>
<tr>
<td>Mahmood Tabaddor</td>
<td>Underwriters Laboratories, USA</td>
</tr>
<tr>
<td>Charlie Thornton</td>
<td>Thunderhead Engineering, USA</td>
</tr>
<tr>
<td>Sebastian Ukleja</td>
<td>University of Ulster, Northern Ireland</td>
</tr>
<tr>
<td>Giacomo Vili</td>
<td>Dipartimento Fisica Tecnica (DfT) UNIPd, Italy</td>
</tr>
<tr>
<td>Andreas Vischer</td>
<td>RWTH Aachen University, Germany</td>
</tr>
<tr>
<td>Karl Wallasch</td>
<td>Hoare Lea Fire Engineering, UK</td>
</tr>
<tr>
<td>Kaoru Wakatsuki</td>
<td>National Research Institute of Fire and Disaster, Japan</td>
</tr>
<tr>
<td>Joachim Wollstädt</td>
<td>Ing. Büro für Brandschutz Riesener, Germany</td>
</tr>
<tr>
<td>Yang Shan-You</td>
<td>State Key Laboratory of Fire Science, China</td>
</tr>
<tr>
<td>Matthias Zähringer</td>
<td>Ing. Büro für Brandschutz Riesener, Germany</td>
</tr>
<tr>
<td>Robin Zevotek</td>
<td>C&amp;S Engineers, Life Safety Services, Syracuse, New York, USA</td>
</tr>
<tr>
<td></td>
<td>GIDAI, University of Cantabria, Spain</td>
</tr>
</tbody>
</table>
# Contents

Preface .......................................................... i
Disclaimer ....................................................... iii
About the Authors ............................................... v
Acknowledgments .............................................. vii

I The Basics of 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 ................................................. 15
    4.1 The Version Number .................................... 15
    4.2 Common Error Statements .............................. 16
    4.3 Support Requests and Bug Tracking .................. 17
    4.4 Known Issues .......................................... 18

II Writing an FDS Input File .................................. 21
  5 The Basic Structure of an Input File ...................... 23
    5.1 Naming the Job .......................................... 23
    5.2 Namelist Formatting .................................... 23
    5.3 Input File Structure .................................... 24
8.3.2 Thermal Properties ................................................. 62
8.3.3 Back Side Boundary Conditions .................................. 63
8.3.4 Initial and Back Side Temperature ............................... 63
8.3.5 Walls with Different Materials Front and Back .............. 63
8.3.6 Special Topic: Non-Planar Walls and Targets ............... 64
8.3.7 Special Topic: Solid Phase Numerical Gridding Issues ...... 64

8.4 Pyrolysis Models ...................................................... 66
8.4.1 A Gas Burner with a Specified Heat Release Rate ........... 66
8.4.2 Special Topic: A Radially-Spreading Fire .................... 66
8.4.3 Solid Fuels that Burn at a Specified Rate .................... 67
8.4.4 Solid Fuels that do NOT Burn at a Specified Rate ......... 68
8.4.5 Liquid Fuels ....................................................... 74
8.4.6 Fuel Burnout ...................................................... 75

8.5 Testing Your Pyrolysis Model ...................................... 78

9 Ventilation ............................................................... 85
9.1 Simple Vents, Fans and Heaters ................................... 85
9.1.1 Supply and Exhaust Vents ....................................... 85
9.1.2 Heaters .......................................................... 86
9.1.3 Total Mass Flux .................................................. 86
9.1.4 Louvered Vents .................................................. 86
9.1.5 Jet Fans .......................................................... 86
9.2 Species and Species Mass Flux Boundary Conditions .......... 88
9.3 Special Topic: Pressure Boundary Conditions .................. 89
9.4 Special Topic: Fires and Flows in the Outdoors ............... 91
9.5 Special Topic: The HVAC Namelist Group ....................... 92
9.6 Tangential Velocity Boundary Conditions at Solid Surfaces .. 94
9.7 Pressure-Related Effects: The ZONE Namelist Group (Table 15.29) . 95
9.7.1 Specifying Pressure Zones ..................................... 95
9.7.2 Leaks .......................................................... 96
9.7.3 Fan Curves ...................................................... 98

10 User-Specified Functions, Ramps and Tables ...................... 103
10.1 Time-Dependent Functions ....................................... 103
10.2 Temperature-Dependent Functions ............................... 105
10.3 Tabular Functions ................................................... 105

11 Combustion and Radiation ........................................... 107
11.1 Mixture Fraction Combustion: The REAC Namelist Group .... 107
11.1.1 Basics ........................................................ 107
11.1.2 Special Topic: Heat of Combustion .......................... 110
11.1.3 Special Topic: Flame Extinction .............................. 110
11.1.4 Special Topic: CO Production ................................ 111
11.1.5 Special Topic: Turbulent Combustion ....................... 111
11.2 Extra Gas Species: The SPEC Namelist Group ............... 113
11.2.1 Basics ........................................................ 113
11.2.2 Special Topic: Gas Species Properties ..................... 114
11.2.3 Special Topic: Yields of Gaseous Species (NU_GAS) ...... 115
11.2.4 Special Topic: Finite-Rate Combustion ........................................ 116
11.3 Radiation Transport: The RADI Namelist Group ............................ 118

12 Particles and Droplets .............................................................. 119
  12.1 Basics ............................................................................. 119
  12.2 Particle and Droplet Insertion ........................................... 120
    12.2.1 Particles Introduced at a Solid Surface .......................... 120
    12.2.2 Droplets Introduced at a Sprinkler or Nozzle .............. 120
    12.2.3 Particles or Droplets Introduced within a Volume .......... 121
    12.2.4 Controlling the Number of Particles and Droplets ....... 121
  12.3 Particle and Droplet Properties .......................................... 121
    12.3.1 Thermal Properties .................................................. 122
    12.3.2 Size Distribution .................................................... 122
    12.3.3 Drag ....................................................................... 122
    12.3.4 Velocity on Solid Surfaces ....................................... 123
    12.3.5 Color ...................................................................... 123
  12.4 Special Types of Particles and Droplets .................................. 124
    12.4.1 Massless Particles .................................................... 124
    12.4.2 Static Particles or Droplets ...................................... 124
    12.4.3 Water Droplets ........................................................ 124
    12.4.4 Fuel Droplets .......................................................... 124
    12.4.5 Solid Particles that do not Evaporate ......................... 125
  12.5 Special Topic: Suppression by Water (Mixture Fraction Model Only) .... 126

13 Devices and Control Logic ......................................................... 129
  13.1 Device Location and Orientation: The DEVC Namelist Group (Table 15.4) .............................................. 129
  13.2 Device Output .................................................................. 130
  13.3 Special Devices and their Properties: The PROP Namelist Group (Table 15.19) ........................................... 131
    13.3.1 Sprinklers .............................................................. 131
    13.3.2 Nozzles .................................................................. 133
    13.3.3 Heat Detectors ........................................................ 134
    13.3.4 Smoke Detectors ..................................................... 135
    13.3.5 Beam Detection Systems ......................................... 136
    13.3.6 Aspiration Detection Systems ................................ 137
    13.3.7 Electrical Cable Failure .......................................... 138
  13.4 Basic Control Logic ............................................................ 141
    13.4.1 Creating and Removing Obstructions ......................... 141
    13.4.2 Activating and Deactivating Vents ......................... 142
  13.5 Advanced Control Functions: The CTRL Namelist Group ........ 143
    13.5.1 Control Functions: ANY, ALL, ONLY, and AT_LEAST ........ 143
    13.5.2 Control Function: TIME_DELAY ............................... 144
    13.5.3 Control Function: DEADBAND ................................. 144
    13.5.4 Control Function: RESTART and KILL ..................... 145
    13.5.5 Control Function: CUSTOM ....................................... 145
    13.5.6 Combining Control Functions: A Pre-Action Sprinkler System .... 146
    13.5.7 Combining Control Functions: A Dry Pipe Sprinkler System .... 146
    13.5.8 Example Case: activate_vents ................................... 147
  13.6 Controlling a RAMP ........................................................... 147
13.7 Visualizing FDS Devices Using Smokeview Objects ........................................... 148
  13.7.1 Static Smokeview Objects ................................................................. 148
  13.7.2 Dynamic Smokeview Objects - Customized Using &PROP Parameters .......... 150
  13.7.3 Dynamic Smokeview Objects - Customized Using &PROP Parameters and Particle File Data ................................................................. 152

14 Output Data ................................................................. 155
  14.1 Output Control Parameters: The DUMP Namelist Group ................................ 155
  14.2 Output File Types ................................................................. 157
    14.2.1 Device Output: The DEVC Namelist Group ........................................ 157
    14.2.2 Quantities within Solids: The PROF Namelist Group ......................... 158
    14.2.3 Animated Planar Slices: The SLCF Namelist Group ............................ 159
    14.2.4 Animated Boundary Quantities: The BNDF Namelist Group .................. 159
    14.2.5 Animated Isosurfaces: The ISOF Namelist Group ................................ 160
    14.2.6 Plot3D Static Data Dumps .............................................................. 160
    14.2.7 SMOKE3D: Realistic Smoke and Fire ................................................ 161
  14.3 Special Output Quantities ................................................................. 162
    14.3.1 Heat Release Rate ............................................................................. 162
    14.3.2 Visibility and Obscuration ............................................................... 163
    14.3.3 Layer Height and the Average Upper and Lower Layer Temperatures ....... 164
    14.3.4 Thermocouples ................................................................................. 164
    14.3.5 Heat Fluxes and Thermal Radiation .................................................. 165
    14.3.6 Droplet Output Quantities ................................................................. 165
    14.3.7 Interfacing with Structural Models ................................................... 167
    14.3.8 Useful Solid Phase Outputs .............................................................. 168
    14.3.9 Fractional Effective Dose (FED) ......................................................... 168
    14.3.10 Spatially-Integrated Outputs ............................................................ 169
    14.3.11 Temporally-Integrated Outputs ....................................................... 171
    14.3.12 Wind and the Pressure Coefficient .................................................. 172
    14.3.13 Dry Volume and Mass Fractions ...................................................... 172
    14.3.14 Gas Velocity .................................................................................... 172
    14.3.15 A Posteriori Mesh Quality Metrics .................................................. 172
  14.4 Extracting Numbers from the Output Data Files ............................................. 177
  14.5 Summary of Frequently-Used Output Quantities ........................................... 178
  14.6 Summary of Infrequently-Used Output Quantities ......................................... 182
  14.7 Summary of HVAC Output Quantities ......................................................... 183

15 Alphabetical List of Input Parameters ................................................................. 185
  15.1 BNDF (Boundary File Parameters) ............................................................ 186
  15.2 CLIP (MIN/MAX Clipping Parameters) ....................................................... 186
  15.3 CTRL (Control Function Parameters) ......................................................... 186
  15.4 DEVC (Device Parameters) ........................................................................ 187
  15.5 DUMP (Output Parameters) ....................................................................... 187
  15.6 HEAD (Header Parameters) ...................................................................... 188
  15.7 HOLE (Obstruction Cutout Parameters) .................................................... 188
  15.8 HVAC (HVAC System Definition) ............................................................... 189
  15.9 INIT (Initial Conditions) .......................................................................... 189
  15.10 ISOF (Isosurface Parameters) ................................................................... 190
15.11 MATL (Material Properties) ..................................................... 190
15.12 MESH (Mesh Parameters) ....................................................... 191
15.13 MISC (Miscellaneous Parameters) ......................................... 191
15.14 MULT (Multiplier Function Parameters) ................................. 193
15.15 OBST (Obstruction Parameters) .............................................. 193
15.16 PART (Lagrangian Particles/Droplets) ..................................... 194
15.17 PRES (Pressure Solver Parameters) ........................................ 195
15.18 PROF (Wall Profile Parameters) ............................................. 195
15.19 PROP (Device Properties) ..................................................... 195
15.20 RADI (Radiation Parameters) ................................................ 196
15.21 RAMP (Ramp Function Parameters) ....................................... 197
15.22 REAC (Reaction Parameters) ................................................ 197
15.23 SLCF (Slice File Parameters) ................................................ 198
15.24 SPEC (Species Parameters) .................................................. 198
15.25 SURF (Surface Properties) .................................................. 199
15.26 TABL (Table Parameters) .................................................... 201
15.27 TIME (Time Parameters) ..................................................... 201
15.28 TRNX, TRNY, TRNZ (MESH Transformations) ......................... 201
15.29 VENT (Vent Parameters) ..................................................... 201
15.30 ZONE (Pressure Zone Parameters) ......................................... 202

16 Conversion of Old Input Files to FDS 5 ....................................... 203
16.1 Numerical Domain Parameters: GRID and PDIM ....................... 203
16.2 Obstructions, Vents, and Holes: OBST, VENT, and HOLE ............ 203
16.3 Surface Parameters: SURF .................................................... 203
16.4 Reaction Parameters: REAC ................................................ 204
16.5 Device Parameters: SPRK, SMOD, HEAT, THCP ....................... 205

III FDS and Smokeview Development Tools .................................. 207

17 The FDS/Smokeview Repository .............................................. 209

18 Compiling FDS ........................................................................ 211
18.1 FDS Source Code ................................................................. 211

19 Output File Formats .................................................................. 213
19.1 Diagnostic Output ............................................................... 213
19.2 Heat Release Rate and Related Quantities ................................ 214
19.3 Device Output Data ............................................................. 214
19.4 Control Output Data ............................................................ 215
19.5 Gas Mass Data ................................................................. 215
19.6 Mixture Fraction State Relations ........................................... 215
19.7 Slice Files ........................................................................ 215
19.8 Plot3D Data ...................................................................... 216
19.9 Boundary Files .................................................................. 217
19.10 Particle Data ................................................................. 217
19.11 Profile Files .................................................................... 218
# List of Figures

6.1 An example of a multiple-mesh geometry. .................................................. 30  
6.2 Rules governing the alignment of meshes. ................................................. 32  
6.3 Piecewise-Linear Mesh Transformation. .................................................. 34  
6.4 Polynomial Mesh Transformation. .............................................................. 34  
6.5 Axi-symmetric helium plume ................................................................. 39  
6.6 Simple example of flow through a duct. .................................................... 43  
7.1 An example of the multiplier function. .................................................... 48  
7.2 Simple example of SAWTOOTH=.FALSE. ................................................... 49  
8.1 Simple demonstration of pyrolysis model. ............................................... 70  
8.2 A more complicated demonstration of the pyrolysis model. ....................... 72  
8.3 Input parameters for sample case pyrolysis_2. ....................................... 73  
8.4 Input parameters for sample case ethanol_pan. ........................................ 74  
8.5 Output of box_burn_away test cases. ..................................................... 77  
8.6 Output of thermoplastic test case. ......................................................... 80  
8.7 Output of charring_solid test case. .......................................................... 81  
8.8 Output of room_fire test case. ............................................................... 83  
9.1 Example of positive pressure at a tunnel entrance. .................................. 90  
9.2 Output of pressure_rise test case. ........................................................... 97  
9.3 Output of zone_break test case. .............................................................. 98  
9.4 Example of a fan curve. ............................................................................. 99  
9.5 Output of the fan_test example. ............................................................... 100  
9.6 Pressure rise in sealed compartment. ....................................................... 101  
11.1 Output of door_crack test case. .............................................................. 111  
11.2 Example of gas filling. ............................................................................. 114  
12.1 Example of water cascading over solid obstructions. ............................... 123  
12.2 HRR of spray burner. .............................................................................. 126  
13.1 Output of the flow rate test case. ............................................................. 135  
13.2 Example of a beam detector. .................................................................. 137  
13.3 Output of aspiration_detector test case. ............................................... 139  
13.4 Example of a vent controls. .................................................................... 147  
14.1 (Left) Resolved signal, MTR is small. (Right) Unresolved signal, MTR is close to unity. .................................................. 173  
14.2 Haar mother wavelet on the interval [0,1]. .............................................. 175  
14.3 Averages and coefficients for local Haar wavelet transforms on four typical signals. .................................................. 176
# List of Tables

4.1 FDS features with known issues or problems. ........................................... 19

5.1 Namelist Group Reference Table ............................................................... 26

7.1 Sample of Color Definitions (A complete list is included on the website)  ........... 55

10.1 Parameters used to control time-dependence. .......................................... 105

11.1 Optional Gas Species ................................................................................. 115

13.1 Suggested Values for Smoke Detector Model. ............................................ 136

13.2 Control function types for CTRL ............................................................... 143

13.3 Single Frame Static Objects .................................................................. 149

13.4 Dual Frame Static Objects ...................................................................... 149

13.4 Dual Frame Static Objects (continued) .................................................... 150

13.5 Dynamic Objects - Customized using SMOKEVIEW_PARAMETERS on a &PROP line 151

13.5 Dynamic Objects (continued) .................................................................. 152

13.6 Dynamic Objects - Customized using SMOKEVIEW_PARAMETERS on a &PROP line and Particle File Data ................................................................. 153

13.6 Dynamic Objects (continued) .................................................................. 154

14.1 Output quantities. ..................................................................................... 179

14.2 Output quantities. ..................................................................................... 182

14.3 Output quantities. ..................................................................................... 183

15.1 Boundary File Parameters ....................................................................... 186

15.2 MIN/MAX Clipping Parameters ............................................................... 186

15.3 Control Function Parameters .................................................................. 186

15.4 Device Parameters .................................................................................. 187

15.5 Output Parameters ................................................................................... 187

15.6 Header Parameters .................................................................................. 188

15.7 Obstruction Cutout Parameters .............................................................. 189

15.8 Initial Conditions .................................................................................... 189

15.9 Initial Conditions .................................................................................... 189

15.10 Isosurface Parameters ......................................................................... 190

15.11 Material Properties .............................................................................. 190

15.12 Mesh Parameters ................................................................................... 191

15.13 Miscellaneous Parameters .................................................................... 191

15.14 Multiplier Function Parameters ............................................................ 193

15.15 Obstruction Parameters ....................................................................... 193
Part I

The Basics of FDS
Chapter 1

Introduction

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

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 a separate user’s guide [2].

1.1 Features of FDS

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

Hydrodynamic Model FDS solves numerically a form of the Navier-Stokes equations appropriate for low-speed, thermally-driven flow with an emphasis on smoke and heat transport from fires. The core algorithm is an explicit predictor-corrector scheme, second order accurate in space and time. Turbulence is treated by means of the 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 13.1 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 13.1 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 13.5 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 9.7 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 8.2.3 for more information.

Development, Distribution and Formal User Support  Starting with FDS 5, an online, open-source development environment 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 Information  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 describes how to use Smokeview. Other tools related to FDS and Smokeview can be found at the web site.

2.1 How to Acquire FDS and Smokeview

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

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

2.2 Computer Hardware Requirements

FDS requires a fast 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.

1Central Processing Unit
2.3 Computer Operating System (OS) and Software Requirements

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

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

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

Linux Pre-compiled executables are available that can be installed in an appropriate directory. Note that the installation package is simply an archive and no path variables are set. If the pre-compiled FDS executable does not work (usually because of library incompatibilities), the FDS 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 workstation, you can use the Windows version to view FDS output.

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

FDS in Parallel For those wishing to use multiple computers to run a single FDS calculation, MPI (Message Passing Interface) must be installed on each of the computers within the network that will be used for FDS computations.
Chapter 3

Running FDS

This chapter describes the procedure to run an FDS calculation. The primary requirement for any calculation is an FDS input file. The creation of an input file is covered in detail in Part II. If you are new to FDS and Smokeview, it is strongly suggested that you start with an existing data file, run it as is, and then make the appropriate changes to the input file for the desired scenario. Sample input files are included as part of the standard installation. By running a sample case, you become familiar with the procedure, learn how to use Smokeview, and ensure that your computer is up to the task before embarking on learning how to create new input files.

3.1 Starting an FDS Calculation

FDS can be run from the command prompt, or with a third party Graphical User Interface (GUI). In the discussion to follow, it is assumed that FDS is being run from the command prompt. FDS can be run on a single computer, using only one CPU, or it can be run on multiple computers and use multiple CPUs. For any operating system, there are two FDS executable files. The single CPU Windows executable is called fds5.exe. The parallel executable is called fds5_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 (click Start, then Run, then type “cmd”), and change directories (“cd”) to where the input file for the case is located, then run the code by typing at the command prompt

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
```

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

**Mac OS X, Unix, Linux**

Depending on the type of installation, you may need to set various path or environment variables in order to invoke FDS without a full path reference to the executable. The easiest way to do this is via an “alias” in your shell start-up script. For the example below, it is assumed that `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 procedure will be simplified in years to come, but for the moment, parallel-processing is still relatively new and requires more expertise in terms of understanding both the operating system and the network connections of a given set of computers.

FDS uses MPI (Message-Passing Interface) [3] to allow multiple computers to run a single FDS job. The main idea is that you must break up the FDS domain into multiple meshes, and then the flow field in each mesh is computed as a different process. Note the subtle difference between these terms – a process does not have the same meaning as a processor. The process can be thought of as a “task” that you would see in the Windows Task Manager or by executing the “top” command on a Linux/Unix machine. The processor refers to the computer hardware. A single processor may run multiple processes, for example. The computation
on a given FDS mesh is thought of as an individual *process*, and MPI handles the transfer of information between these *processes*. Usually, each mesh is assigned its own *process* in a parallel calculation, although it is also possible assign multiple meshes to a single *process*.\(^1\) In this way, large meshes can be computed on dedicated *processors*, while smaller meshes can be clustered together in a single *process* running on a single *processor*, without the need for MPI message passing between themselves.

Also note that FDS refers to its meshes by the numbers 1, 2, 3, and so on, whereas MPI refers to its processes by the numbers 0, 1, 2, and so on. Thus, Mesh 1 is assigned to Process 0; Mesh 2 to Process 1, and so on. As a user, you never actually number the meshes or the processes yourself, but error statements from FDS or from MPI might refer to the meshes or processes by number. As an example, if a five mesh FDS case is run in parallel, the first printout (usually to the screen unless otherwise directed) is:

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

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

There are different implementations of MPI, much like there are different Fortran and C compilers. Each implementation is essentially a library of subroutines called from FDS that transfer data from one 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

---

\(^1\)Assigning multiple meshes to a single process was introduced in FDS version 5.3.
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 implementation from Indiana University, is installed. 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.

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

---

2http://www.lam-mpi.org
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 6.4.2. 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 13.5.
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.

Along with the FDS User’s Guide, there are resources available on the Internet. These include an “Issue Tracker,” that allows you to report bugs, request new features, and ask specific clarifying questions, and “Group Discussions,” which support more general topics than just specific problems. Before using these online 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 specified time limit. Following is a list of common error statements and how to diagnose the problems:

**Input File Errors:** The most common errors in FDS are due to mis-typed input statements. These errors result in the immediate halting of the program and a statement like, “ERROR: Problem with the HEAD line.” For these errors, check the line in the input file named in the error statement. Make sure the parameter names are spelled correctly. Make sure that a / (forward slash) is put at the end of each namelist entry. Make sure that the right type of information is being provided for each parameter, like whether one real number is expected, or several integers, or whatever. Make sure there are no non-ASCII characters being used, as can sometimes happen when text is cut and pasted from other applications or word-processing software. Make sure zeros are zeros and O’s are O’s. Make sure 1’s are not 1’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 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 your responsibility to ensure that the computer has adequate resources to do the calculation. Remember, there is no limit to how big or how long FDS calculations can be – it depends on the resources of the computer. For any new simulation, try running the case with a modest-sized mesh, and gradually make refinements until the computer can no longer handle it. Then back off somewhat on the size of the calculation so that the computer can comfortably run the case. Trying to run with 90% to 100% of computer resources is risky. In fact, for a typical 32 bit Windows PC with 4 GB RAM, only 2 GB will be available to FDS, based on user feedback. If you want to run bigger cases, consider buying a computer with a 64 bit operating system or break up the calculation into multiple meshes and use parallel processing.
**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 Issue Tracker, try to systematically simplify the input file until the error goes away. This process usually brings to light some feature of the calculation responsible for the problem and helps in the debugging.

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

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

The unit, in this case 8598, is just a number that FDS has associated with one of the output files. If this error occurs just after the start of the calculation, you can try adding the phrase `FLUSH_FILE_BUFFERS=.FALSE.` on the `DUMP` line of the input file (see Section 14.1). This will prevent FDS from attempting to flush the contents of the internal buffers, something it does to make it possible to view the FDS output in Smokeview during the FDS simulation.

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

### 4.3 Support Requests and Bug Tracking

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

If that does not solve the problem, report the problem with as much information about the error message and circumstances related to the problem. The input file should be simplified as much as possible so that the bug occurs early in the calculation. Attach the simplified input file if necessary, following the instructions provided at the web site. In this way, the developers can quickly run the problem input file and hopefully diagnose the problem.

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

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

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Symptom of Problem</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO PRODUCTION</td>
<td>Algorithm to predict CO production</td>
<td>Inaccuracies found in comparison to experiments</td>
<td>Research usage only</td>
</tr>
<tr>
<td>Liquid Fuels</td>
<td>Pyrolysis model of evaporating liquid fuel</td>
<td>Inaccuracies found in comparison to experiments; physical and numerical sensitivity</td>
<td>Research usage only</td>
</tr>
<tr>
<td>Solid Fuels</td>
<td>Pyrolysis model of solid fuel</td>
<td>Results in FDS 5.4 and beyond different than previous versions</td>
<td>Read Section 8.4.4</td>
</tr>
</tbody>
</table>
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, and never use tab stops. Some machines do not recognize the spaces or the length of the tab stops. Comments and notes can be written into the file so long as nothing comes before the & except a space and nothing comes between the ampersand & and the slash / except appropriate parameters corresponding to that particular namelist group.

---

\(^1\)ASCII – American Standard Code for Information Interchange

\(^2\)A *namelist* is a Fortran input record.
The parameters in the input file can be integers (\(T_{\text{END}}=5400\)), real numbers (\(\text{CO\_YIELD}=0.008\)), groups of real numbers or integers (\(\text{XYZ}=6.04,0.28,3.65\)) or (\(\text{IJK}=90,36,38\)), character strings:

\[ \text{CHID}='\text{WTC\_05\_v5}' \]
groups of character strings:

\[ \text{SURF\_IDS}='\text{burner}','\text{STEEL}','\text{BRICK}' \]
or logical parameters:

\[ \text{POROUS\_FLOOR}=\text{.FALSE.} \]

A logical parameter is either \text{.TRUE.} or \text{.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 \(\text{DT}=0.02\), 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 \(\text{MATL\_MASS\_FRACTION}(\text{IL},\text{IC})\) is intended to convey to FDS the mass fraction of component \(\text{IC}\) of layer \(\text{IL}\). For example, if the mass fraction of the second material of the third layer is 0.5, then write

\[ \text{MATL\_MASS\_FRACTION}(3,2)=0.5 \]

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

\[ \text{MATL\_MASS\_FRACTION}(1,1:3)=0.5,0.4,0.1 \]

which means that the first three materials of layer 1 have mass fractions of 0.5, 0.4, and 0.1, respectively.

The notation \(1:3\) means array element 1 through 3, inclusive.

---

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

---

Note that depending on compiler and operating system, some text file encodings may not work on all systems. If file reading errors occur and no typographical errors can be found in the input file, try saving the input file using a different encoding. It does not appear that current Fortran compilers support the UTF-8 encoding standard for reading Namelist inputs.

---

### 5.3 Input File Structure

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

\[ \&\text{TAIL} / \]

as the last line at the end of the input file. This completes the file from \&\text{HEAD} to \&\text{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 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_DEV=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.

---

3 The actual input file, WTC_05_v5.fds, is part of the FDS Validation Suite
<table>
<thead>
<tr>
<th>Group Name</th>
<th>Namelist Group Description</th>
<th>Reference Section</th>
<th>Parameter Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>BNDF</td>
<td>Boundary File Output</td>
<td>14.2.4</td>
<td>15.1</td>
</tr>
<tr>
<td>CLIP</td>
<td>Min/Max Clipping Parameters</td>
<td>6.7</td>
<td>15.2</td>
</tr>
<tr>
<td>CTRL</td>
<td>Control Function Parameters</td>
<td>13.5</td>
<td>15.3</td>
</tr>
<tr>
<td>DEVC</td>
<td>Device Parameters</td>
<td>13.1</td>
<td>15.4</td>
</tr>
<tr>
<td>DUMP</td>
<td>Output Parameters</td>
<td>14.1</td>
<td>15.5</td>
</tr>
<tr>
<td>HEAD</td>
<td>Input File Header</td>
<td>6.1</td>
<td>15.6</td>
</tr>
<tr>
<td>HOLE</td>
<td>Obstruction Cutout</td>
<td>7.3</td>
<td>15.7</td>
</tr>
<tr>
<td>HVAC</td>
<td>Heating, Vent., Air Cond.</td>
<td>9.5</td>
<td>15.8</td>
</tr>
<tr>
<td>INIT</td>
<td>Initial Condition</td>
<td>6.5</td>
<td>15.9</td>
</tr>
<tr>
<td>ISOF</td>
<td>Isosurface File Output</td>
<td>14.2.5</td>
<td>15.10</td>
</tr>
<tr>
<td>MATL</td>
<td>Material Property</td>
<td>8.3</td>
<td>15.11</td>
</tr>
<tr>
<td>MESH</td>
<td>Mesh Parameters</td>
<td>6.3</td>
<td>15.12</td>
</tr>
<tr>
<td>MISC</td>
<td>Miscellaneous</td>
<td>6.4</td>
<td>15.13</td>
</tr>
<tr>
<td>MULT</td>
<td>Multiplier Parameters</td>
<td>7.2.2</td>
<td>15.14</td>
</tr>
<tr>
<td>OBST</td>
<td>Obstruction</td>
<td>7.2</td>
<td>15.15</td>
</tr>
<tr>
<td>PART</td>
<td>Lagrangian Particle</td>
<td>12</td>
<td>15.16</td>
</tr>
<tr>
<td>PRES</td>
<td>Pressure Solver Parameters</td>
<td>6.6</td>
<td>15.17</td>
</tr>
<tr>
<td>PROF</td>
<td>Profile Output</td>
<td>14.2.2</td>
<td>15.18</td>
</tr>
<tr>
<td>PROP</td>
<td>Device Property</td>
<td>13.3</td>
<td>15.19</td>
</tr>
<tr>
<td>RADI</td>
<td>Radiation</td>
<td>11.3</td>
<td>15.20</td>
</tr>
<tr>
<td>RAMP</td>
<td>Ramp Profile</td>
<td>10</td>
<td>15.21</td>
</tr>
<tr>
<td>REAC</td>
<td>Reactions</td>
<td>11.1</td>
<td>15.22</td>
</tr>
<tr>
<td>SLCF</td>
<td>Slice File Output</td>
<td>14.2.3</td>
<td>15.23</td>
</tr>
<tr>
<td>SPEC</td>
<td>Species Parameters</td>
<td>11.2</td>
<td>15.24</td>
</tr>
<tr>
<td>SURF</td>
<td>Surface Properties</td>
<td>7.1</td>
<td>15.25</td>
</tr>
<tr>
<td>TABL</td>
<td>Tabulated Particle Data</td>
<td>10</td>
<td>15.26</td>
</tr>
<tr>
<td>TIME</td>
<td>Simulation Time</td>
<td>6.2</td>
<td>15.27</td>
</tr>
<tr>
<td>TRNX</td>
<td>Mesh Stretching</td>
<td>6.3.5</td>
<td>15.28</td>
</tr>
<tr>
<td>VENT</td>
<td>Vent Parameters</td>
<td>7.4</td>
<td>15.29</td>
</tr>
<tr>
<td>ZONE</td>
<td>Pressure Zone Parameters</td>
<td>9.7</td>
<td>15.30</td>
</tr>
</tbody>
</table>
Chapter 6

Setting the Bounds of Time and Space

6.1 Naming the Job: The HEAD Namelist Group (Table 15.6)

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 should be meaningful and help to organize the project. 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. If CHID is not specified, then it will be set to the name of the input file minus everything at and beyond the first period.

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

6.2 Simulation Time: The TIME Namelist Group (Table 15.27)

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

6.2.1 Basics

Usually, only the duration of the simulation is required on this line, via the parameter T_END. The default is 1 s. Note: the older TWFIN 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 use the parameter \texttt{T\_BEGIN} to specify the time written to file for the first time step. This would be useful for matching timelines of experimental data or video recordings.

| Time-based \texttt{RAMP}s are evaluated using the actual time if the \texttt{RAMP} activation time is the same as \texttt{T\_BEGIN}; otherwise, they are evaluated using the time from when the \texttt{RAMP} activates. Therefore, if you are setting \texttt{T\_BEGIN} in order to test a time-based \texttt{CTRL} or \texttt{DEVC} that is ultimately linked to a \texttt{RAMP}, then you should set \texttt{T\_BEGIN} to be slightly less than the time the \texttt{RAMP} will activate. For example if you are testing a \texttt{VENT} that is to open at 10 s whose \texttt{SURF\_ID} uses a \texttt{RAMP}, \texttt{T\_BEGIN} should be set slightly less than 10 s. |

### 6.2.2 Special Topic: Controlling the Time Step

The initial time step size can be specified with \texttt{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 \texttt{DT} is \(5 (\delta x \delta y \delta z) \frac{1}{\sqrt{gH}}\) s, where \(\delta x\), \(\delta y\), and \(\delta z\) are the dimensions of the smallest mesh cell, \(H\) is the height of the computational domain, and \(g\) is the acceleration of gravity. Note that by default the time step is never allowed to increase above its initial value. To allow this to happen, set \texttt{RESTRICT\_TIME\_STEP=.FALSE.}.

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

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

Finally, if you want to prevent FDS from automatically changing the time step, set \texttt{LOCK\_TIME\_STEP=.TRUE.} on the \texttt{TIME} line, in which case the specified time step, \texttt{DT}, will not be adjusted. This parameter is intended for diagnostic purposes only, for example, timing program execution. It can lead to numerical instabilities if the initial time step is set too high.

### 6.2.3 Special Topic: Steady-State Applications

Occasionally, there are applications in which only the steady-state solution (in a time-averaged sense) is desired. However, the time necessary to heat the walls to steady-state can make the cost of the calculation prohibitive. In these situations, if you specify a \texttt{TIME\_SHRINK\_FACTOR} of, say, 10, the specific heats of the various materials is reduced by a factor of 10, speeding up the heating of these materials roughly by 10. An example of an application where this parameter is handy is a validation experiment where a steady heat source warms up a compartment to a nearly equilibrium state at which point time-averaged flow quantities are measured.
6.3 Computational Meshes: The MESH Namelist Group (Table 15.12)

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

6.3.1 Basics

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

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

defines a mesh that spans the volume starting at the origin and extending 1 m in the positive x direction, 2 m in the positive y direction, and 3 m in the positive z direction. The mesh is subdivided into uniform cells via the parameter IJK. In this example, the mesh is divided into 10 cm cubes. If it is desired that the mesh cells in a particular direction not be uniform in size, then the namelist groups TRNX, TRNY and/or TRNZ may be used to alter the uniformity of the mesh (See Section 6.3.5).

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

Note that it is best if the mesh cells resemble cubes, that is, the length, width and height of the cells ought to be roughly the same.

Because an important part of the calculation uses a Poisson solver based on Fast Fourier Transforms (FFTs) in the y and z directions, the second and third dimensions of the mesh should each be of the form $2^l3^m5^n$, where $l$, $m$ and $n$ are integers. For example, $64 = 2^6$, $72 = 2^33^2$ and $108 = 2^23^3$ are good mesh cell divisions, but 37, 99 and 109 are not. The first number of mesh cell divisions (the $i$ in IJK) does not use FFTs and need not be given as a product of small numbers. However, you should experiment with different values of divisions to ensure that those that are ultimately used do not unduly slow down the calculation.

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

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

6.3.2 Two-Dimensional and Axially-Symmetric Calculations

The governing equations solved in FDS are written in terms of a three dimensional Cartesian coordinate system. However, a two dimensional Cartesian or two dimensional cylindrical (axially-symmetric) calculation can be performed by setting the $j$ in the IJK triplet to 1 on the MESH line. For axial symmetry, add CYLINDRICAL=.TRUE. to the MESH line, and the coordinate $x$ is then interpreted as the radial coordinate $r$. No boundary conditions should be set at the planes $y = YMIN=XB(3)$ or $y = YMAX=XB(4)$, nor at $r = XMIN=XB(1)$ in an axially-symmetric calculation in which $r = XB(1)=0$. For better visualizations, the difference between $XB(4)$ and $XB(3)$ should be small so that the Smokeview rendering appears to be in 2-D. An example of an axially-symmetric helium plume (helium_2d) is given in Section 6.4.4.


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

To run FDS in parallel using MPI (Message Passing Interface), you must break up the computational domain into multiple meshes so that the workload can be divided among the available processors. In general, it is better to run multiple mesh cases with the parallel version of FDS if you have the computers available, but be aware that two computers will not necessarily finish the job in half the time as one. For the parallel version to work well, there has to be a comparable number of cells in each mesh, or otherwise most of the computers will sit idle waiting for the one with the largest mesh to finish processing each time step. You can use multiple meshes even when running the serial version of FDS, in which case one CPU will serially process each mesh, one by one. Why do this? For one, if you set SYNCHRONIZE=.FALSE.

on the TIME line, then in each mesh, the governing equations will be solved with a time step based on the flow speed within that particular mesh. Because each mesh can have different time steps, this technique can save CPU time by requiring relatively coarse meshes to be updated only when necessary. Coarse meshes
are best used in regions where temporal and spatial gradients of key quantities are small or unimportant. Be aware, however, that unsynchronized time steps are more likely to lead to numerical instabilities.

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

Usually in a multi-mesh calculation, each mesh is assigned its own process, and each process its own processor. However, it is possible to assign more than one mesh to a single process, and it is possible to assign more than one process to a single processor. Consider a case that involves six meshes:

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

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

### 6.3.4 Mesh Alignment

Whether the calculation is to be run on a single processor, or on multiple processors, the rules of prescribing multiple meshes are similar, with some issues to keep in mind. The most important rule of mesh alignment is that abutting cells ought to have the same cross sectional area, or integral ratios, as shown in Fig. 6.2. The requirement of integral mesh alignment is new starting with FDS version 5.1. It is a more restrictive requirement than in previous versions because it was becoming too difficult to maintain complete flexibility in alignment while trying to improve the accuracy of the methodology. The following rules of thumb should also be followed when setting up a multiple mesh calculation:

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

- In general, there is little advantage to overlapping meshes because information is only exchanged at exterior boundaries. This means that a mesh that is completely embedded within another receives information at its exterior boundary, but the larger mesh receives no information from the mesh embedded within. Essentially, the larger, usually coarser, mesh is doing its own simulation of the scenario and
This is the ideal kind of mesh to mesh alignment.

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

This is allowed, but of questionable value.

This is no longer allowed in FDS 5.1 and higher.

Figure 6.2: Rules governing the alignment of meshes.
is not affected by the smaller, usually finer, mesh embedded within it. Details within the fine mesh, especially related to fire growth and spread, may not be picked up by the coarse mesh. In such cases, it is preferable to isolate the detailed fire behavior within one mesh, and position coarser meshes at the exterior boundary of the fine mesh. Then the fine and coarse meshes mutually exchange information.

- Be careful when using the shortcut convention of declaring an entire face of the domain to be an **OPEN** vent. Every mesh takes on this attribute. See Section 7.4 for more details.

- If a planar obstruction is close to where two meshes abut, make sure that each mesh "sees" the obstruction. If the obstruction is even a millimeter outside of one of the meshes, that mesh does not account for it, in which case information is not transferred properly between meshes.

**Accuracy of the Parallel Calculation**

Experiment with different mesh configurations using relatively coarse mesh cells to ensure that information is being transferred properly from mesh to mesh. There are two issues of concern. First, does it appear that the flow is being badly affected by the mesh boundary? If so, try to move the mesh boundaries away from areas of activity. Second, is there too much of a jump in cell size from one mesh to another? If so, consider whether the loss of information moving from a fine to a coarse mesh is tolerable.

**Efficiency of the Parallel Calculation**

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

6.3.5 **Mesh Stretching: The TRNX, TRNY and/or TRNZ Namelist Groups (Table 15.28)**

By default the mesh cells that fill the computational domain are uniform in size. However, it is possible to specify that the cells be non-uniform in one or two of the three coordinate directions. For a given coordinate direction, x, y or z, a function can be prescribed that transforms the uniformly-spaced mesh to a non-uniformly spaced mesh. **Be careful with mesh transformations!** If you shrink cells in one region you must stretch cells somewhere else. When one or two coordinate directions are transformed, the aspect ratio of the mesh cells in the 3D mesh will vary. To be on the safe side, transformations that alter the aspect ratio of cells beyond 2 or 3 should be avoided. Keep in mind that the large eddy simulation technique is based on the assumption that the numerical mesh should be fine enough to allow the formation of eddies that are responsible for the mixing. In general, eddy formation is limited by the largest dimension of a mesh cell, thus shrinking the mesh resolution in one or two directions may not necessarily lead to a better simulation if the third dimension is large.

Transformations, in general, reduce the efficiency of the computation, with two coordinate transformations impairing efficiency more than a transformation in one coordinate direction. Experiment with different meshing strategies to see how much of a penalty you will pay.
Here is an example of how to do a mesh transformation. Suppose your mesh is defined

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

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

Two types of transformation functions are allowed. The first, and simplest, is a piecewise-linear function. Figure 6.3 gives an example of a piecewise-linear transformation. The graph indicates how 15 uniformly spaced mesh cells along the horizontal axis are transformed into 15 non-uniformly spaced cells along the vertical axis. In this case, the function is made up of straight line segments connecting points (CC,PC), in increasing order, as specified by the following lines in the input file:

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

The parameter CC refers to the Computational Coordinate, \( \xi \), 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

\[
\frac{d^n f(CC)}{d\xi^n} = PC
\]
Figure 6.4 gives an example of a polynomial transformation, for which the parameters are specified (assuming that this is the third mesh):

\[ &\text{TRNX IDERIV}=0, \text{CC}=0.75, \text{PC}=0.75, \text{MESH\_NUMBER}=3 / \]
\[ &\text{TRNX IDERIV}=1, \text{CC}=0.75, \text{PC}=0.50, \text{MESH\_NUMBER}=3 / \]

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

Do not specify either linear transformation points or IDERIV=0 points at coordinate values corresponding to the mesh boundaries.

6.3.6 Mesh Resolution

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

\[
D^* = \left( \frac{Q}{\rho_\infty c_p T_\infty \sqrt{g}} \right)^{\frac{5}{3}}
\]

and \( \delta x \) 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 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\(^2\).

There are a number of special output quantities that provide local measures of grid resolution. See Section 14.3.15 for details.

\(^1\)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.

\(^2\)For the validation study sponsored by the U.S. Nuclear Regulatory Commission [4], the \( D^*/\delta x \) values ranged from 4 to 16.
6.4 Miscellaneous Parameters: The MISC Namelist Group (Table 15.13)

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

6.4.1 Basics

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

```
&MISC 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). This feature is appropriate only for simulations that use mesh cells that are on the order of a millimeter or less in size, or for diagnostic purposes.

GVEC The 3 components of gravity, in m/s². 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.

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

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

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’, a non-reacting solid boundary whose temperature is fixed at TMPA. You do not need to define ’INERT’ via a SURF line.

TMPA Ambient temperature, the temperature of everything at the start of the simulation. The default is 20 °C.

U0, V0, W0 Initial values of the gas velocity in each of the coordinate directions. Normally, these are all 0 m/s, but there are a few applications where it is convenient to start the flow immediately, like in an outdoor simulation involving wind.
6.4.2 Special Topic: 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 13.5) 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.

6.4.3 Special Topic: Defying Gravity

Most users of FDS assume that the acceleration of gravity points in the negative z direction, or more simply, downward. However, to change the direction of gravity to model a sloping roof or tunnel, for example, specify the gravity vector on the MISC line with a triplet of numbers of the form GVEC=0.,0.,-9.81, with units of m/s². This is the default, but it can be changed to be any direction.

There are a few special applications where you might want to vary the gravity vector as a function of time or as a function of the first spatial coordinate, x. For example, on board the Space Shuttle or International Space Station, small motions can cause temporal changes in the normally zero level of gravity, an effect known as “g-jitter.” More commonly, in tunnel fire simulations, it is sometimes convenient to change the direction of gravity to mimic the change in slope. The slope of the tunnel might change as you travel through it; thus, you can tell FDS where to redirect gravity. For either a spatially or temporally varying direction and/or magnitude of gravity, do the following. First, on the MISC line, set the three components of gravity, GVEC, to some “base” state like GVEC=1.,1.,1., which gives you the flexibility to vary all three components. Next, designate “ramps” for the individual components, RAMP_GX, RAMP_GY, and RAMP_GZ,
all of which are specified on the MISC line. There is more discussion of RAMPs in Section 10, but for now you can use the following as a simple template to follow:

```plaintext
&MISC GVEC=1.,0.,1., RAMP_GX='x-ramp', RAMP_GZ='z-ramp' /
&RAMP ID='x-ramp', X= 0., F=0.0 / 
&RAMP ID='x-ramp', X= 50., F=0.0 / 
&RAMP ID='x-ramp', X= 51., F=-0.49 / 
&RAMP ID='x-ramp', X=100., F=-0.49 / 
&RAMP ID='z-ramp', X= 0., F=-9.81 / 
&RAMP ID='z-ramp', X= 50., F=-9.81 / 
&RAMP ID='z-ramp', X= 51., F=-9.80 / 
&RAMP ID='z-ramp', X=100., F=-9.80 / 
```

Note that both the $x$ and $z$ components of gravity are functions of $x$. FDS has been programmed to only allow variation in the $x$ coordinate. Note also that $F$ is just a multiplier of the “base” gravity vector components, given by GVEC. This is why using the number 1 is convenient – it allows you to specify the gravity components on the RAMP lines directly. The effect of these lines is to model the first 50 m of a tunnel without a slope, but the second 50 m with a 5% slope upwards. Note that the angle from vertical of the gravity vector due to a 5% slope is $\tan^{-1}0.05 = 2.86^\circ$ and that 0.49 and 9.80 are equal to the magnitude of the gravity vector, 9.81 m/s$^2$, multiplied by the sine and cosine of 2.86$^\circ$, respectively. To check your math, the sum of the squares of the gravity components ought to equal 9.81. Notice in this case that the $y$ direction has been left out because there is no $y$ variation in the gravity vector.

To vary the direction and/or magnitude of gravity in time, follow the same procedure but replace the $x$ in the RAMP lines with a $T$.

### 6.4.4 Special Topic: The Baroclinic Vorticity

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

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

(6.2)

The pressure term is written like this so that a separable elliptic partial differential equation can be solved for the “total” pressure, $H \equiv |\mathbf{u}|^2/2 + \tilde{p}/\rho$, using a direct solver. The second term is calculated based on the pressure field from the previous time step, a slight approximation necessary to render the pressure equation separable. This term is sometimes referred to as the baroclinic torque, and it is responsible for generating vorticity due to the non-alignment of pressure and density gradients. For an LES calculation, it can be included by setting BAROCLINIC=.TRUE. on the MISC line. For DNS, it is included by default. For example, in the simple helium plume test case below, neglecting the baroclinic torque changes the puffing behavior noticeably. In other applications, however, its effect is less significant. For further discussion of its effect, see Ref. [5].

In versions of FDS prior to 5.5.0, BAROCLINIC=.FALSE. was the default for LES. For FDS 5.5.0 and 5.5.1, however, BAROCLINIC was set to .TRUE. by default. Its use was responsible for numerical instabilities in a number of cases. In test versions of FDS 6, the problems do not occur, thus BAROCLINIC=.TRUE. for test versions of FDS 6.
Example Case: helium_2d

This case demonstrates the use of baroclinic correction for an axially-symmetric helium plume. Note that the governing equations solved in FDS are written in terms of a three dimensional Cartesian coordinate system. However, a two dimensional Cartesian or two dimensional cylindrical (axially-symmetric) calculation can be performed by setting the number of cells in the y direction to 1. An example of an axially-symmetric helium plume is shown in Figure 6.5.

```
&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 TFWIN=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. /
&TAIL /
```

Figure 6.5: Simulation of a helium plume.

6.4.5 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.

Example Case: 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 °C with the surrounding ambient temperature initialized to 10 °C. 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 °C/m.

\[ \rho_0(z) = \rho_\infty e^{\frac{-z}{h}} \quad (6.3) \]

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/m³, 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

\[ \delta P = \frac{W P_0 g}{R_0} \left( \frac{1}{T_{\text{ambient}}} - \frac{1}{T_{\text{building}}} \right) h \quad (6.4) \]

39
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, an error of 0.5 %.

### 6.4.6 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 \( \mu \) is modeled

\[
\mu_{\text{LES}} = \rho (C_s \Delta)^2 \left( 2 \, \mathbf{S}_{ij} : \mathbf{S}_{ij} - \frac{2}{3} (\mathbf{V} \cdot \mathbf{u})^2 \right)^{\frac{1}{2}}
\]

where \( C_s \) is an empirical constant and \( \Delta \) 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

\[
k_{\text{LES}} = \frac{\mu_{\text{LES}} c_p}{\Pr_t}; \quad (\rho D)_{I,\text{LES}} = \frac{\mu_{\text{LES}}}{\Sc_t}
\]

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 \( C_s = 0.2 \), \( \Pr_t = 0.5 \), and \( \Sc_t = 0.5 \) via the parameters \texttt{CSMAG}, \texttt{PR}, and \texttt{SC} on the \texttt{MISC} line. A more detailed discussion of these parameters is given in the FDS Technical Reference Guide [6].

### 6.4.7 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 \( \delta x \) by \( \delta y \) by \( \delta z \) with velocity components \( u, v, \) and \( w \), the CFL number is defined:

\[
\text{CFL} = \delta t \max \left( \frac{|u|}{\delta x}, \frac{|v|}{\delta y}, \frac{|w|}{\delta z} \right)
\]

Every time step, the CFL number is computed in each mesh cell, and the time step, \( \delta t \), is adjusted if the maximum value of the CFL number is not between \texttt{CFL\_MIN} and \texttt{CFL\_MAX}, whose default values are 0.8 and 1.0, respectively. These values are included in the \texttt{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:

\[
\text{VN} = 2 \max \left( v, D, \frac{k}{\rho c_p} \right) \delta t \left( \frac{1}{\delta x^2} + \frac{1}{\delta y^2} + \frac{1}{\delta z^2} \right)
\]

Like the CFL number, VN is computed in each mesh cell, and the time step is adjusted if VN is outside the range between \texttt{VN\_MIN} and \texttt{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.

If you want to prevent FDS from automatically changing the time step, set `LOCK_TIME_STEP=.TRUE.` on the `TIME` line, in which case the specified time step, DT, will not be adjusted. This parameter is intended for diagnostic purposes only, for example, timing program execution. It can lead to numerical instabilities if the initial time step is set too high.
6.5 Special Topic: Unusual Initial Conditions: The \texttt{INIT} Namelist Group

(\textit{Table 15.9})

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

\begin{verbatim}
&INIT XB=0.5,0.8,2.1,3.4,2.5,3.6, TEMPERATURE=30. /
\end{verbatim}

Here, within the region whose bounds are given by the sextuplet $XB$, the initial temperature shall be 30 $^\circ$C instead of the ambient. This construct can also be used for \texttt{DENSITY} or \texttt{MASS\_FRACTION(N)} where $N$ indicates the $N$th species listed in the input file. Make sure that you specify all species (components of \texttt{MASS\_FRACTION(N)}) on the same \texttt{INIT} line.

The \texttt{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 \texttt{TMP\_INNER} on the \texttt{SURF} line. An initial velocity can be prescribed via $U_0$, $V_0$, and $W_0$ on the \texttt{MISC} line.

6.6 Special Topic: Improving the Pressure Solver: The \texttt{PRES} Namelist Group

(\textit{Table 15.17})

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

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

If either the error in the normal component of the velocity at a mesh interface or at a solid boundary is large, you can reduce it by making more than the default number of calls to the pressure solver at each time step. To do so, specify \texttt{VELOCITY\_TOLERANCE} on the \texttt{PRES} line to be the maximum allowable normal velocity component on the solid boundary or the largest error at a mesh interface. It is in units of m/s. If you set this, experiment with different values, and monitor the number of pressure iterations required at each time step to achieve your desired tolerance. The number of iterations are written out to the file \texttt{CHID.out}. If you use a value that is too small, the CPU time required might be prohibitive. The maximum number of iterations per time step is given by \texttt{MAX\_PRESSURE\_ITERATIONS}, also on the \texttt{PRES} line. Its default value is 10000.
Example Case: duct_flow

To demonstrate how to improve the accuracy of the pressure solver, consider the flow of air through a square duct that crosses several meshes. In the case called duct_flow, air is pushed through a 1 m$^2$ duct at 1 m/s.

With no thermal expansion, the volume flow into the duct ought to equal the volume flow out of the duct. Figure 6.6 displays the computed inflow and outflow as a function of time, and the number of pressure iterations required. The outflow does not match the inflow exactly because of inaccuracies at the solid and mesh boundaries. The VELOCITY_TOLERANCE has been set to 0.01 m/s.

![Figure 6.6](image)

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

6.7 Special Topic: Setting Limits: The CLIP Namelist Group (Table 15.2)

The algorithms in FDS are designed to work within a certain range of values for the thermodynamic quantities density, temperature and mass fraction. To prevent unphysical results, there are bounds placed on these variables to prevent a single spurious value from causing a numerical instability. It also prevents out of range errors from calls to temperature-dependent look-up tables. By default, FDS determines the lowest and highest values of the variables based on user input, but it is not possible in all cases to anticipate just how low or high a given value might be. Thus, on rare occasions you might need to set upper or lower bounds on the density, temperature, or species mass fractions. These parameters are input under the namelist group called CLIP. The parameters are listed in Table 15.2. You only need to set these values if you notice that one of them appears to be “cut off” when examining the results in Smokeview. For typical fire scenarios, you need not set these values, but if you anticipate relatively low or high values in an unusual case, take a look at the calculation results to determine if a change in the bounds is needed.
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.

7.1 Bounding Surfaces: The SURF Namelist Group (Table 15.25)

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

The default boundary condition for all solid surfaces is that of a smooth inert wall with the temperature fixed at $\text{TMPA}$, and is referred to as ‘INERT’. If only this boundary condition is needed, there is no need to add any SURF lines to the input file. If additional boundary conditions are desired, they are to be listed one boundary condition at a time. Each SURF line consists of an identification string $\text{ID}=\ldots$ to allow references to it by an obstruction or vent. Thus, on each OBST and VENT line that are to be described below, the character string $\text{SURF_ID}=\ldots$ 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 $\text{SURF_DEFAULT}=\text{CONCRETE}$ on the MISC line.

7.2 Creating Obstructions: The OBST Namelist Group (Table 15.15)

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

7.2.1 Basics

Each OBST line contains the coordinates of a rectangular solid within the flow domain. This solid is defined by two points $(x_1,y_1,z_1)$ and $(x_2,y_2,z_2)$ that are entered on the OBST line in terms of the sextuplet $\text{XB} = X_1, X_2, Y_1, Y_2, Z_1, Z_2$

In addition to the coordinates, the boundary conditions for the obstruction can be specified with the parameter $\text{SURF_ID}$, which designates which SURF group (Section 7.1) to apply at the surface of the obstruction.
If the obstruction has different properties for its top, sides and bottom, do not specify only one SURF_ID. Instead, use SURF_IDS, an array of three character strings specifying the boundary condition IDs for the top, sides and bottom of the obstruction, respectively. If the default boundary condition is desired, then SURF_ID(S) need not be set. However, if at least one of the surface conditions for an obstruction is the inert default, it can be referred to as ’INERT’, but it does not have to be explicitly defined. For example:

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

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

Some additional features of obstructions are as follows:

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

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

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

- Obstructions can have zero thickness. Often, thin sheets, like a window, form a barrier, but if the numerical mesh is coarse relative to the thickness of the barrier, the obstruction might be unnecessarily large if it is assumed to be one layer of mesh cells thick. All faces of an obstruction are shifted to the closest mesh cell. If the obstruction is very thin, the two faces may be approximated on the same cell face. FDS and Smokeview render this obstruction as a thin sheet, but it is allowed to have thermally thick boundary conditions. This feature is fragile, especially in terms of burning and blowing gas. A thin sheet obstruction can only have one velocity vector on its face, thus a gas cannot be injected reliably from a thin obstruction because whatever is pushed from one side is necessarily pulled from the other. For full functionality, the obstruction should be specified to be at least one mesh cell thick. Thin sheet obstructions work fine as flow barriers, but other features are fragile and should be used with caution. To prevent FDS from allowing thin sheet obstructions, set THICKEN_OBSTRUCTIONS=.TRUE. on the MISC line, or THICKEN=.TRUE. on each OBST line for which the thin sheet assumption is not allowed.

- 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 13.4.1 for details.

- If two obstructions overlap at one or more faces, the one listed last in the input file takes precedence over the one listed first, in the sense that the latter’s surface properties will be applied to the overlapping face. Smokeview renders both obstructions independently of each other, often leading to an unsightly cross-hatching of the two surface colors where there is an overlap. A simple remedy for this is to “shrink” the first obstruction slightly by adjusting its coordinates (XB) accordingly. Then, in Smokeview, toggle the “q” key to show the obstructions as you specified them, rather than as FDS rendered them.

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

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

- In rare cases, you might not want to allow a vent to be attached to a particular obstruction, in which case set ALLOW_VENT=.FALSE.

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

- Obstructions are drawn solid in Smokeview. To draw an outline representation, set OUTLINE=.TRUE.

### 7.2.2 Repeated Obstructions: The MULT Namelist Group (Table 15.14)

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

```
&MULT ID='m1', DX=1.2, DY=2.4, I_LOWER=-2, I_UPPER=3, J_LOWER=0, J_UPPER=5 /
&OBST XB=..., MULT_ID='m1' /
```

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

\[
x_i = x_0 + \delta x_0 + i \Delta x \quad ; \quad I_{\text{LOWER}} \leq i \leq I_{\text{UPPER}}
\]

\[
y_j = y_0 + \delta y_0 + j \Delta y \quad ; \quad J_{\text{LOWER}} \leq j \leq J_{\text{UPPER}}
\]

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

A generalization of this idea is to replace the parameters, \( DX, DY, \) and \( DZ, \) with a sextuplet called \( DXB. \) The six entries in \( DXB \) increment the respective values of the obstruction coordinates given by \( XB. \) For example, define the lower \( y \) bound of the original obstruction by \( XB(3,0) \). The \( n \)th obstruction would be:

\[
XB(3,N) = XB(3,0) + N \times DXB(3)
\]

Notice that we use \( N_{\text{LOWER}} \) and \( N_{\text{UPPER}} \) to denote the range of \( N. \) This more flexible input scheme allows you to create, for example, a slanted roof in which the individual roof segments shorten as they ascend to the top. This feature is demonstrated by the following short input file that creates a hollowed out pyramid using the four perimeter obstructions that form the outline of its base:

```
&HEAD CHID='pyramid', TITLE='Simple demo of multiplier function' /
&MESH IJK=100,100,100, XB=0.0,1.0,0.0,1.0,0.0,1.0 /
&TIME T_END=0. /
```
Figure 7.1: An example of the multiplier function.

```plaintext
&MULT ID='south', DXB=0.01,-.01,0.01,0.01,0.01,0.01, N_LOWER=0, N_UPPER=39 /  
&MULT ID='north', DXB=0.01,-.01,-.01,-.01,0.01,0.01, N_LOWER=0, N_UPPER=39 /  
&MULT ID='east', DXB=-.01,-.01,0.01,-.01,0.01,0.01, N_LOWER=0, N_UPPER=39 /  
&MULT ID='west', DXB=0.01,0.01,0.01,-.01,0.01,0.01, N_LOWER=0, N_UPPER=39 /  
&OBST XB=0.10,0.90,0.10,0.11,0.10,0.11, MULT_ID='south', COLOR='RED' /  
&OBST XB=0.10,0.11,0.11,0.89,0.10,0.11, MULT_ID='north', COLOR='BLUE' /  
&OBST XB=0.10,0.11,0.11,0.89,0.10,0.11, MULT_ID='west', COLOR='GREEN' /  
&OBST XB=0.89,0.90,0.11,0.89,0.10,0.11, MULT_ID='east', COLOR='CYAN' /
```

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

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

### 7.2.3 Non-rectangular Geometry and Sloped Ceilings

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

Do not apply `SAWTOOTH=.FALSE.` to obstructions that have any `SURF_IDs` with the attribute `BURN_AWAY=.TRUE.`
Example Case: sawtooth

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

\begin{verbatim}
&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' / 
\end{verbatim}

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

![Figure 7.2: Simple example of SAWTOOTH=.FALSE.](image)
7.3 Creating Voids: The `HOLE` Namelist Group (Table 15.7)

The `HOLE` namelist group is used to define parameters (Table 15.7) to carve a hole out of an existing obstruction or set of obstructions. To do this, add lines of the form

```plaintext
&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:

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

and you want to create a door, add this:

```plaintext
&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 13.4.1 for details. To allow a hole to be controlled with either the `CTRL` or `DEVC` namelist groups, you will need to add the `CTRL_ID` or `DEVC_ID` parameter respectively, to the `HOLE` line.

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

When a `HOLE` is in a `.FALSE.` state, an obstruction is placed in the hole. To make this obstruction transparent, the `TRANSPARENCY` parameter should be specified by a real number from 0 to 1. Note that if `TRANSPARENCY` is specified, then either a `COLOR` or `RGB` triplet ought to be specified as well. A `TRANSPARENCY` value near, but not equal to, zero can be used to simulate a window when the `HOLE`’s `INITIAL_STATE`=.FALSE. When the `DEVC` or `CTRL` is activated and changes the state of the hole to `.TRUE.` the `HOLE` is then open and completely transparent. See Section 13.4.1 for an example.

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

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

Note that a `HOLE` has no effect on a `VENT` or a mesh boundary. It only applies to `OBST`structions.
7.4 Applying Surface Properties: The VENT Namelist Group (Table 15.29)

Whereas the OBST group is used to specify obstructions within the computational domain, the VENT group (Table 15.29) is used to prescribe planes adjacent to obstructions or external walls. Note that the label VENT is used for historical reasons – this group of parameters has evolved well beyond its initial role as simply allowing for air to be blown into, or sucked out of, the computational domain.

7.4.1 Basics

The vents are chosen in a similar manner to the obstructions, with the sextuplet XB denoting a plane abutting a solid surface. Two of the six coordinates must be the same, denoting a plane as opposed to a solid.

Note that only one VENT may be specified for any given wall cell. If additional VENT lines are specified for a given wall cell, FDS will output a warning message and ignore the subsequent lines (i.e. only the first vent will be applied)

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

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

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

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 9.1 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 7.5 and Table 7.1. Also, the parameter OUTLINE=.TRUE. causes the VENT to be drawn as an outline in Smokeview.
7.4.2 Special VENTS

There are 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 6.4.5 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 11.2 for more information about gas species.

If you want to specify a non-ambient pressure at the OPEN boundary, see Section 9.3.

Starting with FDS 5.3.0, vents to the outside of the computational domain (OPEN vents) can be opened or closed during a simulation. It is best done by creating or removing a thin obstruction that covers the OPEN VENT. See Section 13.4.2 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.

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

7.4.3 Controlling VENTS

VENT functionality can be controlled in some cases using “devices” and “controls,” specified via a DEVC_ID or a CTRL_ID. See Section 13.4.2 for details.

7.4.4 Trouble-Shooting VENTS

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

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

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

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

The three RGB integer numbers range from 0 to 255, indicating the amount of Red, Green and Blue that make up the color. If you define the COLOR by name, it is important that you type the name EXACTLY as it is listed in the color tables here in this document and on the FDS website.

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

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

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

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

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

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

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

7.5.1 Texture Mapping

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

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

Assuming that a JPEG file called paneling.jpg exists in the working directory, Smokeview should read it and display the image wherever the paneling is used (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 / 54```
Table 7.1: Sample of Color Definitions (A complete list is included on the website)

<table>
<thead>
<tr>
<th>Name</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>Name</th>
<th>R</th>
<th>G</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>AQUAMARINE</td>
<td>127</td>
<td>255</td>
<td>212</td>
<td>MAROON</td>
<td>128</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ANTIQUE WHITE</td>
<td>250</td>
<td>235</td>
<td>215</td>
<td>MELON</td>
<td>227</td>
<td>168</td>
<td>105</td>
</tr>
<tr>
<td>BEIGE</td>
<td>245</td>
<td>245</td>
<td>220</td>
<td>MIDNIGHT BLUE</td>
<td>25</td>
<td>25</td>
<td>112</td>
</tr>
<tr>
<td>BLACK</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>MINT</td>
<td>189</td>
<td>252</td>
<td>201</td>
</tr>
<tr>
<td>BLUE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>NAVY</td>
<td>0</td>
<td>0</td>
<td>128</td>
</tr>
<tr>
<td>BLUE VIOLET</td>
<td>138</td>
<td>43</td>
<td>226</td>
<td>OLIVE</td>
<td>128</td>
<td>128</td>
<td>0</td>
</tr>
<tr>
<td>BRICK</td>
<td>156</td>
<td>102</td>
<td>31</td>
<td>OLIVE DRAB</td>
<td>107</td>
<td>142</td>
<td>35</td>
</tr>
<tr>
<td>BROWN</td>
<td>165</td>
<td>42</td>
<td>42</td>
<td>ORANGE</td>
<td>255</td>
<td>128</td>
<td>0</td>
</tr>
<tr>
<td>BURNT SIENNA</td>
<td>138</td>
<td>54</td>
<td>15</td>
<td>ORANGE RED</td>
<td>255</td>
<td>69</td>
<td>0</td>
</tr>
<tr>
<td>BURNT UMBER</td>
<td>138</td>
<td>51</td>
<td>36</td>
<td>ORCHID</td>
<td>218</td>
<td>112</td>
<td>214</td>
</tr>
<tr>
<td>CADET BLUE</td>
<td>95</td>
<td>158</td>
<td>160</td>
<td>PINK</td>
<td>255</td>
<td>192</td>
<td>203</td>
</tr>
<tr>
<td>CHOCOLATE</td>
<td>210</td>
<td>105</td>
<td>30</td>
<td>POWDER BLUE</td>
<td>176</td>
<td>224</td>
<td>230</td>
</tr>
<tr>
<td>COBALT</td>
<td>61</td>
<td>89</td>
<td>171</td>
<td>PURPLE</td>
<td>128</td>
<td>0</td>
<td>128</td>
</tr>
<tr>
<td>CORAL</td>
<td>255</td>
<td>127</td>
<td>80</td>
<td>RASPBERRY</td>
<td>135</td>
<td>38</td>
<td>87</td>
</tr>
<tr>
<td>CYAN</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>RED</td>
<td>255</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DIMGRAY</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>ROYAL BLUE</td>
<td>65</td>
<td>105</td>
<td>225</td>
</tr>
<tr>
<td>EMERALD GREEN</td>
<td>0</td>
<td>201</td>
<td>87</td>
<td>SALMON</td>
<td>250</td>
<td>128</td>
<td>114</td>
</tr>
<tr>
<td>FIREBRICK</td>
<td>178</td>
<td>34</td>
<td>34</td>
<td>SANDY BROWN</td>
<td>244</td>
<td>164</td>
<td>96</td>
</tr>
<tr>
<td>FLESH</td>
<td>255</td>
<td>125</td>
<td>64</td>
<td>SEA GREEN</td>
<td>84</td>
<td>255</td>
<td>159</td>
</tr>
<tr>
<td>FOREST GREEN</td>
<td>34</td>
<td>139</td>
<td>34</td>
<td>SEPIA</td>
<td>94</td>
<td>38</td>
<td>18</td>
</tr>
<tr>
<td>GOLD</td>
<td>255</td>
<td>215</td>
<td>0</td>
<td>SIENNA</td>
<td>160</td>
<td>82</td>
<td>45</td>
</tr>
<tr>
<td>GOLDENROD</td>
<td>218</td>
<td>165</td>
<td>32</td>
<td>SILVER</td>
<td>192</td>
<td>192</td>
<td>192</td>
</tr>
<tr>
<td>GRAY</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>SKY BLUE</td>
<td>135</td>
<td>206</td>
<td>235</td>
</tr>
<tr>
<td>GREEN</td>
<td>0</td>
<td>255</td>
<td>0</td>
<td>SLATEBLUE</td>
<td>106</td>
<td>90</td>
<td>205</td>
</tr>
<tr>
<td>GREEN YELLOW</td>
<td>173</td>
<td>255</td>
<td>47</td>
<td>SLATE GRAY</td>
<td>112</td>
<td>128</td>
<td>144</td>
</tr>
<tr>
<td>HONEYDEW</td>
<td>240</td>
<td>255</td>
<td>240</td>
<td>SPRING GREEN</td>
<td>0</td>
<td>255</td>
<td>127</td>
</tr>
<tr>
<td>HOT PINK</td>
<td>255</td>
<td>105</td>
<td>180</td>
<td>STEEL BLUE</td>
<td>70</td>
<td>130</td>
<td>180</td>
</tr>
<tr>
<td>INDIAN RED</td>
<td>205</td>
<td>92</td>
<td>92</td>
<td>TAN</td>
<td>210</td>
<td>180</td>
<td>140</td>
</tr>
<tr>
<td>INDIGO</td>
<td>75</td>
<td>0</td>
<td>130</td>
<td>TEAL</td>
<td>0</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>IVORY</td>
<td>255</td>
<td>255</td>
<td>240</td>
<td>THISTLE</td>
<td>216</td>
<td>191</td>
<td>216</td>
</tr>
<tr>
<td>IVORY BLACK</td>
<td>41</td>
<td>36</td>
<td>33</td>
<td>TOMATO</td>
<td>255</td>
<td>99</td>
<td>71</td>
</tr>
<tr>
<td>KELLY GREEN</td>
<td>0</td>
<td>128</td>
<td>0</td>
<td>TURQUOISE</td>
<td>64</td>
<td>224</td>
<td>208</td>
</tr>
<tr>
<td>KHAKI</td>
<td>240</td>
<td>230</td>
<td>140</td>
<td>VIOLET</td>
<td>238</td>
<td>130</td>
<td>238</td>
</tr>
<tr>
<td>LAVENDER</td>
<td>230</td>
<td>230</td>
<td>250</td>
<td>VIOLET RED</td>
<td>208</td>
<td>32</td>
<td>144</td>
</tr>
<tr>
<td>LIME GREEN</td>
<td>50</td>
<td>205</td>
<td>50</td>
<td>WHITE</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>MAGENTA</td>
<td>255</td>
<td>0</td>
<td>255</td>
<td>YELLOW</td>
<td>255</td>
<td>255</td>
<td>0</td>
</tr>
</tbody>
</table>
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.
Chapter 8

Fire and Thermal Boundary Conditions

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

8.1 Basics

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

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

&SURF ID = 'BRICK WALL'
  MATL_ID = 'BRICK'
  COLOR = 'RED'
  BACKING = 'EXPOSED'
  THICKNESS = 0.20 /

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

define a brick wall that is 4.9 m long, 1 m high, and 20 cm thick.
The thickness of the wall indicated by the OBST line need not match that indicated by the SURF line. The thickness of the material on the surface of the wall is dictated by the parameter THICKNESS. These two parameters are independent for each other, the OBST line describes the overall geometric structure, the SURF line describes the characteristics of the surfaces of the geometry which includes the thickness of the layers of materials applied to that surface.
8.2 Surface Temperature and Heat Flux

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

8.2.1 Specified Solid Surface Temperature

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

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

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

8.2.2 Special Topic: Convective Heat Transfer Options

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

Default Convective Heat Transfer Model

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

\[
\dot{q}^c = h \Delta T \quad \text{W/m}^2 \quad ; \quad h = \max \left[ C_1 |\Delta T|^{\frac{1}{4}}, \frac{k}{L} C_2 \frac{\text{Re}}{\text{Pr}}^{\frac{1}{4}} \right] \quad \text{W/m}^2/\text{K} \quad (8.1)
\]

where \( \Delta T \) is the difference between the wall and the gas temperature, \( C_1 \) is the coefficient for natural convection (1.52 for a horizontal surface and 1.31 for a vertical surface, by default) and \( C_2 \) is the coefficient for forced convection (0.037 by default, see [7] Eq. (5-85)); \( L \) is a characteristic length related to the size of the physical obstruction; \( k \) is the thermal conductivity of the gas, and the Reynolds Re and Prandtl Pr numbers are based on the gas flowing past the obstruction. Since the Reynolds number is proportional to the characteristic length, \( L \), the heat transfer coefficient is weakly related to \( L \) (for high Re, \( h \sim L^{-1/5} \)). For this reason, \( L \) is taken to be 1 m for most calculations. You can change the empirical coefficients using C_HORIZONTAL or C_VERTICAL for \( C_1 \) and C_FORCED for \( C_2 \), all of which are input on the MISC line.

Changing the Convective Heat Transfer Coefficient

If you want to change the default convective heat transfer coefficient, you can set to a constant using H_FIXED on the SURF line in units of W/m²/K. Another option is to use a turbulent convective heat transfer model suggested by Moinuddin and Li of Victoria University, Australia, by setting H_EDDY=.TRUE. on the MISC line [8].
Specifying the Heat Flux at a Solid Surface

Instead of altering the convective heat transfer coefficient, you may specify a fixed heat flux directly. Two methods are available to do this. The first is to specify a NET_HEAT_FLUX in units of kW/m². When this is specified FDS will compute the surface temperature required to ensure that the combined radiative and convective heat flux from the surface is equal to the NET_HEAT_FLUX. The second method is to specify separately the CONVECTIVE_HEAT_FLUX, in units of kW/m², and the radiative heat flux. The radiative heat flux is specified by setting both TMP_FRONT and EMISSIVITY appropriately. Note that if you wish there to be only a convective heat flux from a surface, then the EMISSIVITY should be set to zero. If NET_HEAT_FLUX or CONVECTIVE_HEAT_FLUX is positive, the wall heats up the surrounding gases. If NET_HEAT_FLUX or CONVECTIVE_HEAT_FLUX is negative, the wall cools the surrounding gases.

8.2.3 Special Topic: Adiabatic Surfaces

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

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

Specified temperature or heat flux boundary conditions are easy to apply, but only of limited usefulness in real fire scenarios. In most cases, walls, ceilings and floors are made up of several layers of lining materials. The `MATL` namelist group is used to define the properties of the materials that make up boundary solid surfaces. A solid boundary can consist of multiple layers\(^1\) of different materials, and each layer can consist of multiple material components.

8.3.1 Structure of Solid Boundaries

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

```plaintext
&MATL_ID = 'INSULATOR'
CONDUCTIVITY = 0.041
SPECIFIC_HEAT = 2.09
DENSITY = 229. /
&SURF_ID = 'BRICK WALL'
MATL_ID = 'BRICK','INSULATOR'
COLOR = 'RED'
BACKING = 'EXPOSED'
THICKNESS = 0.20,0.10 /
```

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

When a `SURF` is applied to the face of an `OBST`, the first `MATL_ID` is at the face of the `OBST`, with the other `MATL_IDs` being applied in succession with the final `MATL_ID` being applied on the opposite face of the `OBST`. If in the example above, `BRICK WALL` was applied to the entire `OBST` using `SURF_ID`, then when doing a heat transfer calculation from the +x face to the -x face, FDS would consider the `OBST` to be `BRICK` followed by `INSULATOR` and the same for a heat transfer calculation from the -x face to the +x face. To avoid this the user would need to specify a second `SURF` that has the reverse `MATL_ID` and use `SURF_ID6` to apply the two `SURF` definitions to opposite faces of the `OBST`.

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

```plaintext
&MATL_ID = 'WATER'
CONDUCTIVITY = 0.60
SPECIFIC_HEAT = 4.19
DENSITY = 1000. /
&SURF_ID = 'BRICK WALL'
MATL_ID(1,1:2) = 'BRICK','WATER'
```

\(^1\)The maximum number of material layers is 20. The maximum number of material components is 20.
In this example, the first layer of material, Layer 1, is composed of a mixture of brick and water. This is given by the \texttt{MATL\_ID} array which specifies Component 1 of Layer 1 to be brick, and Component 2 of Layer 1 to be water. The mass fraction of each is specified via \texttt{MATL\_MASS\_FRACTION}. In this case, brick is 95\%, by mass, of Layer 1, and water is 5\%.

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

$$\rho = \left( \sum \frac{Y_i}{\rho_i} \right)^{-1} \quad (8.2)$$

where $Y_i$ are the material mass fractions and $\rho_i$ are the material bulk densities defined on the \texttt{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 cannot be pure brick. Instead there are voids (pores) that are filled with water. If the water is taken away, there is only about 1476 kg/m$^3$ of brick left. To have a density of 1600 kg/m$^3$ for a partially void wall, a higher density should be used for the pure brick.

### 8.3.2 Thermal Properties

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

\begin{verbatim}
&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 /
\end{verbatim}

Notice that with temperature-dependent quantities, the \texttt{RAMP} parameter T means Temperature, and F is the value of either the specific heat or conductivity. In this case, neither \texttt{CONDUCTIVITY} nor \texttt{SPECIFIC\_HEAT} is given on the \texttt{MATL} line, but rather the \texttt{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 \texttt{ABSORPTION\_COEFFICIENT} (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.
8.3.3 Back Side Boundary Conditions

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

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

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

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

8.3.4 Initial and Back Side Temperature

By default, the initial temperature of the solid material is set to ambient (TMPA on the MISC line). Use TMP_INNER on the SURF line to specify a different initial temperature. Also, the back side temperature boundary condition of a solid can be set using the parameter TMP_BACK on the SURF line. TMP_BACK is not the actual back side surface temperature, but rather the gas temperature to which the back side surface is exposed. This parameter has no meaning for surfaces with BACKING='EXPOSED' or BACKING='INSULATED'.

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

8.3.5 Walls with Different Materials Front and Back

If you apply the attribute BACKING='EXPOSED' on a SURF line that is applied to a zero or one-cell thick obstruction, FDS calculates the heat conduction through the entire THICKNESS and it uses the gas phase temperature and heat flux on the front and back sides for boundary conditions. A redundant calculation is performed on the opposite side of the obstruction, so be careful how you specify multiple layers. If the layering is symmetric, the same SURF line can be applied to both sides. However, if the layering is not symmetric, you must create two separate SURF lines and apply one to each side. For example, a hollow box column that is made of steel and covered on the outside by a layer of insulation material and a layer of plastic on top of the insulation material, would have to be described with two SURF lines like the following:

&SURF ID = 'COLUMN EXTERIOR'
COLOR = 'ANTIQUE WHITE'
BACKING = 'EXPOSED'
MATL_ID(1:3,1) = 'PLASTIC','INSULATION','STEEL'
If, in addition, the insulation material and plastic are combustible, and their burning properties are specified on the appropriate MATL lines, then you need to indicate which side of the column would generate the fuel vapor. In this case, the steel is impermeable; thus you should add the parameter LAYER_DIVIDE=2.0 to the SURF line labeled ‘COLUMN EXTERIOR’ to indicate that fuel vapors formed by the heating of the two first layers (‘PLASTIC’ and ‘INSULATION’) are to be driven out of that surface. You need to also specify LAYER_DIVIDE=0.0 on the SURF line labeled ‘COLUMN INTERIOR’ to indicate that no fuel vapors are to driven into the interior of the column. In fact, values from 0.0 to 1.0 would work equally because the material ‘STEEL’ would not generate any fuel vapors.

By default, LAYER_DIVIDE is 0.5 times the number of layers for surfaces with EXPOSED backing, and equal to the number of layers for other surfaces.

### 8.3.6 Special Topic: Non-Planar Walls and Targets

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

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

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

### 8.3.7 Special Topic: Solid Phase Numerical Gridding Issues

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

The default parameters are usually appropriate for simple heat transfer calculations but sometimes the use of pyrolysis reactions makes the temperatures and burning rate fluctuate. 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.

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. When all the material of a shrinking surface is consumed but BURN_AWAY is not prescribed, the surface temperature is set to TMP_BACK, convective heat flux to zero and burning rate to zero.

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

FDS has several approaches for describing the pyrolysis of solids and liquids. The approach to take depends largely on the availability of material properties and the appropriateness of the underlying pyrolysis model. This section provides a description of the parameters that describe a burning solid material when the burning rate is known. In other words, you use these parameters to specify the burning rate.

8.4.1 A Gas Burner with a Specified Heat Release Rate

Solids and liquid fuels can be modeled by specifying their relevant properties via the MATL namelist group. However, if you simply want to specify a fire of a given heat release rate (HRR), you need not specify any material properties. A specified fire is basically modeled as the ejection of gaseous fuel from a solid surface or vent. This is essentially a burner, with a specified Heat Release Rate Per Unit Area, $HRR_{PUA}$, 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 10 to learn how to ramp the heat release rate up and down.

An alternative to $HRR_{PUA}$ with the exact same functionality is $MLR_{PUA}$, except this parameter specifies the Mass Loss Rate of fuel gas Per Unit Area in kg/m$^2$/s. Do not specify both $HRR_{PUA}$ and $MLR_{PUA}$ 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$_2$O from evaporated sprinkler droplets (see Section 11.2 for details). If a finite rate combustion model is desired instead of the default mixture fraction model, see Section 11.2.4.

Specifying $HRR_{PUA}$ or $MLR_{PUA}$ automatically invokes the mixture fraction combustion model.

8.4.2 Special Topic: A Radially-Spreading Fire

Sometimes it is desired that a fire spread radially at some specified rate. Rather than trying to design material properties to achieve this, you can alternatively use a VENT or a SURF line in a special way. First, you need to add a SURF line with a specified heat release rate, $HRR_{PUA}$, and an optional time history parameter, $RAMP_{Q}$ or $TAU_{Q}$. Then, you can also specify $XYZ$ and $SPREAD_{RATE}$ on either the VENT or SURF line. The fire is directed to start at the point $XYZ$ at the time designated by $T_{BEGIN}$ and spread radially at a rate of $SPREAD_{RATE}$ (m/s). The optional ramp-up of the HRR begins at the time when the fire arrives at a given point. For example, the lines

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

create a rectangular patch at $z=0$ on which the fire starts at the point (1.5,4.0,0.0) and spreads outwards at a rate of 0.03 m/s. Each surface cell burns for 30 s as the fire spreads outward, creating a widening ring of fire. Note that the $RAMP_{Q}$ is used in this construct to turn the burning on and off to simulate the consumption
of fuel as the fire spreads radially. It should not be used to mimic the “t-squared” curve – the whole point of the exercise is to mimic this curve in a more natural way. Eventually, the fire goes out as the ring grows past the boundary of the rectangle. Some trial and error is probably required to find the \textit{SPREAD\_RATE} that leads to a desired time history of the heat release rate.

If the starting time of the simulation, \textit{T\_BEGIN}, is not zero, be aware that the default start time of the radially spreading fire is \textit{T\_BEGIN}, not zero. This is also true of \textit{TAU\_Q}, but it is not true of \textit{RAMP\_Q}. Because this might be confusing, if you start the calculation at a time other than zero, do a quick test to ensure that the ramps or fire spread behave as expected.

### 8.4.3 Solid Fuels that Burn at a Specified Rate

Real objects, like furnishings, office equipment, and so on, are often difficult to describe via the \textit{SURF} and \textit{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:

```plaintext
&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 ‘\textit{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 \textit{T} in the \textit{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, \textit{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.

```plaintext
&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 /
```

67
The parameters HRRPUA, IGNITION_TEMPERATURE, and HEAT_OF_VAPORIZATION are all telling FDS that you want to control the burning rate yourself, but you still want to simulate the heating up and “ignition” of the fuel. When these parameters appear on the SURF line, they are acting in concert. If HRRPUA appears alone, the surface will begin burning at the start of the simulation, like a piloted burner. The addition of an IGNITION_TEMPERATURE delays burning until your specified temperature is reached. The addition of HEAT_OF_VAPORIZATION tells FDS to account for the energy used to vaporize the fuel. For any of these options, if a MATL line is invoked by a SURF line containing a specified HRRPUA, then that MATL ought to have only thermal properties. It should have no reaction parameters, product yields, and so on, like those described in the previous sections. By specifying HRRPUA, you are controlling the burning rate rather than letting the material pyrolyze based on the conditions of the surrounding environment.

8.4.4 Solid Fuels that do NOT Burn at a Specified Rate

This section describes the parameters that describe the reactions that occur within solid materials when they are burning. It is strongly recommended before reading this section that you read some background material on solid phase pyrolysis, for example “Thermal Decomposition of Polymers,” by Hirschler and Morgan, or “Flaming Ignition of Solid Fuels,” by Torero, both of which are in the 4th edition of the SFPE Handbook of Fire Protection Engineering.

The Reaction Mechanism

A solid surface in FDS can consist of multiple layers with multiple material components per layer. The material components are described via MATL lines and are specified on the SURF line that describes the structure of the solid. Each MATL can undergo several reactions that may occur at different temperatures. It may not undergo any – it may 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 experimental evidence of multiple reactions does not imply that a single material is undergoing multiple reactions, but rather that multiple material components are undergoing individual reactions at distinct temperatures. Currently, the maximum number of reactions for each material is 10 and the chain of consecutive reactions may contain up to 20 steps.

For a given MATL, the jth reaction can produce a (single) solid whose name is RESIDUE(j), plus 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 in the neighborhood of 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, all as part of the single reaction step. This information is conveyed to FDS via the yields: NU_RESIDUE(j), NU_WATER(j), and NU_FUEL(j), respectively. The integer j indicates to which reaction the parameters apply. 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. A more general way to specify gaseous yields is to use the parameter NU_GAS, as explained in Section 11.2.3.
The Reaction Rates

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

$$\frac{\partial Y_{s,i}}{\partial t} = -\sum_{j=1}^{N_{s}} r_{ij} + \sum_{j=1}^{N_{s}} \sum_{j'=1}^{N_{s}} v_{s,j} r_{ij} \quad (j' \neq i) \quad ; \quad r_{ij} = A_{ij} \rho_{s,i}^{n_{s,ij}} \exp \left( -\frac{E_{ij}}{RT_{s}} \right) \quad ; \quad Y_{s,i} = \left( \frac{\rho_{s,i}}{\rho_{s,0}} \right) \quad (8.3)$$

The term, $r_{ij}$, defines the rate of reaction at the temperature, $T_{s}$, of the $i$th material undergoing its $j$th reaction. The second term on the right of the equation represents the contributions of other materials producing the $i$th material as a residue with a yield of $v_{s,j}$. This term is denoted by $\text{NU_RESIDUE} \ (j)$ on the $i'$-th $\text{MATL}$ line. $\rho_{s,i}$ is the density of the $i$th material component of the layer, defined as the mass of the $i$th material component divided by the volume of the layer. $\rho_{s,0}$ is the initial density of the layer. Thus, $Y_{s,i} = \rho_{s,i}/\rho_{s,0}$ is a quantity that increases if the $i$th material component is produced as a residue of some other reaction, or decreases if the $i$th component decomposes. If the layer is composed of only one material, then $\rho_{s,i}/\rho_{s,0}$ is initially 1. $n_{s,ij}$ is the reaction order and prescribed under the name $\text{N\_S} \ (j)$, and is 1 by default. If the value of $n_{i}$ 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 $\text{A} \ (j)$ on the $\text{MATL}$ line of the $i$th material, with units of $s^{-1}$. $E_{ij}$, the activation energy, is prescribed via $\text{E} \ (j)$ in units of kJ/kmol. Remember that 1 kcal is 4.184 kJ, and be careful with factors of 1000. For a given reaction, specify both $A$ and $E$, or neither. Do not specify only one of these two parameters. Typically, these parameters only have meaning when both are derived from a common set of experiments, like TGA (Thermo-Gravimetric Analysis).

It is very important to keep in mind that $A$ and $E$ are not available for most real materials. If $A$ and $E$ are not known, there are several parameters that can be used by FDS to derive effective values. The most important parameter to specify in place of $A$ and $E$ is the $\text{REFERENCE_TEMPERATURE}$ ($°C$). To understand this parameter, consider the plot shown in Fig. 8.1. These curves represent the results of a hypothetical TGA experiment. The Mass Fraction (blue curve) is the normalized density of the material ($Y$) which decreases as the sample is slowly heated, in this case at a rate of 5 K/min. The Reaction Rate (green curve) is the rate of change of the mass fraction as a function of time ($-dY/dt$). Where this curve peaks is referred to in FDS as the $\text{REFERENCE_TEMPERATURE}$.\(^2\) Note that the $\text{REFERENCE_TEMPERATURE}$ is not the same as an ignition temperature, nor is it necessarily the surface temperature of the burning solid. Rather, it is simply the temperature at which the mass fraction of the material decreases at its maximum rate within the context of a TGA or similar experimental apparatus. The kinetic constants for the reaction are found from the formulae\(^3\):

$$E = \frac{e r_{p} \ R T_{p}^{2}}{Y_{0}} \quad ; \quad A = \frac{e r_{p} \ e^{E/RT_{p}}}{Y_{0}} \quad (8.4)$$

where $T_{p}$ and $r_{p}/Y_{0}$ are the reference temperature and rate, respectively. The $\text{REFERENCE_RATE}$ is the reaction rate, in units of $s^{-1}$, at the given $\text{REFERENCE_TEMPERATURE}$ divided by the mass fraction, $Y_{0}$, of material in the original sample undergoing the reaction. For a single component, single reaction material, $Y_{0} = 1$. The $\text{HEATING_RATE}$ ($T$) is the rate at which the temperature of the TGA (or equivalent) test apparatus was increased. It is input into FDS in units of K/min (in the formula, it is expressed in K/s). Its default value is 5 K/min. In Fig. 8.1, the area under the green curve (Reaction Rate) is equal to the heating rate (in units of K/s).

There are many cases where it is only possible to estimate the $\text{REFERENCE_TEMPERATURE}$ ($T_{p}$) of a particular reaction because micro-scale calorimetry data is unavailable. In such cases, an additional parameter

---

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

\(^3\)These formulas have been derived from an analysis that considers a first-order reaction. When using the proposed method, do not specify non-unity value for the reaction order $\text{N\_S}$ on the $\text{MATL}$ line.
\[
\frac{dY}{dt} = -AY \exp\left(-\frac{E}{RT}\right) \quad Y(0) = 1
\]

\[T_p = 300 \, ^\circ C\]
\[r_p = 0.002 \, \text{s}^{-1}\]
\[\dot{T} = 5 \, \text{K/min}\]
\[\nu_s = 0\]

Figure 8.1: The blue curve represents the normalized mass, \(Y = \rho_s/\rho_{s0}\), of a solid material undergoing heating at a rate of 5 K/min. The green curve represents the reaction rate, \(-dY/dt\). The system of ordinary differential equations that describe the transformation is shown at right. Note that the parameters \(T_p\), \(r_p\), and \(\nu_s\) represent the “reference” temperature, reaction rate, and residue yield of the single reaction. From these parameters, values of \(A\) and \(E\) can be estimated using the formulae in (8.4).

can be specified along with \texttt{REFERENCE_TEMPERATURE} \((T_p)\) to help fine tune the shape of the reaction rate curve, assuming some sort of measurement or estimate has been made to indicate at what temperature, and over what temperature range, the reaction takes place. The \texttt{PYROLYSIS_RANGE} \((\Delta T)\) is the approximate width (in degrees Celsius or Kelvin) of the green curve, assuming its shape to be roughly triangular. Its default value is 80 \(^\circ C\). Using these input parameters, an estimate is made of the peak reaction rate, \(r_p\), with which \(E\), then \(A\), are calculated.

\[
\frac{r_p}{Y_0} = \frac{2T}{\Delta T}(1-\nu_s)
\]

(8.5)

The parameter, \(\nu_s\), is the yield of solid residue.

When in doubt about the values of these parameters, just specify the \texttt{REFERENCE_TEMPERATURE}. Note that FDS will automatically calculate \(A\) and \(E\) using the above formulae. Do not specify \(A\) and \(E\) if you specify \texttt{REFERENCE_TEMPERATURE}, and do not specify \texttt{PYROLYSIS_RANGE} if you specify \texttt{REFERENCE_RATE}.

For the material decomposition shown in Fig. 8.1, the \texttt{MATL} would have the form:

```matlab
&MATL ID = 'My Fuel'
...
HEAT_OF_COMBUSTION = ...
N_REACTIONS = 1
NU_FUEL(1) = 1.
NU_RESIDUE(1) = 0.
REFERENCE_TEMPERATURE(1) = 300.
REFERENCE_RATE(1) = 0.002
HEATING_RATE(1) = 5.
HEAT_OF_REACTION(1) = ...
```

Note that the argument \((1)\) has been added to the reaction parameters to emphasize the fact that these parameters are stored in arrays of length equal to \texttt{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 \texttt{HEAT_OF_COMBUSTION} is the energy released per unit mass of fuel gas that mixes with oxygen and combusts. This has nothing to do with the
pyrolysis process, so why is it specified here? The answer is that there can be only one *gas phase* reaction of fuel and oxygen in FDS, but there can be dozens of different materials and dozens of *solid phase* reactions. To ensure that the fuel vapors from different materials combust to produce the proper amount of energy, it is very important to specify a **HEAT_OF_COMBUSTION** for each material. That way, the mass loss rate of fuel gases is automatically adjusted so that the effective mass loss rate multiplied by the single, global, gas phase heat of combustion produces the expected heat release rate. If, for example, the **HEAT_OF_COMBUSTION** specified on the `REAC` line is twice that specified on the `MATL` line, the mass of contained within wall cell will be decremented by that determined by the pyrolysis model, but the mass added to gas phase would be reduced by 50 %. A different value of heat of combustion can be specified for each reaction and each gaseous species using the notation **HEAT_OF_COMBUSTION**(j, s), as explained in Section 11.2.3.

Note that versions of FDS from 5.0 through 5.3 used a slightly different definition of **REFERENCE_TEMPERATURE** and **REFERENCE_RATE**. This will lead to different results when using the different versions of the model.

**Multiple Solid Phase Reactions**

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

The key input lines for this reaction are shown in Fig. 8.3. Note that the only parameters shown are those that describe the reaction mechanism, and that each of these parameters can be found either from visual inspection of the the mass loss (blue) curve or the reaction rate (green) curve. Even if TGA or similar data were unavailable in this case, you can still model the solid as a combination of water that evaporates at 100 °C and some other material that pyrolyzes in the vicinity of 300 °C, leaving 20 % of its mass as a residue. The full set of parameters for these cases are listed in `pyrolysis_1.fds` and `pyrolysis_2.fds`. Those interested in testing potential solid phase reaction mechanisms ought to use these test cases as templates.

**The Heat of Reaction**

Equation (8.3) 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.
Figure 8.2: The blue curve represents the combined mass fraction, $\sum Y_i$, and the green curve the net reaction rate, $-d/dt(\sum Y_i)$, for a material that contains 10% water (by mass) that evaporates at a temperature of 100°C, and 90% solid material that pyrolyzes at 300°C, leaving a 20% (by mass) residue behind. Note that the numbered subscripts refer to the material component and the reaction, respectively. In this case, there are three material components, and the first two each undergo a single reaction. The third material component is formed as a residue of the reaction of the second material. The system of ordinary differential equations that governs the transformation of the materials is shown at right.

Special Topic: The “Threshold” Temperature

In FDS, the reaction rate expression in Eq. (8.3) includes an optional term:

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

(8.6)

$T_{thr,ij}$ is an optional “threshold” temperature that allows the definition of non-Arrhenius pyrolysis functions and ignition criteria, and is prescribed by $\text{THRESHOLD_TEMPERATURE}(j)$. By default, $T_{thr,ij}$ is -273.15 degrees Celsius, $n_{t,j}$ is zero; thus, the last term of Equation 8.6 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 $\text{N_T}(j)$. 

---

$$\frac{dY_1}{dt} = -A_{1,1} Y_1 \exp(-E_{1,1}/RT) \quad Y_1(0) = 0.1$$
$$\frac{dY_2}{dt} = -A_{2,1} Y_2 \exp(-E_{2,1}/RT) \quad Y_2(0) = 0.9$$
$$\frac{dY_3}{dt} = -v_{s,2,1} \frac{dY_2}{dt} \quad Y_3(0) = 0.0$$

$T_{p,1,1} = 100 + 273 \text{ K} \quad T_{p,2,1} = 300 + 273 \text{ K}$
$r_{p,1,1} = 0.0016 \text{ s}^{-1} \quad r_{p,2,1} = 0.0012 \text{ s}^{-1}$
$v_{s,1,1} = 0 \quad v_{s,2,1} = 0.2$

$\dot{T} = 5 \text{ K/min}$

---

72
Figure 8.3: Input parameters for sample case pyrolysis_2.
8.4.5 Liquid Fuels

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

Figure 8.4 contains the key input parameters to describe a steel pan filled with a thin layer of ethanol. Note that the material properties are not all traceable to a measurement.

```plaintext
&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. /
```

Figure 8.4: Input parameters for sample case ethanol_pan.

The inclusion of BOILING_TEMPERATURE on the MATL line tells FDS to use its liquid pyrolysis model. It also automatically sets N_REACTIONS=1, that is, the only “reaction” is the phase change from liquid to gaseous fuel. Thus, HEAT_OF_REACTION in this case is the latent heat of vaporization. The gaseous fuel yield, NU_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.

The initial value of the fuel vapor volume flux can be specified using the parameter INITIAL_VAPOR_FLUX. Its default value is $5 \cdot 10^{-4} \text{ m}^3/(\text{sm}^2)$.

Note also the ABSORPTION_COEFFICIENT for the liquid. This denotes the absorption in depth of thermal radiation. Liquids do not just absorb radiation at the surface, but rather over a thin layer near the
surface. Its effect on the burning rate is significant.

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

### 8.4.6 Fuel Burnout

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

$$t_b = \frac{\rho_s \delta_s}{\dot{m}''}$$  \hspace{1cm} (8.7)

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

#### Solid Fuel Burnout

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

$$t_b = \frac{\rho_s \delta_s \Delta H}{\dot{q}_f'}$$  \hspace{1cm} (8.8)

Use the RAMP function to stop the burning once the calculated burnout time is reached. The burnout time of a reacting solid fuel is calculated automatically by FDS based on the layer THICKNESS, component DENSITY, and the calculated burning rate.

#### Liquid Fuel Burnout

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

### Special topic: Making Fuels Disappear (BURN_AWAY)

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

The following issues should be kept in mind when using BURN_AWAY:

- For reacting surfaces, the surface density is computed as a sum of the layer densities multiplied by the layer thicknesses. This value can be over-ridden by setting SURFACE_DENSITY on the SURF line. For surfaces with prescribed HRR (HRRPUA), SURFACE_DENSITY parameter is the only way of defining the mass of the object.

- For surfaces with prescribed HRR (HRRPUA) or prescribed mass loss rate (MLRPUA) AND thermally thick heat conduction, the mass flux is affected by the heats of combustion defined for the gas phase reaction and the first listed material (MATL) component.
• Use `BURN_AWAY` parameter cautiously. If an object has the potential of burning away, a significant amount of extra memory has to be set aside to store additional surface information as the rectangular block is eaten away.

• If `BURN_AWAY` is prescribed, the `SURF` should be applied to the entire object, not just a face of the object because it is unclear how to handle edges of solid obstructions that have different `SURF_IDS` on different faces.

• If the volume of the obstruction changes because it has to conform to the uniform mesh, FDS does not adjust the burning rate to account for this as it does with various quantities associated with areas, like `HRRPUA`.

• A parameter called `BULK_DENSITY` (kg/m\(^3\)) can be applied to the `OBST` rather than the `SURF` line. This parameter is used to determine the combustible mass of the solid object. The calculation uses the user-specified object dimensions, not those of the mesh-adjusted object. This parameter over-rides all other parameters from which a combustible mass would be calculated.

• The mass of the object is based on the densities of the all material components (`MATL`), but it is only consumed by mass fluxes of the known species. If the sum of the gaseous yields (Section 11.2.3) is less than one, it will take longer to consume the mass.

**Example Case: box_burn_away**

This is a silly example of a solid block of “foam” that is pyrolyzed until it is completely consumed. The heat flux is generated by placing hot surfaces around the box. There is no combustion. In the first example (box_burn_away1), the released gas is fuel (mixture fraction) and in the second example (box_burn_away2) it is an additional species called ‘GAS’. In the third and fourth examples (box_burn_away3 and 4), the released gas is fuel but the pyrolysis rate is specified (`HRRPUA`). In the fourth case, the heat of combustion for the foam material is set different from that of the gas, with ratio 0.75. The properties of the block of foam were chosen simply to assure a quick calculation. The objective of the test is to check that the released mass and the integrated burning rate is consistent with the material properties of the block. The block is 0.4 m on a side, with a density of 20 kg/m\(^3\). The integrated densities of the pyrolysis product gases (written to box_burn_away#_devc.csv), as well as the integrated burning rate (written to box_burn_away#_hrr.csv) in the end of the 30 s calculation ought to be:

\[
(0.4)^3 \times 20 \text{ kg/m}^3 = 1.28 \text{ kg} \quad (8.9)
\]

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

\[
0.75 \times 1.28 \text{ kg} = 0.96 \text{ kg} \quad (8.10)
\]
Figure 8.5: Output of box\_burn\_away test cases.
8.5 Testing Your Pyrolysis Model

Real materials that can burn can be very complicated. Undoubtedly, the \texttt{SURF} and \texttt{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 (\texttt{IJK=4, 4, 4}) for the pressure solver to function properly.

2. On the \texttt{TIME} line, set \texttt{WALL_INCREMENT=1} to force FDS to update the solid phase every time step (normally it does this every other time step), and set \texttt{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 \texttt{H\_FIXED=0.} on the \texttt{SURF} line. This turns off the convective heat flux from gas to surface and vice versa. The heat flux to the solid is specified via \texttt{EXTERNAL\_FLUX} (kW/m\textsuperscript{2}) on the \texttt{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 \texttt{ASSUMED\_GAS\_TEMPERATURE} on the \texttt{MISC} line, along with a non-zero value of \texttt{H\_FIXED} on \texttt{SURF} in units of W/m\textsuperscript{2}/K.

4. Turn off all the gas phase computations by setting \texttt{SOLID\_PHASE\_ONLY=.TRUE.} on the \texttt{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 \texttt{REAC} line with only \texttt{Y\_O2\_INFTY=0.01}. This sets the background oxygen mass fraction to 0.01, too low to support any burning.

5. Generate \texttt{MATL} lines, plus a single \texttt{SURF} line, as you normally would, except add \texttt{EXTERNAL\_FLUX} to the \texttt{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 \texttt{SURF\_ID} to a \texttt{VENT} that spans the bottom of the computational domain. Create \texttt{OPEN} vents on all other faces.

7. Finally, add solid phase output devices to the solid surface, like \texttt{’WALL\_TEMPERATURE’,’NET\_HEAT\_FLUX’,’BURNING\_RATE’,’GAUGE\_HEAT\_FLUX’, and ’WALL\_THICKNESS’} (assuming the solid is to burn away). Use these to track the condition of the solid as a function of time. In particular, make sure that the \texttt{’BURNING\_RATE’} is appropriate for the particular external heat flux applied. Make sure that the \texttt{’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.

Example Case: thermoplastic

The term “thermoplastic” is often used to describe materials that essentially heat up and gasify without leaving any solid residue. The term is used here merely to indicate the class of materials for which the pyrolysis can be modeled with a single reaction that converts solid to gaseous fuel.
The purpose of this example is to demonstrate how the solid phase pyrolysis model works in FDS. Essentially, the gas phase calculation is shut off except for the imposition of a 50 kW/m² “external” heat flux. The solid in this example is a slab of plastic, similar in composition to PMMA. The solid is described by the following SURF line:

```
&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 COLOR is meaningless except to distinguish it in Smokeview. The EXTERNAL_FLUX is a virtual heat source that is only used for these types of diagnostic exercises. The properties of the material are conveyed via the MATL line:

```
&MATL ID='PMMA'
  CONDUCTIVITY=0.21
  SPECIFIC_HEAT=1.5
  DENSITY=1190.
  N_REACTIONS=1
  NU_FUEL=1.
  HEAT_OF_REACTION=1500.
  HEAT_OF_COMBUSTION=25000.
  REFERENCE_TEMPERATURE=330. /
```

Notice that in addition to \( k \), \( \rho \) and \( c \), there is one reaction specified, the yield of which is 100 % fuel gas (NU_FUEL=1.). The phase change from solid to gas consumes energy at a rate of 1,500 kJ/kg. Although not relevant in this example, the burning of these fuel gases would produce energy at a rate of 25,000 kJ/kg. The reaction nominally takes place at 330 °C, a necessary parameter for the pyrolysis model described above.

The plots shown in Fig. 8.6 contain the results of the FDS simulation along with what is referred to as “simple theory.” By this, we mean that if all of the external heat flux were used to raise the solid temperature to 330 °C and then convert the solid to fuel gas, the results would be as shown by the “simple theory.” Of course, the simple theory neglects heat losses through the solid and losses via radiation from the surface. Because it neglects these losses, the simple theory should be regarded as producing an upper bound on the burning rate. Note that the small discontinuities in the FDS burning rate and temperature curves are numerical rather than physical.

Because there is no solid residue produced by the single reaction, the sample thickness will gradually decrease to zero. Note that this does not necessarily mean that the solid obstruction should disappear entirely from the computational domain, but rather that the fuel should be consumed. The parameter BURN_AWAY is used to indicate that the solid ought to be completely removed, but because in this simple test case the sample is aligned with the external boundary of the computational domain, the parameter BURN_AWAY would have no effect. The slab is originally 0.025 m thick, with a density of 1190 kg/m³. This means there is 29.75 kg/m² of fuel present, which if divided by the burning rate (about 0.018 kg/m²/s) yields a burning time of about 1700 s. After this, the fuel is consumed.
Example Case: charring_solid

This example just uses the solid phase algorithm. Essentially, the gas phase is shut off except for the imposition of a 50 kW/m² “external” heat flux. The reaction mechanism is fairly complicated, as it includes various solids like cellulose, char, and water. The sample itself is described via a single SURF line:

```plaintext
&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. /
```

The sample consists of two layers. The first is a combination of cellulose, water and lignin. There are MATL lines for each. Both cellulose and water have reaction parameters. Cellulose undergoes an endothermic reaction to form an “active” solid, and water evaporates. Lignin does not change form, at least not in this model. The second layer of the sample is a non-reacting slab of calcium silicate board. Note the two parameters on the SURF line called STRETCH_FACTOR and CELL_SIZE_FACTOR. The former tells FDS not to stretch the solid phase nodes used to solve the heat conduction equation. The latter tells FDS to use cells half the size of what it would use by default, based on the thermal properties. The intent of these
two parameters is to increase the spatial resolution of the solid phase mesh to increase the accuracy of the calculation. These kinds of parameters are typically not specified by the user, but a sensitivity analysis might indicate that they ought to be, especially when the reaction sequence is complicated as it is here.

Upon conversion of the virgin cellulose to its “active” component, this latter solid material undergoes two reactions that occur at two different temperatures. The following `MATL` line describes what is to happen:

```
&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'
```

Notice that the reaction parameters are arrays containing the appropriate information for each reaction, referred to by the numbers 1 and 2. The first reaction converts 35 % (by mass) of the “active” solid to char, and 65 % to fuel gases. The second reaction converts all the mass to fuel gases.

Figure 8.7 shows the surface temperature and burning rate of the sample under the 50 kW/m² external heat flux. The burning rate peaks at the start of the simulation, decreases throughout the burning phase, and then increases again at the end due to presence of an external backing material. The initial peak is typical of char-forming solids.

![Graph showing surface temperature and burning rate](image)

**Figure 8.7: Output of charring_solid test case.**

**Example Case: couch and room_fire**

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'
```
Both the fabric and the foam decompose into fuel gases via single-step reactions. The fuel gases from each have different composition and heats of combustion. FDS automatically adjusts the mass loss rate of each so that the “effective” fuel gas is that specified by the user on the REAC line. The attribute BURN_AWAY forces FDS to break up the couch into individual cell-sized blocks that will disappear from the calculation as soon as the fuel is exhausted. The surface is specified as consisting of two layers, with a thickness of 2 mm for the FABRIC and 10 cm for the FOAM. The 10 cm is chosen to be the same as the mesh cell size.

The same couch model is included in a room-scale fire simulation, known as the room_fire test case. Figure 8.8 shows the fire after 5 and 10 minutes, respectively. Note that after 5 minutes, the couch is fully-involved, and after 10 minutes the room has flashed over. Only the reaction zone of the fire is shown; the smoke is hidden so that you can see the fire progressing from the couch to the doorway at the right of the scene. This door is the only opening to the compartment, and after 10 minutes, the flames can be seen flowing out.
Figure 8.8: Output of **room_fire** test case showing fire after 5 and 10 minutes, respectively.
**Example Case: 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 \(^{\circ}\), 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.
This chapter mainly describes how to specify velocity boundary conditions. For fire applications, this is essentially how a ventilation system is modeled.

9.1 Simple Vents, Fans and Heaters

For most applications, the ventilation system of a building is described in FDS using velocity boundary conditions. For example, fresh air can be blown into, and smoke can be drawn from, a compartment by specifying a velocity in the normal direction to a solid surface. However, there are various other facets of velocity boundary conditions that are described below.

9.1.1 Supply and Exhaust Vents

The easiest way to describe a supply or exhaust fan is to create a VENT positioned on a solid surface with a SURF_ID with some form of specified velocity or volume flow rate. The normal component of velocity is usually specified directly via the parameter VEL. If VEL is negative, the flow is directed into the computational domain, i.e., a supply vent. If VEL is positive, the flow is drawn out of the domain, i.e., an exhaust vent. For example, the lines

&SURF ID='SUPPLY', VEL=-1.2, COLOR='BLUE' /
&VENT XB=5.0,5.0,1.0,1.4,2.0,2.4, SURF_ID='SUPPLY' /

create a VENT that supplies air at a velocity of 1.2 m/s through an area of nominally 0.16 m², depending on the realignment of the VENT onto the FDS mesh. Regardless of the orientation of the plane $x = 5$, the flow will be directed into the room because of the sign of VEL. In this example the VENT may not be exactly 0.16 m² in area because it may not align exactly with the computational mesh. If this is the case then VOLUME_FLUX can be prescribed instead of VEL. The units are m³/s. If the flow is entering the computational domain, VOLUME_FLUX should be a negative number, the same convention as for VEL. Note that a SURF with a VOLUME_FLUX prescribed can be invoked by either a VENT or an OBST, but be aware that in the latter case, the resulting velocity on the face or faces of the obstruction will be given by the specified VOLUME_FLUX divided by the area of that particular face. For example:

&SURF ID='SUPPLY', VOLUME_FLUX=-5.0, COLOR='GREEN' /
&OBST XB=..., SURF_ID6='BRICK','SUPPLY','BRICK','BRICK','BRICK','BRICK' /

dictates that the forward x-facing surface of the obstruction is to have a velocity equal to 5 m³/s divided by the area of the face (as approximated within FDS) flowing into the computational domain.
Note that either \texttt{VEL} or \texttt{VOLUME\_FLUX} should be prescribed, not both. The choice depends on whether an exact velocity is desired at a given vent, or whether the given volume flux is desired.

9.1.2 Heaters

You can create a simple heating vent by changing the temperature of the incoming air

\begin{verbatim}
&SURF ID='BLOWER', VEL=-1.2, TMP\_FRONT=50. /
\end{verbatim}

The \texttt{VENT} with \texttt{SURF\_ID='BLOWER'} would blow 50 °C air at 1.2 m/s into the flow domain. Making \texttt{VEL} positive would suck air out, in which case \texttt{TMP\_FRONT} would not be necessary.

Note that if \texttt{HRPPUA} or solid phase reaction parameters are specified, no velocity should be prescribed. The combustible gases are ejected at a velocity computed by FDS.

9.1.3 Total Mass Flux

Most often, you specify a simple supply or exhaust vent by setting either a normal velocity or volume flux at a solid surface. However, you may wish to control the mass flow rate (kg/s), as opposed to the volume flow rate (m$^3$/s), via the parameter \texttt{MASS\_FLUX\_TOTAL}. \texttt{MASS\_FLUX\_TOTAL} uses the same sign convention as \texttt{VEL} above. In fact, the value entered for \texttt{MASS\_FLUX\_TOTAL} is converted internally into a velocity boundary condition whose value for an outflow is adjusted based on the local density.

9.1.4 Louvered Vents

Most real supply vents are covered with some sort of grill or louvers which act to redirect, or diffuse, the incoming air stream. It is possible to mimic this effect, to some extent, by prescribing both a normal and the tangential components of the flow. The normal component is specified with \texttt{VEL} as described above. The tangential is prescribed via a pair of real numbers \texttt{VEL\_T} representing the desired tangential velocity components in the other two coordinate directions (\texttt{x} or \texttt{y} should precede \texttt{y} or \texttt{z}). For example, the line

\begin{verbatim}
&SURF ID='LOUVER', VEL=-1.2, VEL\_T=0.5,-0.3 /
\end{verbatim}

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

In cases of limited mesh resolution, it may not be possible to describe a louvered vent or slot diffuser using \texttt{VEL\_T} because there may not be enough mesh cells spanning the opening. In these cases, you might consider simply specifying a flat plate obstruction in front of the \texttt{VENT} with an offset of one mesh cell. The plate will simply redirect the air flow in all lateral directions.

9.1.5 Jet Fans

Fans do not have to be mounted on a solid wall, like a supply or an exhaust fan. If you just want to blow gases in a particular direction, create a thin (zero cells thick) \texttt{OBST}ruction and apply to it, via \texttt{SURF\_ID} only, a \texttt{SURF} line that has the parameter \texttt{POROUS=.TRUE.}, along with the other velocity parameters described above. This allows hot, smokey gases to pass through the obstruction, much like a free-standing fan. These obstructions are merely flat plates, by necessity. The velocity \texttt{VEL} associated with a \texttt{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.**

You may also want to construct a *shroud* around the fan using four flat plates arranged to form a short passageway that draws gases in one side and expels them out the other. The plate representing the fan itself can be positioned about halfway along the passage.

| A SURF with POROUS=.TRUE. can only be applied to an OBSTRUCTION. It has no meaning when applied to a VENT. |
9.2 Species and Species Mass Flux Boundary Conditions

There are two species boundary conditions that can be specified (see Section 11.2 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 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 `N`th 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 `N`th 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/m²/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)`.
9.3 Special Topic: Pressure Boundary Conditions

In some situations, it is more convenient to specify a pressure, rather than a velocity, at a boundary. Suppose, for example, that you are modeling the interior of a tunnel, and a wind is blowing at one of the portals that affects the overall flow within the tunnel. If (and only if) the portal is defined using an OPEN vent, then the dynamic pressure at the boundary can be specified like this:

```
&VENT XB=..., SURF_ID='OPEN', DYNAMIC_PRESSURE=2.4, PRESSURE_RAMP='wind' /
&RAMP ID='wind', T= 0.0, F=1.0 /
&RAMP ID='wind', T=30.0, F=0.5 /
```

The use of a dynamic pressure boundary affects the FDS algorithm as follows. At OPEN boundaries, the hydrodynamic pressure (head) $\mathcal{H}$ is specified as

\[
\mathcal{H} = \frac{\text{DYNAMIC\_PRESSURE}}{\rho_\infty} + \frac{|\mathbf{u}|^2}{2} \quad \text{(outgoing)}
\]

\[
\mathcal{H} = \frac{\text{DYNAMIC\_PRESSURE}}{\rho_\infty} \quad \text{(incoming)}
\]  

(9.1)

where $\rho_\infty$ is the ambient density and $\mathbf{u}$ is the most recent value of the velocity on the boundary. The PRESSURE_RAMP allows you to alter the pressure as a function of time. Note that you do not need to ramp the pressure up or down starting at zero, like you do for various other ramps. The net effect of a positive dynamic pressure at an otherwise quiescent boundary is to drive a flow into the domain. However, a fire-driven flow of sufficient strength can push back against this incoming flow.

Example Case: pressure_boundary

The following lines, taken from the sample case, pressure_boundary, demonstrates how to do a time-dependent pressure boundary at the end of a tunnel. The tunnel is 10 m long, 1 m wide, 1 m tall with a fire in the middle and a pressure boundary imposed on the right side. The left side (XMIN) is just an OPEN boundary with no pressure specified. It is assumed to be at ambient pressure.

```
&VENT MB = 'XMIN' SURF_ID = 'OPEN' /
&VENT MB = 'XMAX' SURF_ID = 'OPEN', DYNAMIC\_PRESSURE=2.4, PRESSURE\_RAMP='wind\_ramp' /
&RAMP ID='wind\_ramp', T= 0., F= 1. /
&RAMP ID='wind\_ramp', T=15., F= 1. /
&RAMP ID='wind\_ramp', T=16., F=-1. /
```

Figure 9.1 shows two snapshots from Smokeview taken before and after the time when the positive pressure is imposed at the right portal of a tunnel. The fire leans to the left because of the preferential flow in that direction. It leans back to the right when the positive pressure is directed to become negative.
Figure 9.1: Snapshots from the sample case `pressure_boundary` showing a fire in a tunnel leaning left, then right, due to a positive, then negative, pressure imposed at the right portal.
9.4 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 `PROFILE='ATMOSPHERIC'` produces a typical atmospheric wind profile of the form \( u = u_0 \left( \frac{z}{z_0} \right)^p \). If an atmospheric profile is prescribed, also prescribe \( z_0 \) for \( z_0 \) and \( PLE \) for \( p \). `VEL` specifies the reference velocity \( u_0 \). Note that \( z_0 \) is not the ground, but rather some height where the wind speed is measured, like an elevated weather station. It is assumed that the ground is located at \( z = 0 \). To change this assumption, set `GROUND_LEVEL` on the `MISC` line to be the appropriate value of \( z \). Be careful not to apply an atmospheric velocity profile below `GROUND_LEVEL` or FDS will stop with an error.

Another useful parameter for outdoor simulations is the temperature lapse rate of the atmosphere. Typically, in the first few hundred meters of the atmosphere, the temperature decreases several degrees Celsius per kilometer. These few degrees are important when considering the rise of smoke since the temperature of the smoke decreases rapidly as it rises. The `LAPSE_RATE` of the atmosphere can be specified on the `MISC` line in units of \(^\circ\text{C}/\text{m}\). A negative sign indicates that the temperature decreases with height. This need only be set for outdoor calculations where the height of the domain is tens or hundreds of meters. The default value of the `LAPSE_RATE` is 0 \(^\circ\text{C}/\text{m}\).

By default, FDS assumes that the density and pressure decrease with height, regardless of the application or domain size. For most simulations, this effect is negligible, but it can be turned off completely by setting `STRATIFICATION=.FALSE.` on the `MISC` line.
9.5 Special Topic: The HVAC Namelist Group

There are occasions where simply defining fixed flow and fixed species boundary conditions is not sufficient to model the behavior of an HVAC (Heating, Ventilation, Air Conditioning) system. If the ability to transport heat and combustion products through a duct network or the ability to fully account for the pressurization of a compartment due to a fire on the flows in a duct network is important, you can make use of a coupled HVAC network solver. The solver computes the flows through a duct network described as a mapping of duct segments and nodes where a node is either the joining of two or more ducts or where a duct segment connects to the FDS computational domain. The current HVAC solver does not allow for mass storage in the duct network (i.e. what goes in during a time step, goes out during a time step). HVAC components such as fans and binary dampers (fully open or fully closed) can be included in the HVAC network and coupled to the FDS control function capability. The user can select from three fan models.

The HVAC solver is invoked if there is an HVAC namelist group present in the input file. An HVAC network is defined by providing inputs for the ducts, duct nodes, and any fans or dampers present in the system. Additionally the user must define the locations where the HVAC network joins the computational domain. The basic syntax for an HVAC component is:

```
&HVAC TYPE_ID='componenttype', ID='componentname', ... /
```

where:

- **TYPE_ID** is a character string that indicates the type of component that the namelist group is defining. **TYPE_ID can be 'DUCT', 'NODE', or 'FAN'.**
- **ID** is a character string giving a name to the component. The name must be unique amongst all other components of that type.

The inputs for an HVAC duct are:

```
&HVAC TYPE_ID='DUCT', ID='ductname', NODE_ID='node 1','node 2', DIAMETER=2., AREA=3.14, LOSS=1.1., ROUGHNESS=0.001, VOLUME_FLOW=1., DAMPER=.FALSE., FAN='FAN_ID', REVERSE=.FALSE., DEVC_ID='DEVname', CTRL_ID='ctrlname' /
```

where:

- **AREA** is the area of the duct in m$^2$. Either AREA or DIAMETER must be defined.
- **CTRL_ID** is the ID of a CTRL function for a damper or fan in the duct. Only one of CTRL_ID or DEVC_ID can be defined.
- **DAMPER** is a logical variable indicating the presence of a damper in the duct. The state of the damper is controlled by function named by CTRL_ID or DEVC_ID. Only one of either DAMPER or FAN_ID can be specified for a single duct.
- **DEV_ID** is the ID of a DEV for a damper or fan in the duct. Only one of CTRL_ID or DEVC_ID can be defined.
- **DIAMETER** is the diameter (m) of the duct. Either AREA or DIAMETER must be defined.
- **FAN** is the ID of a fan located in the duct. Only one of either DAMPER or FAN_ID can be specified for a single duct.
LOSS is a pair of real numbers giving the forward and reverse flow loss in the duct.

NODE_ID gives the IDs of the nodes on either end of the duct segment. Positive velocity in a duct is defined as flow from the first node to second node.

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

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

VOLUME_FLOW defines a fixed volumetric flow rate (m$^3$/s) in the duct. VOLUME_FLOW cannot be used in conjunction with FAN_ID.

The inputs for an HVAC node are:

```
&HVAC TYPE_ID='NODE', ID='nodename', DUCT_ID='duct 1', 'duct 2',..'duct n', VENT_ID='vent',
LOSS_ARRAY=lossarray, AMBIENT=.FALSE., XYZ=x,y,z /
```

where:

AMBIENT is a logical value. If .TRUE., then the node is connected to the ambient (i.e. it is equivalent to the OPEN boundary condition on a SURF line).

DUCT_ID gives the IDs of the ducts connected to the node. Up to 10 ducts can be connected to a node.

LOSS is an n by n array of real numbers giving the flow losses for the node. LOSS(I,J) is the loss from duct I to duct J.

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

XYZ is a triplet of real numbers giving the coordinates of the node. This location is used to compute buoyancy heads. If the node is connected to the FDS domain, then do not specify XYZ. FDS will compute it as the centroid of the VENT.

The inputs for an HVAC fan are:

```
&HVAC TYPE='FAN', ID='fanname', RAMP_ID='rampname', VOLUME_FLOW=volumeflow,
MAX_FLOW=maxflow, MAX_PRESSURE=maxpressure, LOSS=offloss /
```

where:

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

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

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

RAMP_ID identifies the RAMP that contains a table of volumetric flow rates (m$^3$/s) versus the pressure drop across the fan (Pa). VOLUME_FLOW is the fixed volumetric flow of the fan (m$^3$/s).

Note that only one set of fan model inputs (VOLUME_FLOW, RAMP_ID, or MAX_FLOW + MAX_PRESSURE) should be specified.

Also note that FAN defines a class of fans rather than one specific fan. Therefore, more than one duct can reference a single FAN.
9.6 Tangential Velocity Boundary Conditions at Solid Surfaces

The no-slip condition implies that the continuum tangential gas velocity at a surface is zero. In turbulent flow the velocity increases rapidly through a boundary layer that is only a few millimeters thick to its “free-stream” value. In most practical simulations, it is not possible to resolve this boundary layer directly; thus, an empirical model is used to represent its effect on the overall flow field. For a DNS (Direct Numerical Simulation), the velocity gradient at the wall is computed directly from the resolved velocity near the wall (NO_SLIP=.TRUE. by default). For an LES (Large Eddy Simulation), the Werner-Wengle wall model is applied for smooth walls and a log law is applied for rough walls. The surface roughness (in meters) is set by ROUGHNESS on SURF. See the FDS Technical Reference Guide [6] for wall model details. To force a solid boundary to have a free-slip condition, set FREE_SLIP=.TRUE. on the SURF line. In LES, to override the wall model and force a no-slip boundary condition, set NO_SLIP=.TRUE. on SURF.

The parameter SLIP_FACTOR is no longer used to set the tangential velocity boundary condition, as of FDS version 5.4.0.
9.7 Pressure-Related Effects: The ZONE Namelist Group (Table 15.29)

The basic FDS equation set assumes pressure to be composed of a “background” component, \( p(z,t) \), plus a perturbation, \( \tilde{p}(x,t) \). Most often, \( p \) 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 assumed that the background pressure was the same throughout the computational domain. Because of this, it was only 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 compartments within the computational domain that can have their own “background” pressures, and these compartments, or “pressure zones,” can be connected via leakage and duct paths whose flow rates are tied to the pressure of the adjacent zones.

9.7.1 Specifying Pressure Zones

A pressure zone can by any region within the computational domain that is separated from the rest of the domain, or the exterior, by solid obstructions. There is currently no algorithm within FDS to identify these zones based solely on your specified obstructions. Consequently, it is necessary that you identify these zones explicitly in the input file. The basic syntax for a pressure ZONE is:

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

This means that the rectangular region, \( 0.3 < x < 1.2, 0.4 < y < 2.9, 0.3 < z < 4.5 \), is assumed to be within a sealed compartment. There can be multiple ZONEs declared. The indices of the zones, which are required for the specification of leaks and fans, are determined simply by the order in which they are specified in the input file. By default, the exterior of the computational domain is “Zone 0.” If there are no OPEN boundaries, the entire computational domain will be assumed to be “Zone 1.”

There are several restrictions to assigning pressure zones. First, the declared pressure zones must be completely within a region of the domain that is bordered by solid obstructions. If the sealed region is not rectangular, FDS will extend the specified ZONE boundaries to conform to the non-rectangular domain.\(^1\) It is possible to “break” pressure zones by removing obstructions between them.\(^2\) An example of how to break pressure zones is given in Section 9.3. Second, pressure zones can span multiple meshes, but it is recommended that you check the pressure in each mesh to ensure consistency. Also, if the ZONE does span multiple meshes, make sure that the specified rectangular coordinates do so as well. This allows FDS to determine the actual extent of the ZONE independently for each mesh.

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

\(^1\)The extension of pressure zones to non-rectangular regions is a feature that started with FDS version 5.3.1.

\(^2\)The ability to open pressure zones became available in FDS starting with version 5.3.0. Prior versions prevented it by issuing an error statement.
9.7.2 Leaks

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

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

(9.2)

where $\Delta p$ is the pressure difference between the adjacent compartments (in units of Pa) and $\rho_\infty$ 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 subgrid-scale phenomenon because the leakage area is usually very small. In other words, it is not possible to define a leak directly on the numerical mesh. It is sometimes possible to “lump” the leaks into a single mesh-resolvable hole, but this is problematic for two reasons. First, the leakage area rarely corresponds neatly to the area of a single mesh cell-sized hole. Second, the flow speeds through the hole can be large and cause numerical instabilities.

A better way to handle leakage is by exploiting pressure zones. A pressure zone is a user-specified volume within the computational domain that is entirely surrounded by solid obstructions. For example, the interior of a closed room can be, and should be, declared a pressure zone. Leakage from one compartment to another is then designated on the input lines defining the individual pressure zones:

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

The first line designates a region of the computational domain to be “Pressure Zone” 1. Note that the order of the ZONE lines is important; that is, the order implicitly defines Zone 1, Zone 2, etc. Zone 0 is by default the ambient pressure exterior. In this example, a leak exists between Zone 1 and the exterior Zone 0, and the area of the leak is 0.0001 m$^2$ (1 cm by 1 cm hole, for example). Zone 2 leaks to Zone 1 (and vice versa) with a leak area of 0.0002 m$^2$. Zone 2 also leaks to the outside with an area of 0.0003 m$^2$. Note that zones need not be connected for a leak to occur. 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, but these solids need not form a boundary between the two zones. The SURFaces with the LEAK_PATH attribute lump all of the leakage over these areas. The order of the two pressure zones designated by LEAK_PATH is unimportant.

Example Case: pressure_rise

This example tests several basic features of FDS. A narrow channel, 3 m long, 0.002 m wide, and 1 m tall, has air injected at a rate of 0.1 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. simply to remove the slight change in pressure and density with height. The domain is divided into three meshes, each 1 m long and each with identical gridding. We expect the pressure, temperature and density to rise during the 60 s injection period. Afterwards, the temperature, density, and pressure should remain constant, according to the equation of state. The figures below show the results of this calculation. The density matches exactly showing that FDS is injecting the appropriate amount of mass. The steady state values of the pressure, density and temperature are consistent with the ideal values.
Example Case: zone_break

In the example case called zone_break, three simple compartments are initially isolated from each other and from the ambient environment outside. Air is blown into compartment 1 at a constant rate for 5 s, increasing its pressure approximately 9500 Pa. At 10 s, compartment 1 is opened to compartment 2, decreasing the overall pressure in the two compartments to roughly one-third the original value because the volume of the combined pressure zone has been roughly tripled. At 15 s, the pressure is further decreased by opening a door to compartment 3, and, finally, at 20 s the pressure returns to ambient following the opening of a door to the outside. Figure 9.3 displays the pressure within each compartment.

Notice that the pressure within each compartment does not come to equilibrium instantaneously. Rather, a relaxation factor is applied by FDS to bring the zones into equilibrium over several seconds. This is done for several reasons. First, in reality doors and windows do not magically disappear as they do in FDS. It takes a finite amount of time to fully open them, and the slowing of the pressure increase/decrease is a simple way to simulate the effect. Second, relatively large pressure differences between zones wreak havoc with flow solvers, especially ones like FDS that use a low Mach number approximation. To maintain numerical stability, FDS gradually brings the pressures into equilibrium. This second point ought to be seen as a warning:
Do not use FDS to study the sudden rupture of pressure vessels! Its low Mach number formulation does not allow for high speed, compressible effects that are very important in such analyses. The zone breaking functionality described in this example is only intended to be used for relatively small pressure differences (<0.1 atm) between compartments. Real buildings cannot withstand substantially larger pressures anyway.

9.7.3 Fan Curves

In Section 9.1 there is a discussion of velocity boundary conditions, in which a fan is modeled simply as a solid boundary that blows or sucks air, regardless of the surrounding pressure field. In 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:

\[ V_{\text{fan}} = A_{\text{duct}} U_{\text{max}} \text{ sign}(\Delta p_{\text{max}} - \Delta p) \sqrt{\frac{\Delta p - \Delta p_{\text{max}}}{\Delta p_{\text{max}}}} \]  

(9.3)

The volume flow in the absence of a pressure difference is given by the area of the duct times the velocity of the air. \( A_{\text{duct}} \) is the area of the duct (m\(^2\)), and \( U_{\text{max}} \) is the air velocity (m/s) in the absence of a pressure difference. The pressure difference, \( \Delta p = p_1 - p_2 \), indicates the difference in pressure between the downstream compartment, or “zone,” and the upstream. The subscript 1 indicates downstream and 2 indicates upstream. The term, \( \Delta p_{\text{max}} \), is the maximum pressure difference the fan can operate upon, and it is assumed to be a positive number. Figure 9.4 displays a typical fan curve.

The velocity of the fan in the absence of a pressure difference, \( U_{\text{max}} \), is specified via the parameter \text{VEL} on the appropriate \text{SURF} line. Alternatively, the volume flow rate, \( A_{\text{duct}} U_{\text{max}} \), can be specified using \text{VOLUME\_FLUX}. Do not use both. These parameters were already introduced in Section 7.1. To simulate the behavior of a real fan, a few extra parameters need to be specified. To set \( \Delta p_{\text{max}} \), the maximum operating over-pressure, add \text{MAX\_PRESSURE} to the \text{SURF} line. Note that \text{MAX\_PRESSURE} should always be positive and in units of Pa. If the pressure difference (downstream minus upstream) exceeds the specified \text{MAX\_PRESSURE}, then there will be a backflow in the duct.
Figure 9.4: Fan curve corresponding to $VOLUME\_FLUX=10$ and $MAX\_PRESSURE=500$. Note that a volume flux greater than 10 is brought about by a negative pressure difference; that is, when the downstream pressure is less than the upstream. Note also that when the pressure difference is greater than 500 Pa, the volume flow becomes negative; that is, the flow reverses.

Note that the rules governing the sign of $VEL$ or $VOLUME\_FLUX$ remain in force for fans that are subject to pressure differences between compartments. Simply note that the sign of either $VEL$ or $VOLUME\_FLUX$ defines “upstream” and “downstream.” Thus, the zone into which air is blown in the absence of a pressure difference is the downstream zone.

Example Case: fan_test

Here is an example how fans can be specified. In it, two simple compartments share a common wall. Both compartments are considered as separate “pressure zones.” Two fans are mounted in the Partition Wall, blowing in opposite directions. The relevant input lines are:

```
&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. /
```

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

$$V_{fan} = A_{duct} U_{max} \ | \Delta p \max - \Delta p| \sqrt{\frac{|\Delta p - \Delta p \max|}{\Delta p \max}}$$

(9.4)
where $A_{\text{duct}}$ is the area of the duct (both are 0.04 m$^2$), $U_{\text{max}}$ is the air velocity (0.4 m/s from Zone 1 to Zone 2 and 0.2 m/s from Zone 2 to Zone 1), and $\Delta p_{\text{max}}$ is the maximum pressure difference the fan can operate upon (in this case both fans use 1000 Pa).

In steady state, the volume flow from compartment to compartment (or Zone to Zone) should be equal and opposite in sign.

$$
(0.04 \text{m}^2)(0.4 \text{ m/s}) \sqrt{|p_2 - p_1 - 1000 \text{ Pa}| \over 1000 \text{ Pa}} = (0.04 \text{m}^2)(0.2 \text{ m/s}) \sqrt{|p_1 - p_2 - 1000 \text{ Pa}| \over 1000 \text{ Pa}} \tag{9.5}
$$

The solution is $p_2 = 300 \text{ Pa}$ and $p_1 = -300 \text{ Pa}$ (see Fig. 9.5). Note that the sign of the Volume Flow in FDS has to do with whether the flow is moving in the plus or minus coordinate direction. This convention can make these types of calculations a bit tricky.

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 assigned to a solid wall. The `DUCT_PATH` defines the pressure `ZONE`s downstream and upstream of the fan, respectively. The fan represented by the the `SURF` line `ID='BLOW LEFT'`, for example, blows air into `ZONE` 1 and draws air out of `ZONE` 2 in the absence of a pressure difference between the compartments. The negative value of `VEL` indicates that the fan blows in the negative coordinate direction, into `ZONE` 1 and out of `ZONE` 2, and this is consistent with the order of the `ZONE`s listed by `DUCT_PATH`. In short, `DUCT_PATH` and `VEL` must be coordinated.

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.

**Example Cases: leak_test and leak_test_2**

In the following examples, both leaks and fans are demonstrated. 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”

\[
\dot{V}_{\text{fan}} = A_{\text{duct}} U_{\text{max}} \text{sign}(\Delta p_{\text{max}} - \Delta p) \sqrt{\frac{|\Delta p - \Delta p_{\text{max}}|}{\Delta p_{\text{max}}}}
\]  

(9.6)

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

\[
\dot{V}_{\text{leak}} = A_{\text{leak}} \sqrt{\frac{2\Delta p}{\rho_\infty}}
\]  

(9.7)

where \(A_{\text{leak}} = 0.0001 \text{ m}^2\) and \(\rho_\infty = 1.2 \text{ kg/m}^3\). 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 \(\Delta p\), 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 Fig. 9.6.

Figure 9.6: Output of the leak test cases.
Chapter 10

User-Specified Functions, Ramps and Tables

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). TABL 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).

10.1 Time-Dependent Functions

At the start of any calculation, the temperature is ambient everywhere, the flow velocity is zero everywhere, nothing is burning, and the mass fractions of all species are uniform. When the calculation starts temperatures, velocities, burning rates, etc., are ramped-up from their starting values because nothing can happen instantaneously. By default, everything is ramped-up to their prescribed values in roughly 1 s. However, control the rate at which things turn on, or turn off, by specifying time histories for the boundary conditions that are listed on a given SURF line. The above boundary conditions can be made time-dependent using either prescribed functions or user-defined functions. The parameters TAU_Q, TAU_T, and TAU_V indicate that the heat release rate (HRRPUA); surface temperature (TMP_FRONT); and/or normal velocity (VEL, VOLUME_FLUX), or MASS_FLUX_TOTAL are to ramp up to their prescribed values in TAU seconds and remain there. If TAU_Q is positive, then the heat release rate ramps up like \( \tanh\left(\frac{t}{\tau}\right) \). If negative, then the HRR ramps up like \( \left(\frac{t}{\tau}\right)^2 \). 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 TAU_s 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 /  
```

103
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. Note that each set of RAMP lines must have a unique ID and that the lines must be listed with monotonically increasing $T$. Note also that the TAU's 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_T or RAMP_T to control the ramp-ups for surface temperature. The surface temperature will be computed as

$$T_W(t) = T_{AMB} + f(t) (T_{FRONT} - T_{AMB})$$

(10.1)

where $T_W(t)$ is the surface temperature to be applied, $f(t)$ is the result of evaluating the RAMP_T at time $t$, $T_{AMB}$ is the input specified on the MISC line, and $T_{FRONT}$ is the input specified on the SURF line that the RAMP_T is for.

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

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Group</th>
<th>Input Parameter(s)</th>
<th>Time (s)</th>
<th>RAMP_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heat Release Rate</td>
<td>SURF</td>
<td>HRRPUA</td>
<td>TAU_Q</td>
<td>RAMP_Q</td>
</tr>
<tr>
<td>Temperature</td>
<td>SURF</td>
<td>TMP_FRONT</td>
<td>TAU_T</td>
<td>RAMP_T</td>
</tr>
<tr>
<td>Velocity</td>
<td>SURF</td>
<td>VEL, VOLUME_FLUX, MASS_FLUX_TOTAL</td>
<td>TAU_V</td>
<td>RAMP_V</td>
</tr>
<tr>
<td>Mass Fraction/Flux</td>
<td>SURF</td>
<td>MASS_FRACTION(N), MASS_FLUX(N)</td>
<td>TAU_MF(N)</td>
<td>RAMP_MF(N)</td>
</tr>
<tr>
<td>Particle Mass Flux</td>
<td>SURF</td>
<td>PARTICLE_MASS_FLUX</td>
<td>TAU_PART</td>
<td>RAMP_PART</td>
</tr>
<tr>
<td>External Heat Flux</td>
<td>SURF</td>
<td>EXTERNAL_FLUX</td>
<td>TAU_EF</td>
<td>RAMP_EF</td>
</tr>
<tr>
<td>Pressure</td>
<td>VENT</td>
<td>DYNAMIC_PRESSURE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow</td>
<td>PROP</td>
<td>FLOW_RATE</td>
<td>FLOW_TAU</td>
<td>FLOW_RAMP</td>
</tr>
<tr>
<td>Gravity</td>
<td>MISC</td>
<td>GVEC(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gravity</td>
<td>MISC</td>
<td>GVEC(2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gravity</td>
<td>MISC</td>
<td>GVEC(3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10.2 Temperature-Dependent Functions

Thermal properties like conductivity and specific heat can vary significantly with temperature. In such cases, use the RAMP function like this:

```
&MATL ID = 'STEEL'
FYI = 'A242 Steel'
SPECIFIC_HEAT_RAMP = 'c_steel'
CONDUCTIVITY_RAMP = 'k_steel'
DENSITY = 7850. /
&RAMP ID='c_steel', T= 20., F=0.45 /  
&RAMP ID='c_steel', T=377., F=0.60 /  
&RAMP ID='k_steel', T=677., F=0.85 /  
&RAMP ID='k_steel', T= 48., F=30. /  
&RAMP ID='k_steel', T=677., F=30. /  
```

Note that here (as opposed to time ramps) the parameter F is the actual physical quantity, not just a fraction of some other quantity. Thus, if CONDUCTIVITY_RAMP is used, there should be no value of CONDUCTIVITY given. Note also that for values of temperature, T, below and above the given range, FDS will assume a constant value equal to the first or last F specified.

10.3 Tabular Functions

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

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

A detailed description of the various table entries is given in the sections that describe quantities that use such tables. Currently, only sprinklers and nozzles use this group of parameters to define a complex spray pattern.
Note that each set of TABL lines must have a unique ID. Specific requirements on ordering the lines will depend upon the type of TABL and those requirements are provided in the appropriate section in this guide.
Chapter 11

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, and the combustion process is modeled as one or more finite-rate reactions of these species. 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.

11.1 Mixture Fraction Combustion: The REAC Namelist Group

There are two ways of modeling a fire. The first is to specify explicitly a Heat Release Rate Per Unit Area, HRRPUA, on a SURF line and then apply the SURF_ID to an obstruction or vent. This essentially creates a gas burner whose fuel flow rate you explicitly control. The other way to model a fire is to specify solid phase thermal and pyrolysis properties on one or more MATL lines, assemble these materials via a SURF line, and then apply this boundary condition to obstructions or vents. In this case the burning rate of the fuel depends on the net heat feedback to the surface. In both cases, however, the mixture fraction combustion model is used by default. In fact, the mere presence of these parameters automatically invokes the mixture fraction model\(^1\). Do not specify explicitly gas species like oxygen if you have also specified the heat release rate per unit area, HRRPUA, or solid phase reaction rates.

11.1.1 Basics

The stoichiometry of the gas phase combustion reaction or reactions is specified using the REAC namelist group. For the default mixture fraction model, only a single REAC line is needed. 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.

\(^1\)There is an exception to this rule. It is possible to use the finite-rate combustion model in conjunction with a pyrolysis model that generates the specified gaseous fuel species.
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:

\[ C_x H_y O_z N_v O_{other} + v O_2 \rightarrow r CO_2 + r H_2 O + r CO + r soot + r N_2 + r H_2 + r H_2 O + r CO + r H_2 + r N_2 + r soot + r O_{other} \]  
(11.1)

You need only specify the chemical formula of the fuel along with the yields of CO, soot, and H\text{2}, and the amount of hydrogen in the soot, H_{frac}. For completeness you can specify the N\text{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:

\[
\begin{align*}
    v_{O_2} &= v_{CO_2} + \frac{v_{CO}}{2} + \frac{v_{H_2O}}{2} - \frac{z}{2} \\
    v_{CO_2} &= x - v_{CO} - (1 - H_{frac})v_{soot} \\
    v_{H_2O} &= \frac{y}{2} - H_{frac}v_{soot} - v_{H_2} \\
    v_{CO} &= \frac{w_f}{W_{CO}}v_{CO} \\
    v_{H_2} &= \frac{w_f}{W_{H_2}}v_{H_2} \\
    v_{soot} &= \frac{w_s}{W_s}v_s \\
    v_{N_2} &= \frac{v}{2} \\
    v_{other} &= w \\
    W_s &= H_{frac} W_H + (1 - H_{frac}) W_C
\end{align*}
\]

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.

**FUEL** A character string that identifies fuel species for the reaction. When using the mixture fraction, specifying a fuel will cause FDS to use the thermophysical properties for that fuel when computing quantities such as specific heat or viscosity. This parameter is independent of the inputs for the fuel chemistry, i.e. C, H, O, N, OTHER. Table 11.1 provides a listing of the available species. By default, FDS uses the gas thermophysical properties of ETHYLENE for the fuel.

**ID** A character string that identifies the reaction. Normally, this label is not used by FDS, but it is useful to label the REAC line to more easily identify the fuel species.

**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\text{2}, 28 g/mol)

**Y_O2_INFTY** Ambient mass fraction of oxygen (Mixture Fraction only, default 0.232428)
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 $\Delta H$ (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

$$\Delta H \approx \frac{V_{O_2}W_{O_2}}{V_fW_f} \text{EPUMO2} \text{ kJ/kg}$$

(11.2)

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=.TRUE., then FDS will internally adjust $\Delta H$ to account for products of incomplete combustion. The default value is .FALSE.

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

&REAC ID = 'METHANE'
  C = 1. /

&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. /
11.1.2 Special Topic: Heat of Combustion

By default, the Energy Per Unit Mass of Oxygen, $EPUMO_2$, is used to compute the heat of combustion according to Eq. (11.2). Specifying the $HEAT_OF_COMBUSTION$ will override this 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 gas phase 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 fuel gases generated by the various burning objects in the building. Specify the stoichiometry of the predominant reaction via the $REAC$ namelist group. If the stoichiometry of the burning material differs from the global reaction, the $HEAT_OF_COMBUSTION$ is used to ensure that an equivalent amount of fuel is injected into the flow domain from the burning object.

11.1.3 Special Topic: Flame Extinction

Modeling suppression of a fire due to the introduction of a suppression agent like CO$_2$ or water mist, or due to the exhaustion of oxygen within a compartment is challenging because the relevant physical mechanisms occur at length scales smaller than a single mesh cell. Flames are extinguished due to lowered temperatures and dilution of the oxygen supply. A simple suppression algorithm has been implemented in FDS that attempts to gauge whether or not 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, or turn off suppression completely by setting SUPPRESSION=.FALSE. on the $MISC$ line. This latter approach saves on computing time because it prevents FDS from entering the suppression algorithm altogether.

Example Case: door_crack

This example uses the same simple compartment that was used to test leakage and fan curves. 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 rises due to the heat from the fire and the fan blowing air into the compartment:

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

(11.3)

where $\gamma \approx 1.4$, $\dot{Q} = 160,000 \text{ W}$, $V = 20.7 \text{ m}^3$, and $\dot{V} = 0.016 \text{ m}^3/\text{s}$. However, as heat is lost to the walls, and as the leak begins to relieve the pressure increase, the pressure rise slows, and then reverses at about 0.25 atm. In roughly 150 s, the fire is self-extinguished due to lack of oxygen. As the pressure rise decreases back to zero, and actually goes below zero, the fan, and the leak under the door, increase the oxygen supply, 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 0.25 atm without some sort of relief. The fan and the crack under the door obey simple formulae based on pressure differences, but these assumptions have limits.

• It is likely that the fire in this scenario would indeed extinguish itself as the oxygen volume fraction decreased below about 15 %. But, its re-ignition at the door crack and fan grill would depend on the presence of a spark or hot spot of some sort. FDS continues to flow fuel into the compartment past the point of local extinction, but the compartment cools. The default combustion algorithm in FDS assumes that in every grid cell there is a “virtual spark plug” that initiates combustion if fuel and oxygen are present.

11.1.4 Special Topic: 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 compared to the default two 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.

11.1.5 Special Topic: Turbulent Combustion

Unless you are performing a Direct Numerical Simulation (DNS), the reaction rate of fuel and oxygen is not based on the diffusion of fuel and oxygen at a well-resolved flame sheet. Instead, semi-empirical rules are
invoked by FDS to determine the rate of mixing of fuel and oxygen within a given mesh cell at a given time step. This section provides a brief explanation of these rules and the parameters that control them.

In an LES simulation, the heat release rate is computed as

\[ q''' = \min \left( q'''_{\text{max}}, \frac{\rho \min (Y_F, Y_{O2/s})}{\tau} \Delta H \right) ; \quad s = \frac{W_F}{v_{O2} W_{O2}} \]  

(11.4)

Here, \( \tau \) is a mixing time scale \([9]\) given by

\[ \tau = C (\delta x \delta y \delta z)^{\frac{2}{3}} D_{LES} \]  

(11.5)

The coefficient, \( C \), is taken as 0.1 in FDS by default, based on comparisons to various flame height correlations. It is given by \( C_{\text{EDC}} \) on the \texttt{REAC} line. If you do not want to use this model, set \texttt{EDDY_DISSIPATION=.FALSE.} on the \texttt{REAC} line, in which case \( \tau \) becomes the time step, \( \delta t \). There is an additional bound on the local heat release rate per unit volume, based on an empirical estimate of the average volumetric heat release rate of a fire. Orloff and De Ris \([10]\) suggest a value of 1200 kW/m\(^3\) for the entire fire. FDS uses by default a value of

\[ q'''_{\text{max}} = \frac{q''_{\text{max}}}{\delta x} + \frac{q'''_{\text{max}}}{\delta t} \text{ kW/m}^3 \]  

(11.6)

as a local upper bound. The term, \( q'''_{\text{max}} \), is the maximum heat release rate per unit area of flame sheet. It is specified via \texttt{HRRPUA_SHEET} on the \texttt{REAC} line. It is 0 kW/m\(^2\) by default for LES; 200 for DNS. The term, \( \frac{q'''_{\text{max}}}{\delta x} \), is the maximum heat release rate per unit volume. It is specified via \texttt{HRRPUV_AVERAGE} on the \texttt{REAC} line. It is 2500 kW/m\(^3\) by default for LES; 0 for DNS. Further discussion is found in the FDS Technical Reference Guide, Volume 1.
11.2 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 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\(_2\)O, 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\(^2\). 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.

11.2.1 Basics

The namelist group SPEC is used to specify each additional gas 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 11.1. Here is an example:

\[
\&SPEC \ ID='ARGON', MASS_FRACTION_0=0.1, MW=40. \ /
\]

Once the extra species has been declared, you introduce it at surfaces via the parameters MASS_FRACTION(N) or MASS_FLUX(N). The index \( N \) refers to the order in which the species is listed in the input file. Following is a very simple example of how a gas can be introduced into the simulation.

Sample Case: gas_filling

Consider the short input file:

\[
\&HEAD \ CHID='gas_filling', \ TITLE='Fill an Empty Room with Hydrogen' \ /
\&MESH \ IJK=32,32,15, XB=-3.2,3.2,-3.2,3.2,0.0,3.0 \ /
\&TIME \ T_END=300.0 \ /
\&SPEC \ ID='HYDROGEN' \ /
\&SURF \ ID='LEAK', MASS_FLUX(1)=0.01667, RAMP_MF(1)='leak_ramp' \ /
\&RAMP \ ID='leak_ramp', T= 0., F=0.0 \ /
\&RAMP \ ID='leak_ramp', T= 1., F=1.0 \ /
\&RAMP \ ID='leak_ramp', T=180., F=1.0 \ /
\&RAMP \ ID='leak_ramp', T=181., F=0.0 \ /
\&VENT \ XB=-0.6,0.4,-0.6,0.4,0.0,0.0, SURF_ID='LEAK', COLOR='RED' \ /
\&DUMP \ MASS_FILE=.TRUE. \ /
\&SLCF \ PBY=0.0, QUANTITY='HYDROGEN' \ /
\&TAIL \ /
\]

The case is nothing more than hydrogen gas filling a box. The gas is injected through a 1 m by 1 m vent at a rate of 0.01667 kg/m\(^2\)/s and shut off after 3 m. The total mass of hydrogen at that point ought to be 3 kg (see

\(^2\) 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.
Fig. 11.2). Notice that no properties were needed for the HYDROGEN because it is a species whose properties have already been programmed into FDS. The background species in this case is assumed to be air. The mass flow rate of the hydrogen is controlled via the ramping parameter \texttt{RAMP_MF(1)}, and the index 1 refers to the first, and only, gas species that is specified in the input file. The parameter \texttt{MASS_FILE=.TRUE.} instructs FDS to produce an output file that contains a time history of the hydrogen mass.

![Graph showing hydrogen mass vs. time](image)

**Figure 11.2:** Hydrogen mass vs. time for \texttt{gas\_filling} test case.

### 11.2.2 Special Topic: Gas Species Properties

Gases whose properties are hardwired in FDS are listed in Table 11.1. 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 11.1, its molecular weight $MW$ must be specified on the \texttt{SPEC} line in units of g/mol. In addition, if a DNS calculation is being performed, either the Lennard-Jones potential parameters $\sigma$ (\texttt{SIGMALJ}) and $\epsilon/k$ (\texttt{EPSILONKLJ}) should be specified; or the \texttt{VISCOSITY} (kg/m/s), \texttt{CONDUCTIVITY} (W/m/K), and \texttt{DIFFUSIVITY} (m$^2$/s) between the given species and the background species should be specified.

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 11.2.4) – 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 \texttt{BACKGROUND_SPECIES}, use the \texttt{MISC} line. If this species is not listed in Table 11.1, specify its molecular weight, $MW$, and (optionally) its \texttt{VISCOSITY}; \texttt{CONDUCTIVITY}; and \texttt{SPECIFIC_HEAT}, \texttt{SPECIFIC_ENTHALPY}, and \texttt{REFERENCE_TEMPERATURE}. If either \texttt{SPECIFIC_HEAT} or \texttt{ENTHALPY} is specified, then both must be specified. The \texttt{REFERENCE_TEMPERATURE} is the temperature that corresponds to the specified value of \texttt{SPECIFIC_ENTHALPY}. In the absence of any of these parameters, the appropriate values of ‘AIR’ are assumed. If the species listed in Table 11.1, is indicated as a liquid, that means that liquid thermo-physical properties do not need to be given for those species.

Recognized species that are emissive will been defined as \texttt{ABSORBING} and radiative absorption for those species will be computed. The keyword \texttt{ABSORBING} can be specified on the \texttt{SPEC} line as well. If \texttt{.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.
**Table 11.1: Optional Gas Species [11]**

<table>
<thead>
<tr>
<th>Species</th>
<th>Mol. Wgt. (g/mol)</th>
<th>σ (Å)</th>
<th>ε/k (K)</th>
<th>Liquid</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIR</td>
<td>29</td>
<td>3.711</td>
<td>78.6</td>
<td></td>
</tr>
<tr>
<td>ARGON</td>
<td>40</td>
<td>3.42</td>
<td>124.0</td>
<td></td>
</tr>
<tr>
<td>CARBON DIOXIDE</td>
<td>44</td>
<td>3.941</td>
<td>195.2</td>
<td></td>
</tr>
<tr>
<td>CARBON MONOXIDE</td>
<td>28</td>
<td>3.690</td>
<td>91.7</td>
<td></td>
</tr>
<tr>
<td>ETHANOL</td>
<td>46</td>
<td>4.530</td>
<td>362.6</td>
<td>Y</td>
</tr>
<tr>
<td>ETHYLENE</td>
<td>28</td>
<td>4.163</td>
<td>224.7</td>
<td></td>
</tr>
<tr>
<td>HELIUM</td>
<td>4</td>
<td>2.551</td>
<td>10.22</td>
<td></td>
</tr>
<tr>
<td>HYDROGEN</td>
<td>2</td>
<td>2.827</td>
<td>59.7</td>
<td></td>
</tr>
<tr>
<td>HYDROGEN BROMIDE</td>
<td>81</td>
<td>3.353</td>
<td>449.0</td>
<td></td>
</tr>
<tr>
<td>HYDROGEN CHLORIDE</td>
<td>36</td>
<td>3.339</td>
<td>344.7</td>
<td></td>
</tr>
<tr>
<td>HYDROGEN CYANIDE</td>
<td>26</td>
<td>3.63</td>
<td>569.1</td>
<td></td>
</tr>
<tr>
<td>HYDROGEN FLOURIDE</td>
<td>20</td>
<td>3.148</td>
<td>330.0</td>
<td></td>
</tr>
<tr>
<td>METHANE</td>
<td>16</td>
<td>3.758</td>
<td>148.6</td>
<td></td>
</tr>
<tr>
<td>METHANOL</td>
<td>32</td>
<td>3.626</td>
<td>481.8</td>
<td>Y</td>
</tr>
<tr>
<td>N-Hexane</td>
<td>86</td>
<td>4.524</td>
<td>199.41</td>
<td>Y</td>
</tr>
<tr>
<td>N-HEPTANE</td>
<td>100</td>
<td>4.701</td>
<td>205.75</td>
<td>Y</td>
</tr>
<tr>
<td>N-OCTANE</td>
<td>114</td>
<td>4.892</td>
<td>231.16</td>
<td></td>
</tr>
<tr>
<td>N-DECANE</td>
<td>142</td>
<td>5.233</td>
<td>226.46</td>
<td></td>
</tr>
<tr>
<td>NITROGEN</td>
<td>28</td>
<td>3.798</td>
<td>71.4</td>
<td></td>
</tr>
<tr>
<td>OXYGEN</td>
<td>32</td>
<td>3.467</td>
<td>106.7</td>
<td></td>
</tr>
<tr>
<td>PROPANE</td>
<td>44</td>
<td>5.118</td>
<td>237.1</td>
<td></td>
</tr>
<tr>
<td>TOLUENE</td>
<td>92</td>
<td>5.698</td>
<td>480.0</td>
<td></td>
</tr>
<tr>
<td>WATER VAPOR</td>
<td>18</td>
<td>2.641</td>
<td>809.1</td>
<td>Y</td>
</tr>
</tbody>
</table>

### 11.2.3 Special Topic: Yields of Gaseous Species (NU_GAS)

The yields of fuel and water gases are usually specified with the parameters NU_FUEL and NU_WATER on the MATL line. However, the yield of any explicitly-defined (via the SPEC line) gas species can be specified using the parameter NU_GAS(j,k), where j refers to the j’th reaction of the material and k refers to the k’th explicitly-defined gas species. NU_GAS can also be used to specify the yields of the mixture fraction and water vapor, assuming each is implicitly-defined and its order in the list of species is obtained from the output file CHID.out.

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

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

```verbatim
&SPEC ID = 'PYROLYZATE', MW=53.6 /
&SPEC ID = 'OXYGEN', MASS_FRACTION_0 = 0.23 /
```

115
11.2.4 Special Topic: 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 11.2 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 evaluate 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 oxidizer.

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. This means that NU(0) is not a valid input parameter.

N_S Array containing the exponents for the finite rate equation for each SPEC. Note that it is possible that a given SPEC can be assigned a value of N_S greater than zero and a value of NU equal to zero. In other words, the rate equation can be dependent on a species that does not participate directly in the reaction. Note also that the background species cannot participate in the reaction. This means that N_S(0) is not a valid input parameter.
HEAT_OF_COMBUSTION The effective heat of combustion the chemical reaction in units of kJ/kg. (Default 40,000 kJ/kg)
11.3 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. In calculations in which the mesh cells are on the order of a centimeter or larger, the temperature near the flame surface cannot be relied upon when computing the source term in the radiation transport equation, especially because of the $T^4$ dependence.

As a practical alternative, the parameter RADIATIVE_FRACTION on the RADI line allows you to specify explicitly the fraction of the total combustion energy that is released in the form of thermal radiation. Some of that energy may be reabsorbed elsewhere, yielding a net radiative loss from the fire or compartment that is less than the RADIATIVE_FRACTION, depending mainly on the size of the fire and the soot loading. If it is desired to use the radiation transport equation as is, then RADIATIVE_FRACTION ought to be set to zero, and the source term in the radiative transport equation is then based solely on the gas temperature and the chemical composition. By default, the RADIATIVE_FRACTION is 0.35 for an LES calculation, and zero for DNS.

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 [6]. Modifications to these models can be made via parameters on the RADI line. 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 called TIME_STEP_INCREMENT. The increment over which the angles are updated can be reduced from 5 with the integer called 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 when WIDE_BAND_MODEL=.TRUE., the ABSORPTION_COEFFICIENT output quantity becomes practically useless, because it then corresponds to one individual band of the spectrum.

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 11.1.2.

In simulations with no combustion nor radiating species, it is possible to use a constant absorption coefficient by specifying KAPPA0 on the RADI line.
Chapter 12

Particles and Droplets

Lagrangian particles\(^1\) are used in FDS to represent water or liquid fuel droplets, flow tracers, and various other objects that are not defined or confined by the numerical mesh. Sometimes the particles have mass, sometimes they do not. Some evaporate, absorb radiation, \textit{etc.} \texttt{PART} is the namelist group that is used to prescribe parameters associated with Lagrangian particles.

All Lagrangian particles must be explicitly defined via the \texttt{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 \texttt{WATER=.TRUE.} and \texttt{MASSLESS=.TRUE.}.

12.1 Basics

Properties of different types of Lagrangian particles are designated via the \texttt{PART} namelist group. Once a particular type of particle or droplet has been described using a \texttt{PART} line, then the name of that particle or droplet type is invoked elsewhere in the input file via the parameter \texttt{PART_ID}. There are no reserved \texttt{PART_ID}s – all must be defined. For example, an input file may have several \texttt{PART} lines that include the properties of different types of Lagrangian particles:

\begin{verbatim}
&PART ID='my smoke',... /
&PART ID='my water',... /
\end{verbatim}

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

\begin{verbatim}
&SURF ...,,PART_ID='my smoke' /
\end{verbatim}

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

\begin{verbatim}
&PROP ID='Acme Spk-123', QUANTITY='SPRINKLER LINK TEMPERATURE', PART_ID='my water', ... /
\end{verbatim}

\(^1\)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.
12.2 Particle and Droplet Insertion

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

12.2.1 Particles Introduced at a Solid Surface

If the particles have mass and are introduced from a solid surface, specify PARTICLE_MASS_FLUX on the SURF line. The number of particles inserted at each solid cell every DT_INSERT seconds is specified by NPPC on the SURF line defining the solid surface. The default value of 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/m²/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.

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.

Note also that you can independently control particles that emanate from a solid surface. For example, a device might control the activation of a fan, but you can over-ride the device and control the particles separately. To do this, specify either a device or controller via a DEVC_ID or CTRL_ID on the PART line that defines the particles. For more information on devices and controls, see Sections 13.4 and 13.5.

12.2.2 Droplets Introduced at a Sprinkler or Nozzle

DROPLETS_PER_SECOND is the number of droplets inserted every second per active sprinkler or nozzle (Default 5000). It is listed on the PROP line that includes other properties of the sprinkler or nozzle. Note that this parameter only affects sprinklers and nozzles. Changing this parameter does not change the flow rate, but rather the number of droplets used to represent the flow. Also note that the number of droplets introduced per “batch” is DROPLETS_PER_SECOND times DT_INSERT.

Note that DROPLETS_PER_SECOND can be a very important parameter. In some simulations, it is a good idea to increase this number so that the liquid mass is distributed more uniformly over the droplets. If this parameter is too small, it can lead to a non-physical evaporation pattern, sometimes even to the point of causing a numerical instability. If you encounter a numerical instability shortly after the activation of a sprinkler or nozzle, consider increasing DROPLETS_PER_SECOND to produce a smoother evaporation pattern that is more realistic. Keep in mind that for a real sprinkler or nozzle, there are many more droplets created per second than the number that can be simulated.
12.2.3 Particles or Droplets Introduced within a Volume

Sometimes it is convenient to introduce droplets or particles at the start of the simulation. For this purpose, add `NUMBER_INITIAL_DROPLETS` to the `INIT` line\(^2\) to indicate 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.

```plaintext
&PART ID='drops', DIAMETER=750., SAMPLING_FACTOR=1, COLOR='RED', EVAPORATE=.FALSE. / 
&INIT PART_ID='drops', XB=..., NUMBER_INITIAL_DROPLETS=1000, MASS_PER_VOLUME=3.5 / 
&INIT PART_ID='drops', XB=..., NUMBER_INITIAL_DROPLETS=2000, MASS_PER_VOLUME=2.5 / 
```

If you want to introduce droplets or particles within a given volume periodically in time and not just at the initial time, set `DT_INSERT` on the `INIT` line to a positive value indicating the time increment (s) for insertion. The parameter `NUMBER_INITIAL_DROPLETS` now indicates the number of droplets/particles inserted every `DT_INSERT` seconds. If the droplets/particles have mass, use `MASS_PER_TIME` (kg/s) instead of `MASS_PER_VOLUME` to indicate how much mass is to be introduced per second.

12.2.4 Controlling the Number of Particles and Droplets

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

- **DT_INSERT** Time increment in seconds between the introduction of a “batch” of particles or droplets. 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 0.01 s. Note that this parameter should be specified on the `SURF`, `PROP`, or `INIT` line, depending on whether the particles/droplets originate at a surface, a sprinkler or nozzle, or a volume. Versions prior to FDS 5.5 had this parameter listed on the `PART` line. Note that if you introduce particles or droplets via an `INIT` line, you must specify a value for `DT_INSERT`. Otherwise, particles or droplets will only be introduced initially.

- **SAMPLING_FACTOR** Sampling factor for the output file `CHID.prt5`. This parameter can be used to reduce the size of the particle output file used to animate the simulation. The default value is 1 for `MASSLESS` particles, meaning that every particle or droplet will be shown in Smokeview. The default is 10 for all other types of particles. `MASSLESS` particles are discussed in Section 12.4.

- **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.

12.3 Particle and Droplet Properties

Lagrangian particles are used to represent a wide variety of objects that cannot be explicitly resolved on the numerical mesh. As a result, there are a considerable number of parameters that define them, many of which may not be applicable to a particular type of particle or droplet.

---

\(^2\)Prior to FDS 5.5, it was possible to use the `PART` line to specify `NUMBER_INITIAL_DROPLETS`. 

121
12.3.1 Thermal Properties

For Lagrangian particles that are not MASSLESS, the following parameters should be included on the PART line to indicate how they heat up and possibly evaporate. It is assumed by default that non-massless particles are liquid droplets, but you can specify EVAPORATE=.FALSE. to change this.

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.

VAPORIZATION_TEMPERATURE Boiling temperature of liquid droplet.

MELTING_TEMPERATURE Melting (solidification) temperature of liquid droplet.

INITIAL_TEMPERATURE Initial temperature of liquid droplet.

HEAT_OF_VAPORIZATION Latent heat of vaporization of liquid droplet.

H_V_REFERENCE_TEMPERATURE The temperature corresponding to the provided HEAT_OF_VAPORIZATION.

Notice that the default DENSITY is that of water. If the drops are specified as WATER or the drops are specified as L FUEL and the FUEL species is shown as a liquid in Table 11.1, then only the DENSITY needs to be provided.

12.3.2 Size Distribution

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

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

(12.1)

Note that the parameter $\sigma$ is given the value $\sigma = 2/\sqrt{2\pi}(\ln 2) \gamma = 1.15/\gamma$ which ensures that the two functions are smoothly joined at $d = d_m$. You can also add a value for SIGMA_D to the PART line if you want to over-ride this feature. The larger the value of $\gamma$, 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.

12.3.3 Drag

For massive particles the default drag law (i.e., the drag coefficient correlation as a function of Reynolds number [based on particle diameter]) is that of a sphere. To invoke the cylinder drag law set DRAG_LAW='CYLINDER' on the PART line. If neither of these options is applicable, the user may specify a constant value of the drag coefficient for a particle class (a specific PART_ID) by setting a USER_DRAG_COEFFICIENT on PART. The USER_DRAG_COEFFICIENT trumps the DRAG_LAW.
12.3.4 Velocity on Solid Surfaces

When a droplet strikes a solid surface, it sticks and is reassigned a new speed and direction. If the surface is horizontal, the direction is randomly chosen. If vertical, the direction is downwards. The rate at which the droplets move over the horizontal and vertical surfaces is difficult to quantify. The parameters HORIZONTAL VELOCITY and VERTICAL VELOCITY on the PART line allow you to control the rate at which droplets move horizontally or vertically (downward). The defaults are 0.2 m/s and 0.5 m/s, respectively.

There are some applications, like the suppression of racked storage commodities, where it is useful to allow water droplets to move horizontally along the underside of a solid object. It is difficult to model this phenomenon precisely because it involves surface tension, surface porosity and absorption, and complicated geometry. However, a way to capture some of the effect is to set ALLOW UNDERSIDE DROPLETS=.TRUE. on the MISC line. It is normally false.

Be aware that when droplets hit obstructions, the vertical direction is assumed to coincide with the z axis, regardless of any change to the gravity vector, GVEC.

A useful sample case to demonstrate various features of droplet motion on solid obstructions is the test case called cascade.fds. Figure 12.1 shows a stream of water droplets impinging on the top of a box followed by the cascading of water droplets over the top edge.

Figure 12.1: Smokeview rendering of the cascade test case.

12.3.5 Color

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.

12.4 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 thermal properties because they are only to be used a flow tracers in Smokeview.

12.4.1 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 pointless since MASSLESS particles are for visualization only.

12.4.2 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 on the INIT line. The default value of STATIC is .FALSE.

12.4.3 Water Droplets

WATER=.TRUE. declares that the liquid droplets evaporate into ‘WATER VAPOR’, a separate gas 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 11.2), and then designate the appropriate SPEC_ID on the PART line.

12.4.4 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 to the PART line. 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. If a spray nozzle is used to generate the fuel droplets, its characteristics are specified in the same way as those for a sprinkler.

**Example Case: spray_burner**

Controlled fire experiments are often conducted using a spray burner, where a liquid fuel is sprayed out of a nozzle and ignited. In this example (spray_burner.fds), heptane from two nozzles is sprayed downwards into a steel pan. The flow rate is increased linearly so that the fire grows to 2 MW in 20 s, burns steadily for another 20 s, and then ramps down linearly in 20 s. The key input parameters are given here:

```plaintext
&DEV IC ID='nozzle_1', XYZ=4.0,-.3,0.5, PROP_ID='nozzle', QUANTITY='TIME', SETPOINT=0. /  
&DEV IC 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., DENSITY=688.,  
    QUANTITIES(1:2)='DIAMETER','DROPLET_TEMPERATURE',  
    DIAMETER=1000., HEAT_OF_COMBUSTION=44500., SAMPLING_FACTOR=1 /  
&PROP ID='nozzle', CLASS='NOZZLE', PART_ID='heptane droplets',  
    FLOW_RATE=1.96, FLOW_RAMP='fuel', 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. Note that a 2 MW fire is achieved via 2 nozzles flowing heptane at 1.96 L/min each:

\[
2 \times 1.96 \text{ L min}^{-1} \times \frac{1}{60} \text{ s} \times 688 \text{ kg m}^{-3} \times \frac{1}{1000} \text{ m}^3 \text{ L}^{-1} \times 44500 \text{ kJ kg}^{-1} = 2000 \text{ kW} \quad (12.2)
\]

The parameter HEAT_OF_COMBUSTION over-rides that for the overall reaction scheme. Thus, if other droplets or solid objects have different heats of combustion, the effective burning rates are adjusted so that the total heat release rate is that which the user expects. However, exercises like this ought to be conducted just to ensure that this is the case. The HRR curve for this example is given in Fig. 12.2.

Note that FUEL=.TRUE. automatically invokes a mixture fraction calculation in which fuel from the evaporating fuel droplets is burned according to the overall reaction scheme. Because the default mixture fraction combustion model assumes that fuel and oxygen burn when mixed (assuming that the oxygen concentration is above an empirically determined threshold), there is no need to specify an ignition source. For most liquid fuels, the small amount of evaporation at ambient temperature is enough to start the combustion process. In some sense, there is an implicit pilot flame.

Note also that this feature is subject to mesh dependence. If the mesh cells are too coarse, the evaporating fuel can be diluted to such a degree that it may not burn. Proper resolution depends on the type of fuel and the amount of fuel being ejected from the nozzle. Always test your burner at the resolution of your overall simulation.

### 12.4.5 Solid 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 govern its heating 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 [6]. However, for arbitrary particles or droplets, there is no assumed radiative absorption.

### 12.5 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 11.1.2 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 because this type of simulated fire essentially acts like a gas burner whose flow rate is explicitly specified. An empirical way to account for fire suppression by water is to characterize the reduction of the pyrolysis rate in terms of an exponential function. The local mass loss rate of the fuel is expressed in the form

$$\dot{m}''_f(t) = \dot{m}_{f,0}(t)e^{-\int k(t) dt} \quad (12.3)$$

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$^2$.

\[
k(t) = E\_COEFFICIENT m''_w(t) \text{ s}^{-1}
\]  
(12.4)

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.
Sprinklers, smoke detectors, heat flux gauges, and thermocouples may seem to be completely unrelated, but from the point of view of FDS, they are simply devices that operate in specific ways depending on the properties assigned to them. They can be used to record some quantity of the simulated environment, like a thermocouple, or they can represent a mathematical model of a complex sensor, like a smoke detector, and in some cases they can trigger events to happen, like a timer.

Versions of FDS prior to FDS 5 used device-specific namelist groups, like \textit{SPRK}, \textit{HEAT}, \textit{SMOD}, and \textit{THCP}, but the number and variety of fire-specific sensing and measurement devices continues to expand, and the data structures in FDS could not easily accommodate all possibilities. In addition, the logic associated with sensor activation and subsequent actions, like a vent opening, had become too complicated and prone to bugs. Starting in FDS 5, all devices, in the broadest sense of the word, are designated via the namelist group \textit{DEVC}. In addition, advanced functionality and properties are accommodated via additional namelists groups called \textit{CTRL} (Control) and \textit{PROP} (Properties).

### 13.1 Device Location and Orientation: The DEVC Namelist Group (Table 15.4)

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, \textit{XYZ}. The properties of the device are contained on the \textit{PROP} line designated by \textit{PROP_ID}, which will be explained below for each of the special devices included in FDS. The character string \textit{ID} is merely a descriptor to identify the device in the output files, and if any action is tied to its activation.

Not all devices need to be associated with a particular set of properties via the \textit{PROP_ID}. For example, pointwise output quantities are specified with a single \textit{DEVC} line, like

```
&DEVC ID='TC-23', XYZ=3.0,5.6,2.3, QUANTITY='TEMPERATURE' /
```

which tells FDS to record the temperature at the given point as a function of time. The \textit{ID} is a label in the output file whose name is \textit{CHID\_devc.csv}.

Some devices have a particular orientation which can be specified with various parameters; \textit{IOR}, \textit{ORIENTATION}, \textit{ROTATION}. \textit{IOR} or the Index of Orientation, is necessary for any device that is placed on the surface of a
solid. The values ±1 or ±2 or ±3 indicate the direction that the device “points”, where 1 is parallel to the x-axis, 2 is parallel to the y-axis and 3 is parallel to the z-axis.

**ORIENTATION** is used for devices that are not on a surface and require a directional specification, like a sprinkler. **ORIENTATION** is specified with a triplet of real number values that indicate the components of the direction vector. The default value of **ORIENTATION** is (0,0,-1). For example, a default downward-directed sprinkler spray can be redirected in other direction. If you were to specify

```plaintext
&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.

### 13.2 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.

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

All devices must have a specified **QUANTITY**. Some special devices (Section 13.3) have their **QUANTITY** specified on a **PROP** line. A **QUANTITY** specified on a **PROP** line associated with a **DEVC** line will override a **QUANTITY** specified on the **DEVC** line.
13.3 Special Devices and their Properties: The PROP Namelist Group (Table 15.19)

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.

13.3.1 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 $\sqrt{\text{m\cdot s}}$. (Default 100.)
- **C_FACTOR** in units of $\sqrt{\text{m/s}}$. (Default 0.)
- **ACTIVATION_TEMPERATURE** in units of °C. (Default 74 °C)
- **INITIAL_TEMPERATURE** of the link in units of °C. (Default TMPA)
- **FLOW_RATE** 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 0 m/s)

¹Prior to FDS 5, a separate file was used to store properties of a given sprinkler. This file is no longer used.
ORIFICE_DIAMETER  Diameter of the nozzle orifice in m (default 0 m). This input provides an alternative way to set droplet velocity by giving values for FLOW_RATE and ORIFICE_DIAMETER, in which case the droplet velocity is computed by dividing the flow rate by the orifice area. Use this method if you do not have any information about droplet velocity. However, quite often the user must fine-tune DROPLET_VELOCITY in order to reproduce certain spray profile. The ORIFICE_DIAMETER input is not used if either DROPLET_VELOCITY or SPRAY_PATTERN_TABLE is specified.

SPRAY_ANGLE  A pair of angles (in degrees) through which the droplets are sprayed. The angles outline a conical spray pattern relative to the south pole of the sphere centered at the sprinkler with radius OFFSET. For example, SPRAY_ANGLE=30.,80. directs the water droplets to leave the sprinkler through a band between 60° and 10° south latitude, assuming the orientation of the sprinkler is (0,0,-1), the default. 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 Chapter 12 for additional details.

PRESSURE_RAMP  The name of the RAMP lines specifying the dependence of pipe pressure on the number of active sprinklers and nozzles.

Be aware that sprinklers produce many droplets that need to be tracked in the calculation. To limit the computational cost, sprinkler droplets disappear when they hit the lower boundary of the computational domain, regardless of whether it is solid or not. To stop FDS from removing sprinkler droplets from the lower boundary of the computational domain, add the phrase POROUS_FLOOR=.FALSE. to the MISC (Section 6.4) line. Be aware, however, that droplets that land on the floor continue to move horizontally in randomly selected directions; bouncing off obstructions, and consuming CPU time.

For more information about sprinklers, their activation and spray dynamics, read the FDS Technical Reference Guide [6].

Special Topic: Specifying Complex Spray Patterns

If a more complex spray pattern is desired than can be achieved by using SPRAY_ANGLE, DROPLET_VELOCITY, and FLOW_RATE, then a SPRAY_PATTERN_TABLE can be specified using the TABL (Section 10) namelist group. For a spray pattern, specify the total flow using FLOW_RATE of the PROP line, the name of the spray pattern using SPRAY_PATTERN_TABLE and then one or more TABL lines of the format:

```
&TABL ID='table_id', TABLE_DATA=LAT1,LAT2,LON1,LON2,VELO,FRAC /
```

where each TABL line for a given ‘table_id’ provides information about the spherical distribution of the spray pattern for a specified solid angle. LAT1 and LAT2 are the bounds of the solid angle measured in degrees from the south pole (0 is the south pole and 90 is the equator, 180 is the north pole). Note that this is not the conventional way of specifying a latitude, but rather a convenient system based on the fact that a typical sprinkler sprays water downwards, which is why 0 degrees is assigned to the “south pole,” or the −z direction. The parameters LON1 and LON2 are the bounds of the solid angle (also in degrees), where 0 (or 360) is aligned with the −x axis and 90 is aligned with the −y axis. VELO is the velocity (m/s) of the droplets at their point of insertion. FRAC the fraction of the total flow rate of liquid that should emerge from that particular solid angle.

In the test case called bucket_test_2, the spray pattern is defined as two jets, each with a velocity of 10 m/s and a flow rate of 60 L/min. The first jet contains 0.2 of the total flow, the second, 0.8 of the total. The jets are centered at points 30° below the “equator,” and are separated by 180°.
**Special Topic: Varying Pipe Pressure**

In real sprinkler systems, the pipe pressure is affected by the number of actuated sprinklers. Typically, the pressure is higher than the design value when the first sprinkler activates, and decreases towards the design value and below that when more and more sprinklers are activated. The pipe pressure has an effect on flow rate, droplet velocity and droplet size distribution.

In FDS, the varying pipe pressure can be specified on a PROP line using PRESSURE_RAMP. On each RAMP line, the number of open sprinklers or nozzles is associated with certain pipe pressure (atm). For example:

```plaintext
&PROP ID='My nozzle'
  OFFSET=0.1
  PART_ID='water drops'
  FLOW_RATE=0.9
  OPERATING_PRESSURE = 10.0
  DROPLET_VELOCITY=15.0
  SPRAY_ANGLE=0.0,80.0
  PRESSURE_RAMP = 'PR1' /

&RAMP ID = 'PR1' T = 1, F = 16.0 /
&RAMP ID = 'PR1' T = 2, F = 10.0 /
&RAMP ID = 'PR1' T = 3, F = 8.0 /
```

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

When pressure ramps are used, both FLOW_RATE and DROPLET_VELOCITY are associated with the OPERATING_PRESSURE. Specify either the FLOW_RATE, or the K_FACTOR and OPERATING_PRESSURE. In the latter case, the flow rate is given by \( \dot{m}_w = K \sqrt{p} \) and droplet velocity by \( v = C \sqrt{p} \). If spray pattern table is used, the coefficient \( C \) is determined separately for each line of the table. The median diameter of the particle size distribution is scaled as \( d_m(p) = d_m(p_o) (p_o/p)^{1/3} \), where \( p_o \) is the OPERATING_PRESSURE and \( d_m(p_o) \) is specified by parameter DIAMETER on the corresponding PART line.

### 13.3.2 Nozzles

Nozzles are very much like sprinklers, only they do not activate based on the standard RTI model. To simulate a nozzle that activates at a given time, specify a QUANTITY and SETPOINT directly on the DEVC line. The following lines:
designate two nozzles of the same type, one which activates at time zero, the other at 5 s. Note that nozzles must have a designated PROP_ID, and the PROP line must have a designated PART_ID to describe the liquid droplets.

Example Case: flow_rate

This example demonstrates the use of pressure ramps and control logic for opening and closing nozzles. It also serves as a verification test for the water flow rate. There are four nozzles that open at designated times: 0 s, 15 s, 30 s and 45 s. At time 60 s, all the nozzles are closed. The number of open nozzles is measured using a device with quantity ‘OPEN NOZZLES’. A comparison of the FDS result and the exact, intended values is shown in left part of Figure 13.1.

The pressure ramp has been designed to deliver a total flow rate of 10 l/min at all values of open nozzles. Mathematically this means that

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

(13.1)

where $N_n$ is the number of open nozzles. The corresponding nozzle and pressure ramp definitions are

The accumulated water is tracked using a device measuring the accumulated mass per unit area, integrated over the total floor area. The total mass of accumulated water should increase from zero to 10 kg in 60 s. A comparison of the FDS prediction and this analytical result is shown in the right side of Figure 13.1. The small delay of the FDS result is caused by the time it takes from the droplets to fall down on the floor.

13.3.3 Heat Detectors

QUANTITY='LINK TEMPERATURE' defines a heat detector, which uses essentially the same activation algorithm as a sprinkler, without the water spray.
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).

### 13.3.4 Smoke Detectors

A smoke detector is defined in the input file with an entry similar to:

```plaintext
&DEVC ID='SD_29', PROP_ID='Acme Smoke Detector', XYZ=2.3,4.6,3.4 / 
&PROP ID='Acme Smoke Detector', QUANTITY='CHAMBER OBSCURATION', LENGTH=1.8, 
ACTIVATION_OBSCURATION=3.28 /
```

for the single parameter Heskestad model. Note that a PROP line is mandatory for a smoke detector, in which case the DEVC QUANTITY can be specified on the PROP line. For the four parameter Cleary model, use a PROP line like:

```plaintext
&PROP ID='Acme Smoke Detector I2',QUANTITY='CHAMBER OBSCURATION', 
ALPHA_E=1.8,BETA_E=-1.1,ALPHA_C=1.0,BETA_C=-0.8,ACTIVATION_OBSCURATION=3.28 /
```

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

$$\delta t_e = \alpha_e u^\beta_e \quad ; \quad \delta t_c = \alpha_c u^\beta_c$$

(13.2)

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 13.1. 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).

#### Defining Smoke

By default, FDS assumes that the smoke from a fire is generated in direct proportion to the heat release rate. A value of SOOT_YIELD=0.01 on the REAC line means that the smoke generation rate is 0.01 of the
Table 13.1: Suggested Values for Smoke Detector Model. See Ref. [12] for others.

<table>
<thead>
<tr>
<th>Detector</th>
<th>$\alpha_e$</th>
<th>$\beta_e$</th>
<th>$\alpha_c, L$</th>
<th>$\beta_c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cleary Ionization I1</td>
<td>2.5</td>
<td>-0.7</td>
<td>0.8</td>
<td>-0.9</td>
</tr>
<tr>
<td>Cleary Ionization I2</td>
<td>1.8</td>
<td>-1.1</td>
<td>1.0</td>
<td>-0.8</td>
</tr>
<tr>
<td>Cleary Photoelectric P1</td>
<td>1.8</td>
<td>-1.0</td>
<td>1.0</td>
<td>-0.8</td>
</tr>
<tr>
<td>Cleary Photoelectric P2</td>
<td>1.8</td>
<td>-0.8</td>
<td>0.8</td>
<td>-0.8</td>
</tr>
<tr>
<td>Heskestad Ionization</td>
<td>—</td>
<td>—</td>
<td>1.8</td>
<td>—</td>
</tr>
</tbody>
</table>

fuel burning rate. The “smoke” in this case is not explicitly tracked by FDS, but rather is assumed to be a function of the mixture fraction.

Suppose, however, that you want to define your own “smoke” and that you want to specify its production rate independently of the HRR (or even in lieu of an actual fire, like a smouldering source). You might also want to define a mass extinction coefficient for the smoke and an assumed molecular weight (as it will be tracked like a gas species). Finally, you also want to visualize the smoke using the “SMOKE3D” feature in Smokeview. Use the following lines:

```
&SPEC ID='MY SMOKE', MW=29., MASS_EXTINCTION_COEFFICIENT=8700. /
&SURF ID='SMOULDER', TMP_FRONT=1000., MASS_FLUX(1)=0.0001, COLOR='RED' /
&VENT XB=0.6,1.0,0.3,0.7,0.0,0.0, SURF_ID='SMOULDER' /

&PROP ID='Acme Smoke', QUANTITY='CHAMBER OBSCURATION', SPEC_ID='MY SMOKE' /
&DEVC XY2=1.00,0.50,0.95, PROP_ID='Acme Smoke', ID='smoke_1' /

&DUMP SMOKE3D_QUANTITY='MY SMOKE', DT_PL3D=30. /
```

The same smoke detector model is used that was described above. Only now, the mass fraction of your species ‘MY SMOKE’ is used in the algorithm, rather than that associated with the mixture fraction. Note that your species will not participate in the radiation calculation. It will merely serve as a surrogate for smoke. Note also that if you specify explicitly a smoke surrogate, you should set SOOT_YIELD=0 on the REAC line to prevent FDS from including smoke as a mixture fraction species.

### 13.3.5 Beam Detection Systems

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

\[
\text{Obscuration} = \left(1 - \exp\left(-K_m \sum_{i=1}^{N} \rho_{\text{soot},i} \Delta x_i\right)\right) \times 100 \% \quad (13.3)
\]

where $i$ is a mesh cell along the path of the beam, $\rho_{\text{soot},i}$ is the soot density of the mesh cell, $\Delta x_i$ is the distance within the mesh cell that is traversed by the beam, and $K_m$ is the mass extinction coefficient. The line in the input file has the form:

```
&DEVC XB=x1,x2,y1,y2,z1,z2, QUANTITY='PATH OBSCURATION', ID='beam1', SETPOINT=0.33 /
```
Since a single linear path cannot span more than one mesh, having a beam detector that crosses multiple meshes will require post processing. Break the beam detector path into multiple DEVC lines, one for each mesh that the beam crosses. The total obscuration is given by

\[ \text{Obscuration} = \left[ 1 - \prod_{i=1}^{N} \left( 1 - \frac{\text{output}_i}{100} \right) \right] \times 100 \% \quad (13.4) \]

where output\(_i\) is the FDS output for the beam detector of the \(i\)th path and the symbol, \(\prod\), indicates a product rather than a sum.

**Example Case: beam_detector**

A 10 m by 10 m by 4 m compartment is filled with smoke, represented as 0.006 kg/kg of the mixture fraction species variable, MIXTURE_FRACTION_2\(^2\). The default soot yield is 0.01 kg/kg, resulting in a uniform soot density of 71.9 mg/m\(^3\). Using the default mass extinction coefficient of 8700 m\(^2\)/kg, the optical depth is calculated to be 0.626 m\(^{-1}\). The compartment has a series of obstructions located at increasing distance from the front in increments of 1 m. The correlation for the output quantity VISIBILITY, Eq. (14.9), produces a visibility distance of 4.8 m. When viewing the smoke levels with Smokeview, you should just barely see the fifth obstacle which is at a distance of 5 m from the front of the compartment. If this is the case, Smokeview is properly displaying the obscuration of the smoke. Three beam detectors are also placed in the compartment. These all have a path length of 10 m, but are at different orientations within the compartment. Using the optical depth of 0.626 m\(^{-1}\) and the path length of 10 m, the expected total obscuration is 99.81 %, which is the result computed by FDS for each of the three detectors.

**13.3.6 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:

\(^2\text{In its default mode, the mixture fraction combustion model requires transport equations for two gas species – MIXTURE_FRACTION_1 and MIXTURE_FRACTION_2. The first is just the fuel gas, the second is a combination of the products of combustion.}\)
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 %. The output of the aspiration system is computed as

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

where \( \dot{m}_i \) is the mass FLOWRATE of the \( i \)-th sampling location, \( \rho_{\text{soot},i}(t - t_{d,i}) \) is the soot density at the \( i \)-th sampling location \( t_{d,i} \) s prior (DELAY) to the current time \( t \), and \( K_m \) is the MASS_EXTINCTION_COEFFICIENT associated with visible light.

Example Case: aspiration_detector

A cubical compartment, 2 m on a side has a three sampling location aspiration system. The three locations have equal flow rates of 0.3 kg/s, and transport times of 50, 100, and 150 s, respectively. No bypass flow rate is specified for the aspiration detector. Combustion products are forced into the bottom of the compartment at a rate of 1 kg/s. The SOOT_YIELD=0.001. Mass is removed from the top of the compartment at a rate of 1 kg/s. The aspiration detector shows an increasing obscuration over time. There is a delay of slightly over 50 s in the initial increase which results from the 50 s transport time for the first sampling location plus a short period of time to transport the combustion products to the sampling location. The detector response has three plateaus that result from the delay times of the sampling locations. The sampling points are co-located, so each plateau represents an additional one third of the soot being transported to the detector. The soot density at the sampling point is \(3.493 \times 10^{-4} \text{ kg/m}^3\). Using this value the plateaus are computed as 63.7 %, 86.8 %, and 95.2 %, as seen in Fig. 13.3.

13.3.7 Electrical Cable Failure

Petra Andersson and Patrick Van Hees of the Swedish National Testing and Research Institute (SP) have proposed that the thermally-induced electrical failure (THIEF) of a cable can be predicted via a simple one-dimensional heat transfer calculation, under the assumption that the cable can be treated as a homogenous cylinder [13]. Their results for PVC cables were encouraging and suggested that the simplification of the analysis is reasonable and that it should extend to other types of cables. The assumptions underlying the THIEF model are as follows:

1. The heat penetration into a cable of circular cross section is largely in the radial direction. This greatly simplifies the analysis, and it is also conservative because it is assumed that the cable is completely surrounded by the heat source.
2. The cable is homogenous in composition. In reality, a cable is constructed of several different types of polymeric materials, cellulosic fillers, and a conducting metal, most often copper.

3. The thermal properties – conductivity, specific heat, and density – of the assumed homogenous cable are independent of temperature. In reality, both the thermal conductivity and specific heat of polymers are temperature-dependent, but this information is very difficult to obtain from manufacturers.

4. It is assumed that no decomposition reactions occur within the cable during its heating, and ignition and burning are not considered in the model. In fact, thermoplastic cables melt, thermosets form a char layer, and both off-gas volatiles up to and beyond the point of electrical failure.

5. Electrical failure occurs when the temperature just inside the cable jacket reaches an experimentally determined value.

Obviously, there are considerable assumptions inherent in the Andersson and Van Hees THIEF model, but their results for various polyvinyl chloride (PVC) cables suggested that it may be sufficient for engineering analyses of a wider variety of cables. The U.S. Nuclear Regulatory Commission sponsored a study of cable failure known as CAROLFIRE [14]. The primary project objective of CAROLFIRE was to characterize the various modes of electrical failure (e.g. hot shorts, shorts to ground) within bundles of power, control and instrument cables. A secondary objective of the project was to develop a simple model to predict thermally-induced electrical failure when a given interior region of the cable reaches an empirically determined threshold temperature. The measurements used for these purposes are described in Volume II of the CAROLFIRE test report. Volume III describes the modeling.

The THIEF model can only predict the temperature profile within the cable as a function of time, given a time-dependent exposing temperature or heat flux. The model does not predict at what temperature the cable fails electrically. This information is gathered from experiment. The CAROLFIRE experimental program included bench-scale, single cable experiments in which temperature measurements were made on the surface of, and at various points within, cables subjected to a uniform heat flux. These experiments provided the link between internal cable temperature and electrical failure. The model can only predict the
interior temperature and infer electrical failure when a given temperature is reached. It is presumed that the temperature of the centermost point in the cable is not necessarily the indicator of electrical failure. This analysis method uses the temperature just inside the cable jacket rather than the centermost temperature, as that is where electrical shorts in a multi-conductor cable are most likely to occur first.

To use the THIEF model in FDS, add lines similar to the following to the input file:

```
&PROP ID='Acme Cable', QUANTITY='CABLE TEMPERATURE', CABLE_MASS_PER_LENGTH=0.529,
   CABLE_FAILURE_TEMPERATURE=400., CABLE_DIAMETER=0.0163,
   CABLE_JACKET_THICKNESS=0.00152 /
&DEVC XYZ=..., ID='Cable 1', PROP_ID='Acme Cable', ORIENTATION=0,0,1 /
&DEVC XYZ=..., ID='Cable 2', PROP_ID='Acme Cable', ORIENTATION=1,0,1 /
```

Most of the terms are self-explanatory, and the logic is similar to that of a sprinkler. The THIEF model is invoked by the specific use of the QUANTITY='CABLE TEMPERATURE'. The cable mass per unit length is in units of kg/m, failure temperature in °C, diameter in m, jacket thickness in m. Note that you can use any orientation, rather than just the 6 coordinate directions.
13.4 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 13.5 for more on CTRL functions. The simplest example of a device is just a timer:

```
&DEVC XYZ=1.2,3.4,5.6, ID='my clock', QUANTITY='TIME', SETPOINT=30. /
```

Anything associated with the device via the parameter, DEVC_ID='my clock’, will change its state at 30 s. For example, if the text were added to an OBST line, that obstruction would change from its INITIAL_STATE of .FALSE. to .TRUE. after 30 s. In other words, it would be created at 30 s instead of at the start of the simulation. This is a simple way to open a door or window.

13.4.1 Creating and Removing Obstructions

In many fire scenarios, the opening or closing of a door or window can lead to dramatic changes in the course of the fire. Sometimes these actions are taken intentionally, sometimes as a result of the fire. Within the framework of an FDS calculation, these actions are represented by the creation or removal of solid obstacles, or the opening or closing of exterior vents.

Remove or create a solid obstruction by assigning the character string DEVC_ID the name of a DEVC ID on the OBST line that is to be created or removed. This will direct FDS to remove or create the obstruction when the device changes state to .FALSE. or .TRUE., respectively. For example, the lines

```
&OBST XB=..., SURF_ID='...', DEVC_ID='det2' /
&DEVC XYZ=..., PROP_ID='...', ID='det1' /
&DEVC XYZ=..., PROP_ID='...', ID='det2', INITIAL_STATE=.TRUE. /
```

will cause the given obstruction to be removed when the specified DEVC changes state.

Creation or removal at a predetermined time can be performed using a DEVC that has TIME as its measured quantity. For example, the following instructions will cause the specified HOLES and OBSTstractions to appear/disappear at the various designated times. These lines are part of the simple test case called create_remove.fds.
The blue obstruction appears at 3 s because its initial state is false, meaning that it does not exist initially. The pink obstruction disappears at 4 s because it does exist initially. The red hole is created at 1 s because it does not exist initially (it is filled in with a red obstruction). The green hole is filled in at 2 s because it does exist (as a hole) initially. **You should always try a simple example first before embarking on a complicated creation/removal scheme for obstructions and holes.**

To learn how to create and remove obstructions multiple times, see Section 13.5.5 for information about the custom control feature.

Starting with FDS version 5.3.0, an obstruction that makes up the boundary of a “pressure zone” (see Section 9.7) **can** be created or removed. The reason for this restriction prior to 5.3 was that abrupt changes in pressure could cause numerical instabilities.

### 13.4.2 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:

```plaintext
&SURF ID='FAN', VOLUME_FLUX=5. /  
&VENT XB=..., SURF_ID='FAN', DEVC_ID='det2' /  
&DEVC ID='det2', XYZ=..., QUANTITY='TIME', SETPOINT=30., INITIAL_STATE=.FALSE. /  
```

Note that at 30 s, the “state” of the ‘FAN’ changes from .FALSE. to .TRUE., or more simply, the ‘FAN’ turns on. Since there is no explicit time function associated with the ‘FAN’, the default 1 s ramp-up will begin at 30 s instead of at 0 s.

If in this example INITIAL_STATE=.TRUE., then the fan should “deactivate,” or turn off at 30 s. Essentially, “activation” of a VENT causes all associated time functions to be delayed until the device SETPOINT is reached. “Deactivation” of a VENT turns off all time functions. Usually this means that the parameters on the SURF line are all nullified, so it is a good idea to check the functionality with a simple example.

Until further notice, a ‘MIRROR’ or ‘OPEN’ VENT should not be activated or deactivated. You can, however, place an obstruction in front of an ‘OPEN’ VENT and then create it or remove it to model the closing or opening of a door or window.
13.5 Advanced Control Functions: The CTRL Namelist Group

There are many systems whose functionality cannot be described by a simple device with a single “setpoint.” Consider for example, a typical HVAC system. It is controlled by a thermostat that is given a temperature setpoint. The system turns on when the temperature goes below the setpoint by some amount and then turns off when the temperature rises above that same setpoint by some amount. This behavior can not be defined by merely specifying a single setpoint. You must also define the range or “deadband” around the setpoint, and whether an increasing or decreasing temperature activates the system. For the HVAC example, crossing the lower edge of the deadband activates heating; crossing the upper edge activates cooling.

While HVAC is not the primary purpose of FDS, there are numerous situations where a system responds to the fire in a non-trivial way. The CTRL input is used to define these more complicated behaviors. A control function will take as input the outputs of one or more devices and/or control functions. In this manner, complicated behaviors can be simulated by making functions of other functions. For most of the control function types, the logical value output of the devices and control functions and the time they last changed state are used as the inputs.

For any object for which a DEVC_ID can be specified (such as OBST or VENT), a CTRL_ID can be specified instead. A control is identified by the ID parameter. The inputs to the control are identified by 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.

### 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' /
```

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANY</td>
<td>Changes state if any INPUTS are .TRUE.</td>
</tr>
<tr>
<td>ALL</td>
<td>Changes state if all INPUTS are .TRUE.</td>
</tr>
<tr>
<td>ONLY</td>
<td>Changes state if and only if N INPUTS are .TRUE.</td>
</tr>
<tr>
<td>AT_LEAST</td>
<td>Changes state if at least N INPUTS are .TRUE.</td>
</tr>
<tr>
<td>TIME_DELAY</td>
<td>Changes state DELAY s after INPUT becomes .TRUE.</td>
</tr>
<tr>
<td>CUSTOM</td>
<td>Changes state based on evaluating a RAMP of the function’s input</td>
</tr>
<tr>
<td>DEADBAND</td>
<td>Behaves like a thermostat</td>
</tr>
<tr>
<td>KILL</td>
<td>Terminates code execution if its sole INPUT is .TRUE.</td>
</tr>
<tr>
<td>RESTART</td>
<td>Dumps restart files if its sole INPUT is .TRUE.</td>
</tr>
</tbody>
</table>
The \texttt{INITIAL\_STATE} of the control function \texttt{SD} is \texttt{.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 \texttt{INITIAL\_STATE} of the control function \texttt{SD} is \texttt{.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:

\begin{verbatim}
&CTRL ID='SD', FUNCTION_TYPE='ALL', INPUT_ID='SD_1','SD_2','SD_3','SD_4' /
\end{verbatim}

The control functions \texttt{AT LEAST} and \texttt{ONLY} are generalizations of \texttt{ANY} and \texttt{ALL}. For example,

\begin{verbatim}
&CTRL ID='SD', FUNCTION_TYPE='AT\_LEAST', N=3, INPUT_ID='SD_1','SD_2','SD_3','SD_4' /
\end{verbatim}

changes the state from \texttt{.FALSE.} to \texttt{.TRUE.} when at least 3 detectors activate. Note that in this example, and the example below, the parameter \texttt{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,

\begin{verbatim}
&CTRL ID='SD', FUNCTION_TYPE='ONLY', N=3, INPUT_ID='SD_1','SD_2','SD_3','SD_4' /
\end{verbatim}

changes the state from \texttt{.FALSE.} to \texttt{.TRUE.} when 3, and only 3, detectors activate.

\section*{13.5.2 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.

\begin{verbatim}
&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. /
\end{verbatim}

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 \texttt{Spk\_29}, measured by \texttt{Spk\_29\_link}, before water can flow out of the head.

\section*{13.5.3 Control Function: DEADBAND}

This control function behaves like an HVAC thermostat. It can operate in one of two modes analogous to heating or cooling. The function is provided with an \texttt{INPUT\_ID} which is the \texttt{DEVC} whose value is used by the function, a \texttt{DEADBAND}, and the mode of operation by \texttt{ON\_BOUND}. If \texttt{ON\_BOUND=’LOWER’}, the function changes state from its \texttt{INITIAL\_VALUE} when the value of the \texttt{INPUT\_ID} drops below the lower value in \texttt{DEADBAND} and reverts when it increases past the upper value, i.e. like a heating system. The reverse will occur if \texttt{ON\_BOUND=’UPPER’}, i.e. like a cooling system.

For an HVAC example, the following lines of input would set up a simple thermostat:
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.

**13.5.4 Control Function: RESTART and KILL**

There are times when you might only want to run a simulation until some goal is reached. Previously this could generally only be done by constantly monitoring the simulation’s output and manually stopping the calculation when you determine that the goal has been reached. The KILL control function can do this automatically.

Additionally there are analyses where you might want to create some baseline condition and then run multiple permutations of that baseline. For example, you might want to run a series of simulations where different mitigation strategies are tested once a detector alarms. Using the RESTART control function, you can cause a restart file to be created once a desired condition is met. The simulation can continue and the restart files can be copied to have the job identifying string, CHID, of the various permutations (providing of course that the usual restrictions on the use of restart files are followed). For example, the lines

```
&DEVC ID='temp', QUANTITY='TEMPERATURE', SETPOINT=1000., XYZ=4.5,6.7,3.6 /  
&DEVC ID='velo', QUANTITY='VELOCITY', SETPOINT=10., XYZ=4.5,6.7,3.6 /  
&CTRL ID='kill', FUNCTION_TYPE='KILL', INPUT_ID='temp' /  
&CTRL ID='restart', FUNCTION_TYPE='RESTART', INPUT_ID='velo' /
```

will “kill” the job and output restart files when the temperature at the given point rises above 1000 °C; or just force restart files to be output when the velocity at a given point exceeds 10 m/s.

**13.5.5 Control Function: CUSTOM**

For most of the control function types, the logical (true/false) output of the devices and control functions and the time they last changed state are taken as inputs. A CUSTOM function uses the numerical output of a DEVC along with a RAMP to determine the output of the function. When the RAMP output for the DEVC value is negative, the CTRL will have the value of its INITIAL_STATE. When the RAMP output for the DEVC value is positive, the CTRL will have the opposite value of its INITIAL_STATE. In the case below, the CUSTOM control function uses the numerical output of a timer device as its input. The function returns true (the default value for INITIAL_STATE is .FALSE.) when the F parameter in the ramp specified with RAMP_ID is a positive value and false when the RAMP F value is negative. In this case, the control would start false and would switch to true when the timer reaches 60 s. It would then stay in a true state until the timer reaches 120 s and would then change back to false.

Note that when using control functions the IDs assigned to both the CTRL and the DEVC inputs must be unique across both sets of inputs, i.e. you cannot use the same ID for both a control function and a device.
In the HVAC example above, we could set the system to function on a fixed cycle by using a CUSTOM control function based on time:

```
&SURF ID='FAN', TMP_FRONT=40., VOLUME_FLUX=-1. / 
&VENT XB=-0.3,0.3,-0.3,0.3,0,0,0,0, &SURF_ID='FAN', CTRL_ID='cycling timer' / 
&DEVC ID='TIMER', XYZ=2.4,5.7,3.6, QUANTITY='TIME' / 
&CTRL ID='cycling timer', FUNCTION_TYPE='CUSTOM', INPUT_ID='TIMER', RAMP_ID='cycle' / 
&RAMP ID='cycle', T= 59, F=-1 / 
&RAMP ID='cycle', T= 61, F= 1 / 
&RAMP ID='cycle', T=119, F= 1 / 
&RAMP ID='cycle', T=121, F=-1 / 
```

In the above example the fan will be off initially, turn on at 60 s and then turn off at 120 s.

You can make an obstruction appear and disappear multiple times by using lines like

```
&OBST XB=..., SURF_ID='whatever', CTRL_ID='cycling timer' / 
&DEVC ID='TIMER', XYZ=..., QUANTITY='TIME' / 
&CTRL ID='cycling timer', FUNCTION_TYPE='CUSTOM', INPUT_ID='TIMER', RAMP_ID='cycle' / 
&RAMP ID='cycle', T= 0, F=-1 / 
&RAMP ID='cycle', T= 59, F=-1 / 
&RAMP ID='cycle', T= 61, F= 1 / 
&RAMP ID='cycle', T=119, F= 1 / 
&RAMP ID='cycle', T=121, F=-1 / 
```

The above will have the obstacle initially removed, then added at 60 s, and removed again at 120 s.

Experiment with these combinations using a simple case before trying a case to make sure that FDS indeed is doing what is intended.

### 13.5.6 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' / 
```

### 13.5.7 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.
13.5.8 Example Case: activate_vents

The simple test case called activate_vents demonstrates the several of the control functions. Figure 13.4 shows seven multiply-colored vents that activate at different times, depending on the particular timing or control function.

![Figure 13.4: Output of the activate_vents test case at 5, 10 and 15 s.]

13.6 Controlling a RAMP

For any user defined RAMP, the normal independent variable, for example time for RAMP_V, can be replaced by the output of a DEVC. This is done by specifying the input DEVC_ID one on of the RAMP input lines. In the following example a blower is turned on when the temperature exceeds 100 °C and is turned off when the temperature exceeds 200 °C. This is similar functionality to the CUSTOM control function, but it allows for variable response rather than just on or off.

![Example RAMP configuration]
13.7 Visualizing FDS Devices Using Smokeview Objects

Smokeview generates visual representations of FDS devices using instructions found in a data file named objects.svo. These instructions correspond to OpenGL library calls, the same type of calls Smokeview uses to visualize FDS cases. New objects may be designed and drawn without modifying Smokeview and more importantly may be created by any user not just the FDS/Smokeview developers. This section gives an overview of Smokeview objects detailing what objects are available and how to modify them. Further documentation giving underlying technical details may be found in the Smokeview User’s Guide [2].

Smokeview objects may be static or dynamic. A static object is defined entirely in terms of data and instructions found in the objects.svo file. For example, the sensor object is static, it is drawn as a small green sphere with a fixed diameter. Its appearance remains the same regardless of how an FDS input file is set up. A dynamic object is also defined using instructions found in objects.svo but in addition uses data specified on the &PROP namelist statement and/or data contained in a particle file. As a result, the appearance of dynamic objects depends on the particular FDS case that is run. For example, the tsphere object is dynamic. The diameter and an image used to cover the sphere (known as a texture map) is specified in an FDS input file.

13.7.1 Static Smokeview Objects

Smokeview objects consist of one or more frames or views. Smokeview then displays FDS devices in a normal/inactive state or in an active state. A sprinkler, for example, is drawn differently depending on whether it has activated or not. When FDS determines that a device has activated it places a message in the .smv file indicating the object number, the activation time and the state (0 for inactive or 1 for active). Smokeview then draws the corresponding frame. Tables 13.3 and 13.4 give a list of various static objects. Each entry shows an image of the object in its normal/inactive state and in its active state if it has one. The intersection of black tubes indicate the origin, the part of the device displayed at the \((x,y,z)\) coordinate specified on the &DEVC input line.

The SMOKEVIEW_ID keyword found on the &PROP namelist statement is used to associate an FDS device with a Smokeview object. The following FDS input file lines were used to display the target device in Table 13.3.

```
&PROP ID='target' SMOKEVIEW_ID='target' /
&DEVC XYZ=0.5,0.8,0.6, QUANTITY='TEMPERATURE' PROP_ID='target' /
```
### Table 13.3: Single Frame Static Objects

<table>
<thead>
<tr>
<th>SMOKEVIEW_ID</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensor</td>
<td><img src="image1" alt="Sensor Image" /></td>
</tr>
<tr>
<td>target</td>
<td><img src="image2" alt="Target Image" /></td>
</tr>
</tbody>
</table>

### Table 13.4: Dual Frame Static Objects

<table>
<thead>
<tr>
<th>SMOKEVIEW_ID</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>heat_detector</td>
<td><img src="image3" alt="Heat Detector Image" /></td>
</tr>
<tr>
<td>nozzle</td>
<td><img src="image4" alt="Nozzle Image" /></td>
</tr>
</tbody>
</table>
13.7.2 Dynamic Smokeview Objects - Customized Using &PROP Parameters

The appearance of several Smokeview objects may be modified using data specified on the &PROP namelist statement in an FDS input file. Objects may also be customized using data stored in a particle file. This is discussed in the next section.

The SMOKEVIEW_PARAMETERS keyword on the &PROP namelist statement is used to customize the appearance of Smokeview objects. For example, the &DEV and &PROP statements:

```
&PROP ID='sphere' SMOKEVIEW_PARAMETERS(1:4)='R=0','G=255','B=0',
'D=0.5' SMOKEVIEW_ID='sphere' /
&DEV XYZ=0.5,0.8,1.5, QUANTITY='TEMPERATURE' PROP_ID='sphere' /
```

create an FDS device drawn as a sphere colored green with diameter 0.5. Each parameter specified using the SMOKEVIEW_PARAMETERS keyword is a text string enclosed in single quotes. The text string is of the
form ‘keyword=value’ where possible keywords are found in the objects.svo file (labels beginning with ‘:’). For example, R, G, B and D may be used as keywords to customize the following sphere object:

OBJECTDEF // object for particle file sphere
sphere
 :R=0 :G=0 :B=0 :D=0.1
$R $G $B setrgb
$D drawsphere

Another, Smokeview object, the tsphere, uses a texture map or picture to alter the appearance of the object. The texture map is specified using SMOKEVIEW_PARAMETERS keyword by placing the characters t% before the texture file name (e.g. t%texturefile.jpg).

Table 13.5 gives a list of dynamic objects and the keyword/parameter pairs used to specify them. Each entry shows an image of the object and the parameters used to customize its appearance.

Table 13.5: Dynamic Objects - Customized using SMOKEVIEW_PARAMETERS on a &PROP line

<table>
<thead>
<tr>
<th>SMOKEVIEW_ID</th>
<th>SMOKEVIEW_PARAMETERS</th>
<th>Image</th>
</tr>
</thead>
</table>
| ball         | SMOKEVIEW_PARAMETERS(1:6)=
               'R=128','G=192','B=255',
               'DX=0.5','DY=.75','DZ=1.0'
               R, G, B - red, green, blue color components ranging from 0 to 255
               DX, DY, DZ - amount ball is stretched along x, y, z axis respectively |
| cone         | SMOKEVIEW_PARAMETERS(1:5)=
               'R=128','G=255','B=192',
               'D=0.4','H=0.6'
               R, G, B - red, green, blue color components ranging from 0 to 255
               D, H - diameter and length of cone respectively |
| fan          | SMOKEVIEW_PARAMETERS(1:11)=
               'HUB_R=0','HUB_G=0','HUB_B=0',
               'HUB_D=0.1','HUB_L=0.12',
               'BLADE_R=128','BLADE_G=64',
               'BLADE_B=32','BLADE_ANGLE=60.0',
               'BLADE_D=0.5','BLADE_H=0.09'
               HUB_R, HUB_G, HUB_B - red, green, blue color components of fan hub ranging from 0 to 255
               HUB_D, HUB_L - diameter and length of fan hub
               BLADE_R, BLADE_G, BLADE_B - red, green, blue color components of fan blades ranging from 0 to 255
               BLADE_ANGLE, BLADE_D, BLADE_H - angle, diameter and height of a fan blade |
### Table 13.5: Dynamic Objects (continued)

<table>
<thead>
<tr>
<th>SMOKEVIEW_ID</th>
<th>SMOKEVIEW_PARAMETERS</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>tsphere</td>
<td><code>SMOKEVIEW_PARAMETERS(1:9) = 'R=255','G=255','B=255', 'AX0=0.0','ELEV0=90.0', 'ROT0=0.0','ROTATION_RATE=10.0', 'D=1.0', 'tfile=&quot;t%sphere_cover_04.png&quot;'</code></td>
<td><img src="image" alt="Red Sphere" /></td>
</tr>
<tr>
<td></td>
<td>R, G, B - red, green, blue color components ranging from 0 to 255</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AX0, ELEV0, ROT0 - initial azimuth, elevation and rotation angle respectively</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ROTATION_RATE - rotation rate about z axis in degrees per second</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D - diameter of sphere</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tfile - name of texture map file</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>vent</th>
<th><code>SMOKEVIEW_PARAMETERS(1:6) = 'R=192','G=192','B=128', 'W=0.5','H=1.0', 'ROT=90.0'</code></th>
<th><img src="image" alt="Active Vent" /></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R, G, B - red, green, blue color components ranging from 0 to 255</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W, H - width and height of vent respectively</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ROT - angle that vent is rotated</td>
<td></td>
</tr>
</tbody>
</table>

### 13.7.3 Dynamic Smokeview Objects - Customized Using &PROP Parameters and Particle File Data

Particle file data may be used to customize the appearance of Smokeview objects. Any objects that have color labels named R, G, B (including those objects in Table 13.5) may be colored using particle file data. In addition, objects that use variable names that match shortened particle file quantity names may be customized. This data may be used to alter the geometry or structure of the object using particle file data. For example, `U-VEL`, `V-VEL`, `W-VEL` and `temp` are shortened quantity names that correspond to the full names `U-VELOCITY`, `V-VELOCITY`, `W-VELOCITY` and `TEMPERATURE`. These full names are documented in Table 14.1 in this guide.

---

3Short forms of particle file quantity names appear in the Smokeview colorbar label and in the Smokeview File/Bounds dialog box.
The first three lines of the **velegg** object definition are:

```
OBJECTDEF // color with FDS quantity, stretch with velocity
velegg
 :R=0 :G=0 :B=0 :D :U-VEL=1.0 :V-VEL=1.0 :W-VEL=1.0
```

The variables **U-VEL**, **V-VEL**, and **W-VEL** in the above line are also particle file quantities (shortened version) that may be selected in an FDS input file. If they are selected, then the **velegg** object may be used to display particle file information. This object colors the sphere using the currently selected Smokeview particle variable and stretches a sphere in the x, y and z directions using the **U-VEL**, **V-VEL**, and **W-VEL** velocity particle data respectively.

Table 13.6 documents those objects that can be customized using particle file data. These objects may be customized as before using data specified with the **SMOKEVIEW_PARAMETERS** keyword or using particle file data.

**Table 13.6: Dynamic Objects - Customized using SMOKEVIEW_PARAMETERS on a &PROP line and Particle File Data**

<table>
<thead>
<tr>
<th>SMOKEVIEW_ID</th>
<th>SMOKEVIEW_PARAMETERS</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>box</td>
<td><code>SMOKEVIEW_PARAMETERS(1:6)= 'R=192','G=255','B=128', 'DX=0.25','DY=.5','DZ=0.125'</code></td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td>tube</td>
<td><code>SMOKEVIEW_PARAMETERS(1:5)= 'R=255','G=0','B=0', 'D=0.2','L=0.6'</code></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>velegg</td>
<td><code>SMOKEVIEW_PARAMETERS(1:9)= :R=0 :G=0 :B=0 :U-VEL=1.0 :V-VEL=1.0 :W-VEL=1.0 :VELMIN :VELMAX :D</code></td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Table 13.6: Dynamic Objects (continued)

<table>
<thead>
<tr>
<th>SMOKEVIEW_ID</th>
<th>SMOKEVIEW_PARAMETERS</th>
<th>Image</th>
</tr>
</thead>
</table>
| veltube      | SMOKEVIEW_PARAMETERS(1:9) =
               :R=0 :G=0 :B=0
               :U-VEL=1.0 :V-VEL=1.0 :W-VEL=1.0
               :VELMIN :VELMAX :D |
|              | R, G, B - red, green, blue color components ranging from 0 to 255 |
|              | U-VEL, V-VEL, W-VEL - u, v, w components of velocity |
|              | VELMIN, VELMAX - minimum and maximum velocity |
|              | D - diameter of tube at VELMAX |
Chapter 14

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 PLo3D 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.

14.1 Output Control Parameters: The DUMP Namelist Group

The namelist group DUMP contains parameters (Table 15.5) 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.
STATE_FILE  If .TRUE., produce an output file listing the species mass fractions associated with the mixture fraction, assuming complete combustion. It is .FALSE. by default, and only makes sense for calculations involving the mixture fraction model.

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  FDS purges the output file buffers periodically and forces the data to be written out into the respective output files. It does this to make it easier to view the case in Smokeview while it is running. It has been noticed on Windows machines that occasionally a runtime error occurs because of file access problems related to the buffer flushing. If this happens, set this parameter to .FALSE., but be aware that it may not be possible to look at output in Smokeview until after the calculation is finished. You may also set DT_FLUSH to control the frequency of the file flushing. Its default value is the duration of the simulation divided by NFRAMES.

STATUS_FILES  If .TRUE., produces an output file CHID.notready which is deleted, if the simulation is completed successfully. This file can be used as an error indicator. It is .FALSE. by default.
14.2 Output File Types

FDS has various types of output files that store computed data. Some of the files are in binary format and intended to be read and rendered by Smokeview. Some of the files are just comma-delimited text files. It is important to remember that you must explicitly declare in the input file most of the FDS output data. A considerable amount of the input file is usually devoted to this.

14.2.1 Device Output: The DEVC Namelist Group

For many commonly used measurement devices there is no need to associate a specific PROP line to the DEVC entry. In such cases, use the character string QUANTITY to indicate that a particular gas or solid phase quantity at the point should be recorded in the output file with the suffix _devc.csv. The quantities are listed in Table 14.1. 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

```verbatim
&DEVC XYZ=6.7,2.9,2.1, QUANTITY='TEMPERATURE', ID='T-1' /
```

and a column will be added to the output file CHID_devc.csv under the label ‘T-1’. In this case, the ID has no other role than as a column label in the output file. Note that versions of FDS prior to version 5 used an 8 cell linear interpolation for a given gas phase point measurement. In other words, if you specified a point via the triplet of real numbers, XYZ, FDS would calculate the value of the quantity by linearly interpolating the values defined at the centers of the 8 nearest cells. Starting in FDS 5, this is no longer done. Instead, FDS reports the value of the QUANTITY in the cell where the point XYZ is located.

Devices on Solid Surfaces

When prescribing a solid phase quantity, be sure to position the probe at a solid surface. It is not always obvious where the solid surface is since the mesh does not always align with the input obstruction locations. To help locate the appropriate surface, the parameter IOR must be included when designating a solid phase quantity, except when using the STATISTICS feature described in Section 14.3.10 in which case the output quantity is not associated with just a single point on the surface. If the orientation of the solid surface is in the positive x direction IOR=1, negative x direction IOR=-1, positive y IOR=2, negative IOR=-2, positive z IOR=3, and negative z IOR=-3. There are still instances where FDS cannot determine which solid surface is being designated, in which case an error message appears in the diagnostic output file. Re-position the probe and try again. For example, the line

```verbatim
&DEVC XYZ=0.7,0.9,2.1, QUANTITY='WALL TEMPERATURE', IOR=-2, ID='...' /
```

designates the surface temperature of a wall facing the negative y direction.

Non-Pointwise Devices

In addition to point measurements, the DEVC group can be used to report integrated quantities (See Table 14.1). For example, you may want to know the mass flow out of a door or window. To report this, add the line

```verbatim
&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 \( \mathbf{X_B} \) 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 \( \mathbf{X_B} \) – avoid planes or volumes that cross multiple mesh boundaries. FDS has to decide which mesh to use in the integration, and it chooses the finest mesh overlapping the centroid of the designated plane or volume.

**Linear Array of Point Devices**

The `DEVC` input construct enables you to specify a linear array of devices. By adding the parameter `POINTS` and using the sextuple coordinate array \( \mathbf{X_B} \), you can direct FDS to create a line of devices from \((x_1, y_1, z_1)\) to \((x_2, y_2, z_2)\) as follows:

```plaintext
&DEVC XB=X1,X2,Y1,Y2,Z1,Z2, QUANTITY='TEMPERATURE', ID='TC Tree', POINTS=20 /
```

In a file called `CHID_line.csv`, there will be 4 columns of data (3 spatial coordinates and the value) with the 20 temperature points presented as a running average. The averaging time is given by `DT_DEVC_LINE` on the `DUMP` line. It is 0.25 times the total simulation time by default. This is a convenient way to output a time-averaged linear profile of a quantity, like an array of thermocouples.

**14.2.2 Quantities within Solids: The `PROF` Namelist Group**

FDS uses a fine, non-uniform, one-dimensional mesh at each boundary cell to compute heat transfer within a solid. The parameters (Table 15.18) to specify a given `PROFILE` are similar to those used to specify a surface quantity in the `DEVC` group. \( \mathbf{XYZ} \) designates the triplet of coordinates, \( \mathbf{QUANTITY} \) is the physical quantity to monitor, \( \mathbf{IOR} \) the orientation, and \( \mathbf{ID} \) 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:

```plaintext
&PROF XYZ=..., QUANTITY='TEMPERATURE', ID='TU1SA_FDS', IOR=3 /
```

Other possible quantities are the total density of the wall (\( \mathbf{QUANTITY} = \text{`DENSITY'} \)) or densities of solid material components (\( \mathbf{QUANTITY} = \text{`MATL_ID'} \)), 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:

```plaintext
&DEVC XYZ=..., QUANTITY='INSIDE WALL TEMPERATURE', DEPTH=0.005, ID='Temp_1', IOR=3 /
```

The parameter \( \mathbf{DEPTH} \) (m) indicates the distance inside the solid surface. Note that this \( \mathbf{QUANTITY} \) 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. If you just want to know the temperature of the back surface of the “wall,” then use

```plaintext
&DEVC XYZ=..., QUANTITY='BACK WALL TEMPERATURE', ID='Temp_b', IOR=3 /
```

Note that this quantity is only meaningful if the front or exposed surface of the “wall” has the attribute \( \text{BACKING}='EXPOSED' \) on the `SURF` line that defines it. The coordinates, \( \mathbf{XYZ} \), and orientation, \( \mathbf{IOR} \), refer to the front surface. To check that the heat conduction calculation is being done properly, you can add the additional line

158
&DEVC XYZ=..., QUANTITY='WALL TEMPERATURE', ID='Temp_f', IOR=-3 /

where now XYZ and IOR refer to the coordinates and orientation of the back side of the wall. These two wall temperatures ought to be the same. Remember that the “wall” in this case can only be at most one mesh cell thick, and its THICKNESS need not be the same as the mesh cell width. Rather, the THICKNESS ought to be the actual thickness of the “wall” through which FDS performs a 1-D heat conduction calculation.

14.2.3 Animated Planar Slices: The SLCF Namelist Group

The SLCF (“slice file”) namelist group parameters (Table 15.23) allows you to record various gas phase quantities at more than a single point. A “slice” refers to a subset of the whole domain. It can be a line, plane, or volume, depending on the values of XB. The sextuplet XB indicates the boundaries of the “slice” plane. XB is prescribed as in the OBST or VENT groups, with the possibility that 0, 2, or 4 out of the 6 values be the same to indicate a volume, plane or line, respectively. A handy trick is to specify, for example, PBY=5.3 instead of XB if it is desired that the entire plane $y = 5.3$ slicing through the domain be saved. PBX and PBZ control planes perpendicular to the $x$ and $z$ axes, respectively.

If the “slice” is a volume specified with XB, then its output frequency is controlled by the parameter DT_SL3D. By default, FDS sets \( DT_{SL3D} = (T\_END-T\_BEGIN)/5 \), the same default frequency as Plot3D file output. The user may specify a different value of DT_SL3D on DUMP. The user is cautioned that 3D slice files can become extremely large if DT_SL3D is small.

Animated vectors can be created in Smokeview if a given SLCF line has the attribute VECTOR=.TRUE. If two SLCF entries are in the same plane, then only one of the lines needs to have VECTOR=.TRUE. Otherwise, a redundant set of velocity component slices will be created.

Normally, FDS averages slice file data at cell corners. For example, gas temperatures are computed at cell centers, but they are linearly interpolated to cell corners and output to a file that is read by Smokeview. To prevent this from happening, set CELL_CENTERED=.TRUE.. This forces FDS to output the actual cell-centered data with no averaging. Note that this feature is mainly useful for diagnostics because it enables you to visualize the values that FDS actually computes. Note also that this feature should only be used for scalar quantities that are computed at cell centers, like temperatures, mass fractions, etc.

Slice file information is recorded in files (See Section 19.7) 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 14.4 for more details.

14.2.4 Animated Boundary Quantities: The BNDF Namelist Group

The BNDF (“boundary file”) namelist group parameters allows you to record surface quantities at all solid obstructions. As with the SLCF group, each quantity is prescribed with a separate BNDF line, and the output files are of the form CHID_n bf. No physical coordinates need be specified, however, just QUANTITY. See Table 14.1. 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 19.9) 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).

159
Normally, FDS averages boundary file data at cell corners. For example, surface temperatures are computed at the center of each surface cell, but they are linearly interpolated to cell corners and output to a file that is read by Smokeview. To prevent this from happening, set `CELL_CENTERED=.TRUE.` on the `BNDF` line. This forces FDS to output the actual cell-centered data with no averaging. Note that this feature is mainly useful for diagnostics because it enables you to visualize the values that FDS actually computes.

**14.2.5 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 (`DT_ISOF` on the `DUMP` line).

Any gas phase quantity can be animated via iso-surfaces, but use caution. To render an iso-surface, the desired quantity must be computed in every mesh cell at every output time step. For quantities like `TEMPERATURE`, this is not a problem, as FDS computes it and saves it anyway. However, soot density or oxygen demand substantial amounts of time to compute at each mesh cell.

**14.2.6 Plot3D Static Data Dumps**

By default, flow field data in Plot3D format is output 5 times a run. Five quantities are written out to a file at one instant in time. The default specification is:

```
&DUMP ..., PLOT3D_QUANTITY(1:5)= 'TEMPERATURE',
         'U-VELOCITY', 'V-VELOCITY', 'W-VELOCITY', 'HRRPUV' /
```

It's best to leave the velocity components as is, because Smokeview uses them to draw velocity vectors. The first and fifth quantities can be changed with the parameters `PLOT3D_QUANTITY(1)` and `PLOT3D_QUANTITY(5)` on the `DUMP` line. If any of the specified quantities require the additional specification of a particular species, use `PLOT3D_SPEC_ID(n)` to provide the `SPEC_ID` for `PLOT3D_QUANTITY(n)`.

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

Data stored in Plot3D [15] files (See Section 19.8) 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.
14.2.7 SMOKE3D: Realistic Smoke and Fire

When you do a fire simulation using the default mixture fraction combustion model, FDS automatically creates two output files that are rendered by Smokeview as realistic looking smoke and fire. By default, the output quantities are the ‘MASS FRACTION’ of ‘soot’ and ‘HRRPUV’ (Heat Release Rate Per Unit Volume) are used in the visualization. You have the option of rendering any other species mass fraction instead of ‘soot’, so long as the MASS_EXTINCTION_COEFFICIENT (either from the REAC line, or overridden by the value on the SPEC line) is appropriate in describing the attenuation of visible light by the specified gas species. The alternative gas species is given by SMOKE3D_QUANTITY on the DUMP line. If the specified quantity requires the additional specification of a particular species, use SMOKE3D_SPEC_ID to provide the SPEC_ID. See the Smokeview User’s Guide for more details on how these quantities are rendered.

Here is an example of how to control the smoke species. Normally, you do not need to do this as the “smoke” is an assumed part of the default mixture fraction model.

```
&SPEC ID='MY SMOKE', MW=29., MASS_EXTINCTION_COEFFICIENT=8700. /
&SURF ID='NO FIRE', TMP_FRONT=1000., MASS_FLUX(1)=0.0001, COLOR='RED' /
&VENT XB=0.6,1.0,0.3,0.7,0.0,0.0, SURF_ID='NO FIRE' /
&DUMP SMOKE3D_QUANTITY='MASS FRACTION', SMOKE3D_SPEC_ID='MY SMOKE' /
```

The production rate of ‘MY SMOKE’ is 0.0001 kg/m$^2$/s, applied to an area of 0.16 m$^2$. The MASS_EXTINCTION_COEFFICIENT is passed to Smokeview to be used for visualization.
14.3 Special Output Quantities

This section lists a variety of output quantities that are useful for studying thermally-driven flows, combustion, pyrolysis, and so forth. Note that some of the output quantities can be produced in a variety of ways.

14.3.1 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. Note that the reported value of the burning rate is not adjusted to account for the possibility that each individual material might have a different heat of combustion. For this reason, it is not always the case that the reported total burning rate multiplied by the gas phase heat of combustion is equal to the reported heat release rate.

Let $\Omega$ denote the unblocked computational domain, i.e. the volume within the bounding rectangle occupied by gas. Let $\partial \Omega$ by the boundary of $\Omega$. 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

$$\dot{Q} = \int_{\Omega} \dot{q}'''' dV$$ (14.1)

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

$$\dot{Q}_r = \int_{\Omega} \nabla \cdot \mathbf{q}_r \, dV = \int_{\partial \Omega} \mathbf{q}_r \cdot d\mathbf{S} = \int_{\partial \Omega} \dot{q}_r'' dA$$ (14.2)

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

$$\dot{Q}_{conv} = \int_{\partial \Omega} c_p \rho (T - T_\infty) \mathbf{u} \cdot d\mathbf{S} + \int_{\partial \Omega_2} \dot{q}_r'' dA$$ (14.3)

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

$$\dot{Q}_{cond} = \int_{\partial \Omega_1} \dot{q}_r'' + \dot{q}_c'' dA$$ (14.4)

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

$$\dot{Q} \approx \dot{Q}_{conv} + \dot{Q}_{cond}$$ (14.5)

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.
Note that, in axially symmetric simulations, the coordinates \( \text{XB}(3) \) and \( \text{XB}(4) \) refer to the width of the cylindrical sector at 1 m distance from the cylinder axis, and the volume \( \Omega \) is the volume of the cylindrical sector

\[
\Omega = \frac{1}{2R_1} (\text{XB}(2)^2 - \text{XB}(1)^2) (\text{XB}(4) - \text{XB}(3)) (\text{XB}(6) - \text{XB}(5))
\]  

(14.6)

where \( R_1 = 1 \text{ m} \).

**14.3.2 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*, \( K \) [16]. The intensity of monochromatic light passing a distance \( L \) through smoke is attenuated according to

\[
\frac{I}{I_0} = e^{-KL}
\]  

(14.7)

The light extinction coefficient, \( K \), is a product of the density of smoke particulate, \( \rho Y_s \), and a mass specific extinction coefficient that is fuel dependent

\[
K = K_m \rho Y_s
\]  

(14.8)

Devices that output a % obscuration such as a **DEVC** with a **QUANTITY** of **ASPIRATION**, **CHAMBER OBSCURATION** (smoke detector), or **PATH OBSCURATION** (beam detector) are discussed respectively in Section 13.3.6, Section 13.3.4, and Section 13.3.5.

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

\[
S = C/K
\]  

(14.9)

where \( C \) is a non-dimensional constant characteristic of the type of object being viewed through the smoke, i.e. \( C = 8 \) for a light-emitting sign and \( C = 3 \) for a light-reflecting sign [16]. Since \( K \) varies from point to point in the domain, the visibility \( S \) does as well. Keep in mind that FDS can only track smoke whose production rate and composition are specified. Predicting either is beyond the capability of the present version of the model.

Three parameter control smoke production and visibility; each parameter is input on the **REAC** line. The first parameter is **SOOT_YIELD**, which is the fraction of fuel mass that is converted to soot if the mixture fraction model is being used. The second parameter is called the **MASS_EXTINCTION_COEFFICIENT**, and it is the \( K_m \) in Eq. (14.8). The default value is 8700 m\(^2\)/kg, a value suggested for flaming combustion of wood and plastics\(^1\). The third parameter is called the **VISIBILITY_FACTOR**, the constant \( C \) in Eq. (14.9). It is 3 by default.

The gas phase output quantity ‘**EXTINCTION COEFFICIENT**’ is \( K \). A similar quantity is the ‘**OPTICAL DENSITY**’, \( K/2.3 \), the result of using \( \log_{10} \) in the definition

\[
D \equiv -\frac{1}{L} \log_{10} \left( \frac{I}{I_0} \right) = K \log_{10} e
\]  

(14.10)

The visibility \( S \) is output via the keyword **VISIBILITY**. Note that, by default, the visibility is associated with the smoke that is implicitly defined by the mixture fraction model. However, this quantity can also be associated with an explicitly defined species via the inclusion of a **SPEC_ID**. In other words, you can specify the output quantity ‘**VISIBILITY**’ along with a **SPEC_ID**. This does not require that you do a mixture fraction calculation; only that you have specified the given species via a separate **SPEC** line. You can specify a unique **MASS_EXTINCTION_COEFFICIENT** on the **SPEC** line as well.

\(^1\)For most flaming fuels, a suggested value for \( K_m \) is 8700 m\(^2\)/kg ± 1100 m\(^2\)/kg at a wavelength of 633 nm [17]
14.3.3 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 \cite{18} is as follows: Consider a continuous function \( T(z) \) defining temperature \( T \) as a function of height above the floor \( z \), where \( z = 0 \) is the floor and \( z = H \) is the ceiling. Define \( T_u \) as the upper layer temperature, \( T_l \) as the lower layer temperature, and \( z_{int} \) as the interface height. Compute the quantities:

\[
(H - z_{int}) T_u + z_{int} T_l = \int_0^H T(z) \, dz = I_1
\]

\[
(H - z_{int}) \frac{1}{T_u} + z_{int} \frac{1}{T_l} = \int_0^H \frac{1}{T(z)} \, dz = I_2
\]

Solve for \( z_{int} \):

\[
z_{int} = \frac{T_l(I_1I_2 - H^2)}{I_1 + I_2T_l^2 - 2T_lH}
\] (14.11)

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

\[
(H - z_{int}) T_u = \int_{z_{int}}^H T(z) \, dz
\] (14.12)

Further discussion of similar procedures can be found in Ref. \cite{19}.

The quantities LAYER HEIGHT, UPPER TEMPERATURE and LOWER TEMPERATURE can be designated via “device” (DEVC) lines in the input file\cite{2}. For example, the entry

\&DEVC XB=2.0,2.0,3.0,3.0,0,0,0,3.0, 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.

14.3.4 Thermocouples

The output quantity THERMOCOUPLE is the temperature of a modeled thermocouple. The thermocouple temperature lags the true gas temperature by an amount determined mainly by its bead size. It is found by solving the following equation for the thermocouple temperature, \( T_{rc} \) \cite{20}

\[
\rho_{rc} c_{rc} \frac{dT_{rc}}{dt} = \varepsilon_{rc} (U/4 - \sigma T_{rc}^4) + h(T_g - T_{rc}) = 0
\] (14.13)

where \( \varepsilon_{rc} \) is the emissivity of the thermocouple, \( U \) is the integrated radiative intensity, \( T_g \) is the true gas temperature, and \( h \) is the heat transfer coefficient to a small sphere, \( h = k \text{Nu}/d_{rc} \). The bead BEAD_DIAMETER, BEAD_EMISSIVITY, BEAD_DENSITY, and BEAD_SPECIFIC_HEAT are given on the associated PROP line.

The default value for the bead diameter is 0.001 m. The default emissivity is 0.85. The default values for the bead density and specific heat are that of nickel; 8908 kg/m\(^3\) and 0.44 kJ/kg/K, respectively. See the discussion on heat transfer to a water droplet in the Technical Reference Guide for details of the convective heat transfer to a small sphere.

\footnote{Note that in FDS 5 and beyond, these quantities are no longer available as slice files.}
14.3.5 Heat Fluxes and Thermal Radiation

There are various ways of recording the heat flux at a solid boundary. If you want to record the net heat flux to the surface, \( \dot{q}_c'' + \dot{q}_r'' \), use the QUANTITY called ‘NET HEAT FLUX’. The individual components, the net convective and radiative fluxes, are ‘CONVECTIVE HEAT FLUX’ and ‘RADIATIVE HEAT FLUX’, respectively. If you want to compare predicted heat flux with a measurement, you often need to use ‘GAUGE HEAT FLUX’. The difference between ‘NET HEAT FLUX’ and ‘GAUGE HEAT FLUX’ is that the former is the rate at which energy is absorbed by the solid surface; the latter is the amount of energy that would be absorbed if the surface were cold (or some specified temperature \( T_G \)):

\[
\dot{q}_c''/\epsilon + \dot{q}_r'' + h(T_w - T_G) + \sigma(T_w^4 - T_G^4)
\]

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

\[
\dot{q}_r''/\epsilon + \sigma(T_w^4 - T^4)\]

For diagnostic purposes it is sometimes convenient to output the ‘INCIDENT HEAT FLUX’:

\[
\dot{q}_r''/\epsilon + \sigma T_w^4 + \dot{q}_c''
\]

All of the above heat flux output quantities are defined at a solid surface. However, ‘RADIATIVE HEAT FLUX GAS’ acts like a radiometer that is not attached to a solid surface. This quantity integrates the incoming radiative flux over \( 2\pi \) solid angles around the vector defined by ORIENTATION, a triplet of real numbers that defines the orientation of the device. For example, ORIENTATION=-1.0,0.0,0.0 means that the device points in the negative x direction. Unlike IOR, the ORIENTATION can be in any direction, not just those associated with the three coordinate directions.

Note that the sign of the output of heat flux is different than the sign of the input of a heat flux. A positive output quantity for heat flux means heat is being transferred into the surface.

14.3.6 Droplet Output Quantities

Droplet Quantities on Solid Surfaces

It is possible to record various properties of droplets and particles. Some of the output quantities are associated with solid boundaries. For example, ‘MPUA’ is the Mass Per Unit Area of the droplets named PART_ID. Likewise, ‘AMPUA’ is the Accumulated Mass Per Unit Area. Both of these are given in units of kg/m\(^2\). Think of these outputs as measures of the instantaneous mass density per unit area, and the accumulated total, respectively. 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 ‘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, ‘MPUA’ will be zero, but ‘AMPUA’ will not. FDS stores the droplet mass just before removing the droplet from the simulation for the purpose of saving CPU time.

In the test case bucket_test, a single sprinkler is mounted 10 cm below a 5 m ceiling. Water flows for 30 s at a constant rate of 180 L/min (ramped up and down in 1 s). The simulation continues for another 10 s to allow water drops time to reach the floor. The total mass of water discharged is

\[
180 \text{ L/min} \times 1 \text{ kg/L} \times \frac{1}{60} \text{ min/s} \times 30 \text{ s} = 90 \text{ kg}
\]
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 88.3 kg. Where is the missing water? Some droplets evaporate, and some droplets fly beyond the computational domain. 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)
       35 36
       Enter orientation: (plus or minus 1, 2 or 3)
       3
       Enter number of variables
       1
       Enter boundary file index for variable 1
       1
       Enter output file name:
       bucket_test_fds2ascii.csv
       Writing to file... bucket_test_fds2ascii.csv
```

Note that there really is no need to time-average the results. The quantity is inherently accumulating. Also, the “orientation” refers to direction of the surface. In this case, we’re interested in the positive $z$ direction, or 3.

**Droplet Mass and Fluxes in Gas Phase**

Away from solid surfaces, ‘MPUV’ is the Mass Per Unit Volume of the droplets as they fly through the air, in units of kg/m$^3$. ‘DROPLET FLUX X’, ‘DROPLET FLUX Y’, and ‘DROPLET 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.

**Local Spray Properties**

Detailed experimental measurements of sprays are often performed using Phase Doppler Particle Analysis (PDPA) which can be used to obtain information on the droplet size distribution, speed and concentration. Special device type has been defined to simulate the PDPA measurement. The actual quantity to measure, and the details of the measurement are defined using an associated Property. Note that in FDS, the PDPA device cannot produce complete droplet size distributions, but only various mean properties.
The PDPA device output at time $t$ is computed as

$$f(t) = \frac{1}{\min(t, t_e) - t_s} \int_{t_s}^{\min(t, t_e)} \left( \sum_i n_i d_i^m x_i \right) \frac{1}{(m-n)} dt$$  \hspace{1cm} (14.15)

where $x$ is the quantity to be measured, or just one in case of mean diameters. The summation goes over all the particles within a sphere with radius $PDPA\_RADIUS$ and centered at the location given by the device $XYZ$. The properties of the PDPA device are defined using the following keywords on the $PROP$ line:

**PART_ID** Name of the particle group to limit the computation to. Do not specify to account for all particles.

**PDPA\_START** $t_s$, starting time of time integration in seconds. PDPA output is always a running average over time. As the spray simulation may contain some initial transient phase, it may be useful to specify the starting time of data collection.

**PDPA\_END** $t_e$, ending time of time integration in seconds.

**PDPA\_M** $m$, exponent of diameter.

**PDPA\_N** $n$, exponent of diameter. In case $m = n$, the exponent $1/(m-n)$ is removed from the formula.

**PDPA\_RADIUS** Radius (m) of the sphere, centered at the device location, inside which the particles are monitored.

**QUANTITY** Particle property that is used for variable $x$. Possible variables are 'VELOCITY', 'U-VELOCITY', 'V-VELOCITY', 'W-VELOCITY', 'TEMPERATURE' and 'NUMBER CONCENTRATION'. Leave unspecified to compute mean diameters.

The following example is used to measure the Sauter mean diameter $D_{32}$ of the particle type 'water drops', starting from time 5.0 s.

```
&PID='pdpa_d32'
    PART_ID='water drops'
    PDPA_M=3
    PDPA_N=2
    PDPA_RADIUS=0.01
    PDPA_START=5.0 /
&DEV XYZ=0.0,0.0,1.0, QUANTITY='PDPA', PID='pdpa_d32' /
```

14.3.7 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 [21], the following equation can be solved via a simple iterative technique to determine an effective gas temperature, $T_{AST}$:

$$q''_r + q''_c = \varepsilon \sigma \left( T_{AST}^4 - T_w^4 \right) + h(T_{AST} - T_w)$$  \hspace{1cm} (14.16)
The sum \( \dot{q}_r'' + \dot{q}_c'' \) is the net heat flux onto the solid surface, whose temperature is \( T_w \). The heat fluxes and surface temperature are computed in FDS, and they are inter-dependent. The computed wall temperature affects the net heat flux and vice versa. However, because FDS only computes the solution to the 1-D heat conduction equation in the solid, it may be prone to error if lateral heat conduction within the solid is significant. Thus, in some scenarios neither the FDS-predicted heat fluxes or the surface temperature can be used as an accurate indicator of the thermal insult from the hot, smokey gases onto solid objects.

Of course, both the heat fluxes, \( \dot{q}_r'' \) and \( \dot{q}_c'' \), and the surface temperature, \( T_w \), can be passed from FDS to the other model, and suitable corrections can be made based on a presumably more accurate prediction of the solid temperature. Alternatively, the single quantity, \( T_{AST} \), can be transferred, as this is the temperature that the solid surface effectively “sees.” It represents the gas phase thermal environment, however complicated, but it does not carry along the uncertainty associated with the simple solid phase heat conduction model within FDS. Obviously, the objective in passing information to a more detailed model is to get a better prediction of the solid temperature (and ultimately its mechanical response) than FDS can provide.

### 14.3.8 Useful Solid Phase Outputs

In addition to the PROFILE output, there are various additional quantities that are useful for monitoring reacting surfaces. For example, ‘WALL THICKNESS’ gives the overall thickness of the solid surface element. ‘SURFACE DENSITY’ gives the overall mass per unit area for the solid surface element, computed as an integral of material density over wall thickness. Both quantities are available both as DEVC and BNDF.

To record the change in a material component’s density with time, use the output quantity ‘SOLID DENSITY’ in the following way:

```
&DEVC ID='...', XYZ=..., IOR=3, QUANTITY='SOLID DENSITY', MATL_ID='wood', DEPTH=0.001 /
```

This produces a time history of the density of the material referred to as ‘wood’ on a MATL line. The density is recorded 1 mm beneath the surface which is oriented in the positive z direction. Note that if ‘wood’ is part of a mixture, the density represents the mass of ‘wood’ per unit volume of the mixture. Note also that ‘SOLID DENSITY’ is only available as a DEVC (device) quantity.

### 14.3.9 Fractional Effective Dose (FED)

The Fractional Effective Dose index (FED), developed by Purser [22], is a commonly used measure of human incapacitation due to the combustion gases. The present version of FDS uses only the concentrations of the gases CO, CO\(_2\), and O\(_2\) to calculate the FED value as

\[
FED_{tot} = FED_{CO} \times HV_{CO_2} + FED_{O_2}
\]  

The fraction of an incapacitating dose of CO is calculated as

\[
FED_{CO} = 4.607 \times 10^{-7} \times (CO)^{1.036} \times t
\]  

where \( t \) is time in seconds and \( C_{CO} \) is the CO concentration (ppm). The fraction of an incapacitating dose of low O\(_2\) hypoxia is calculated as

\[
FED_{O_2} = \frac{t}{60 \exp [8.13 - 0.54(20.9 - C_{O_2})]}
\]  

where \( C_{O_2} \) is the O\(_2\) concentration (volume per cent). The hyperventilation factor induced by carbon dioxide is calculated as

\[
HV_{CO_2} = \frac{\exp(0.1930C_{CO_2} + 2.0004)}{7.1}
\]
where \( C_{\text{CO}_2} \) is the \( \text{CO}_2 \) concentration (percent).

Note that the spatial integration features (Section 14.3.10) cannot be used with FED output because FED makes use of the \text{TIME INTEGRAL} (Section 14.3.11). For the same reason, FED output is only available as a point measurement.

### 14.3.10 Spatially-Integrated Outputs

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

```plaintext
&DEVC XB=2.3,4.5,2.8,6.7,3.6,7.8, QUANTITY='TEMPERATURE', ID='maxT', STATISTICS='MAX' /
```

causes FDS to write out the maximum gas phase temperature over the volume bounded by XB. Note that it does not compute the maximum over the entire computational domain, just the specified volume, and this volume must lie within a single mesh. Other \text{STATISTICS} are discussed below. Note that some are appropriate for gas phase output quantities, some for solid phase, and some for both.

For solid phase output quantities, like heat fluxes and surface temperatures, the specification of a \text{SURF_ID} along with the appropriate statistic limits the calculation to only those surfaces. You can further limit the search by using the sextuplet of coordinates XB to force FDS to only compute statistics for surface cells within the given volume. Be careful to account for the fact that the solid surface might shift to conform to the underlying numerical grid. Also, be careful not to specify a volume that extends beyond a single mesh. Note that you do not (and should not) specify an orientation via the parameter \text{IOR} when using a spatial statistic. \text{IOR} is only needed to find a specific point on the solid surface.

Use the \text{STATISTICS} feature with caution because it demands that FDS evaluate the given \text{QUANTITY} in all gas or solid phase cells.

#### Minimum or Maximum Value

For a given gas phase scalar output quantity defined at the center of each grid cell, \( \phi_{ijk} \), \text{STATISTICS}='MIN' or \text{STATISTICS}='MAX' computes the minimum or maximum value, respectively

\[
\min_{ijk} \phi_{ijk} \quad ; \quad \max_{ijk} \phi_{ijk}
\]

over the cells that are included in the specified volume bounded by XB. Note that this statistic is only appropriate for gas phase quantities. Note also that you must specify a volume to sum over via the coordinate parameters, XB, all of which must be contained within the same mesh.

#### Average Value

For a given gas phase scalar output quantity defined at the center of each grid cell, \( \phi_{ijk} \), \text{STATISTICS}='MEAN' computes the average value,

\[
\frac{1}{N} \sum_{ijk} \phi_{ijk}
\]

over the cells that are included in the specified volume bounded by XB. Note that this statistic is only appropriate for gas phase quantities. Note also that you must specify a volume to sum over via the coordinate parameters, XB, all of which must be contained within the same mesh.
Volume-Weighted Mean

For a given gas phase output quantity, \( \phi(x,y,z) \), \text{STATISTICS}='VOLUME MEAN' produces the discrete analog of

\[
\frac{1}{V} \int \phi(x,y,z) \, dx \, dy \, dz
\]  

(14.23)

which is very similar to 'MEAN', but it weights the values according to the relative size of the mesh cell. Note that this statistic is only appropriate for gas phase quantities. Note also that you must specify a volume to sum over via the coordinate parameters, \( \text{XB} \), all of which must be contained within the same mesh.

Mass-Weighted Mean

For a given gas phase output quantity, \( \phi(x,y,z) \), \text{STATISTICS}='MASS MEAN' produces the discrete analog of

\[
\frac{\int \rho(x,y,z) \phi(x,y,z) \, dx \, dy \, dz}{\int \rho \, dx \, dy \, dz}
\]  

(14.24)

which is similar to 'VOLUME MEAN', but it weights the values according to the relative mass of the mesh cell. Note that this statistic is only appropriate for gas phase quantities. Note also that you must specify a volume to sum over via the coordinate parameters, \( \text{XB} \), all of which must be contained within the same mesh.

Volume Integral

For a given gas phase output quantity, \( \phi(x,y,z) \), \text{STATISTICS}='VOLUME INTEGRAL' produces the discrete analog of

\[
\int \phi(x,y,z) \, dx \, dy \, dz
\]  

(14.25)

Note that this statistic is only appropriate for gas phase quantities, in particular those whose units involve \( m^{-3} \). For example, heat release rate per unit volume is an appropriate output quantity. Note also that you must specify a volume to sum over via the coordinate parameters, \( \text{XB} \), all of which must be contained within the same mesh.

Mass Integral

For a given gas phase output quantity, \( \phi(x,y,z) \), \text{STATISTICS}='MASS INTEGRAL' produces the discrete analog of

\[
\int \rho(x,y,z) \phi(x,y,z) \, dx \, dy \, dz
\]  

(14.26)

Note that this statistic is only appropriate for gas phase quantities. Note also that you must specify a volume to sum over via the coordinate parameters, \( \text{XB} \), all of which must be contained within the same mesh.

Area Integral

For a given gas phase output quantity, \( \phi(x,y,z) \), \text{STATISTICS}='AREA INTEGRAL' produces the discrete analog of

\[
\int \phi(x,y,z) \, dA
\]  

(14.27)

where \( dA \) depends on the coordinates you specify for \( \text{XB} \). Note that this statistic is only appropriate for gas phase quantities, in particular those whose units involve \( m^{-2} \). For example, the quantity 'MASS FLUX X'
along with \texttt{SPEC\_ID='my gas'} is an appropriate output quantity if you want to know the mass flux of the gas species that you have named ‘my gas’ through an area normal to the \(x\) direction. Note also that you must specify an area to sum over via the coordinate parameters, \(XB\), all of which must be contained within the same mesh.

**Surface Integral**

For a given solid phase output quantity, \(\phi\), \texttt{STATISTICS='SURFACE INTEGRAL'} produces the discrete analog of
\[
\int \phi \, dA
\]
(14.28)

Note that this statistic is only appropriate for solid phase quantities, in particular those whose units involve \(m^{-2}\). For example, the various heat and mass fluxes are appropriate output quantities.

**Volume, Mass, and Heat Flow**

The net flow of mass and energy into or out of compartments can be useful for many applications. There are several outputs that address these. All are prescribed via the device (\texttt{DEVC}) namelist group only. For example:

\texttt{&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:

\[
\dot{V} = \int \mathbf{u} \cdot d\mathbf{S}
\]
\[
\dot{m} = \int \rho \mathbf{u} \cdot d\mathbf{S}
\]
\[
\dot{q} = \int c_p \rho (T - T_\infty) \mathbf{u} \cdot d\mathbf{S}
\]

The addition of a + or - to the \texttt{QUANTITY} names yields the integral of the flow in the positive or negative direction only. In other words, if you want to know the mass flow out of a compartment, use ‘MASS FLOW +’ or ‘MASS FLOW -’, depending on the orientation of the door.

The quantities ‘MASS FLOW’ and ‘HEAT FLOW’ should not be applied at a solid boundary.

**14.3.11 Temporally-Integrated Outputs**

In addition to the spatial statistics, a time integral of an \texttt{DEVC} output can be computed by specifying \texttt{STATISTICS = 'TIME INTEGRAL'} on the \texttt{DEVC} line. This produces a discrete analog of
\[
\int_{t_0}^t \phi(\tau) \, d\tau
\]
(14.29)

Note that the spatial and time integrals cannot be used simultaneously.
14.3.12 Wind and the Pressure Coefficient

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

\[ C_p = \frac{p - p_\infty}{\frac{1}{2} \rho_\infty U^2} \]  \hspace{1cm} (14.30)

\( p_\infty \) is the ambient, or “free stream” pressure, and \( \rho_\infty \) is the ambient density. The parameter \( U \) is the free-stream wind speed, given as **CHARACTERISTIC_VELOCITY** on the **PROP** line.

\[ \&BNDF \text{ QUANTITY='PRESSURE COEFFICIENT', PROP_ID='whatever'} / \]
\[ \&DEVC \text{ ID='pressure tap', XYZ=..., IOR=2, QUANTITY='PRESSURE COEFFICIENT', PROP_ID='whatever'} / \]
\[ \&PROP \text{ ID='whatever', CHARACTERISTIC VELOCITY=3.4} / \]

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.

14.3.13 Dry Volume and Mass Fractions

During actual experiments, species such as CO and CO\(_2\) are typically measured “dry”; that is, the water vapor is removed from the gas sample prior to analysis. For easier comparison of FDS predictions with measured data, you can specify the logical parameter **DRY on a DEVC line that reports the ’MASS FRACTION’ or ’VOLUME FRACTION’ of a species when using the mixture fraction combustion model. For example, the first line reports the actual volume fraction of CO, and the second line reports the output of a gas analyzer in a typical experiment.

\[ \&DEVC \text{ ID='wet CO', XYZ=..., QUANTITY='VOLUME FRACTION', SPEC_ID='carbon monoxide'} / \]
\[ \&DEVC \text{ ID='dry CO', XYZ=..., QUANTITY='VOLUME FRACTION', SPEC_ID='carbon monoxide', DRY=.TRUE.} / \]

14.3.14 Gas Velocity

The gas velocity components, \( u \), \( v \), and \( w \), can be output as slice (**SLCF**), point device (**DEVC**), isosurface (**ISOF**), or Plot3D quantities using the names ’U-VELOCITY’, ’V-VELOCITY’, and ’W-VELOCITY’. The total velocity is specified as just ’VELOCITY’. Normally, the velocity is always positive, but you can use the parameter **VELO_INDEX** with a value of 1, 2 or 3 to indicate that the velocity ought to have the same sign as \( u \), \( v \), or \( w \), respectively. This is handy if you are comparing velocity predictions with measurements. For Plot3D files, add **PLOT3D_VELO_INDEX**(N)=... to the **DUMP** line, where N refers to the Plot3D quantity 1, 2, 3, 4 or 5.

14.3.15 A Posteriori Mesh Quality Metrics

The quality of a particular simulation is most directly tied to grid resolution. Three slice file output quantities are suggested here for measuring errors in the velocity and scalar fields:

1. A model for the fraction of unresolved kinetic energy called the **measure of turbulence resolution** (similar to what is often called the ‘Pope criterion’ [23]), MTR

2. A model for the fraction of unresolved scalar energy fluctuations called **measure of scalar resolution** [24], MSR

3. A **wavelet-based error measure** [25], WEM

Examples:
Note that an additional scalar QUANTITY2 is required for MSR and WEM. QUANTITY2 may be any output quantity appropriate for SLCF. Also, CELL_CENTERED is optional for any of the three metrics.

**Measure of Turbulence Resolution**

In FDS, the user may output a scalar quantity which we refer to as the measure of turbulence resolution, defined locally as

$$MTR(x,t) = \frac{k_{sgs}}{k_{res} + k_{sgs}}$$  \hspace{1cm} (14.31)

where

$$k_{res} = \frac{1}{2} \tilde{u}_i \tilde{u}_i$$ \hspace{1cm} (14.32)

$$k_{sgs} = \frac{1}{2} (\tilde{u}_i - \hat{\tilde{u}})(\tilde{u}_i - \hat{\tilde{u}})$$ \hspace{1cm} (14.33)

Here, \(\tilde{u}_i\) is the resolved LES velocity and \(\hat{\tilde{u}}\) is test filtered at a scale 2\(\Delta\) where \(\Delta\) is the LES filter width (in FDS, \(\Delta = \delta x\)). The model for the SGS fluctuations is taken from scale similarity [26]. Cross-term energy is ignored. The basic idea is to provide the user with an approximation to the Pope criterion [23], \(M\), which is easily accessible in Smokeview (the FDS visualization tool). In Smokeview, the user may readily time average MTR in a specified plane. The time average of MTR is a reasonable estimate of \(M\). The measure falls within the range [0,1], with 0 indicating perfect resolution and 1 indicating poor resolution. The concept is illustrated in Figure 14.1. Notice that on the left the difference between the grid signal and the test signal is very small. On the right, the grid signal is highly turbulent and the corresponding test signal is much smoother. We infer then that the flow is under-resolved.

![Figure 14.1](image)

**Figure 14.1:** (Left) Resolved signal, MTR is small. (Right) Unresolved signal, MTR is close to unity.

For the canonical case of isotropic turbulence Pope actually defines LES such that \(M < 0.2\). That is, LES requires resolution of 80% of the kinetic energy in the flow field (because this puts the grid Nyquist limit within the inertial subrange). The question remains as to whether this critical value is sufficient or necessary for a given engineering problem. As shown in [25], maintaining mean values of MTR near 0.2 indeed provides satisfactory results (simulation results within experimental error bounds) for mean velocities and species concentrations in nonreacting, buoyant plumes.
Measure of Scalar Resolution

The measure of turbulence resolution is defined locally as

\[ \text{MSR}(x,t) = \frac{T_{\text{sgs}}}{T_{\text{res}} + T_{\text{sgs}}} \]  

(14.34)

where

\[ T_{\text{res}} = \bar{\phi}^2 \]  

(14.35)

\[ T_{\text{sgs}} = (\bar{\phi} - \hat{\phi})^2 \]  

(14.36)

Here again the model for the SGS scalar energy fluctuations is taken from scale similarity [26]. The field \( \hat{\phi} \) is test filtered at a scale \( 2\Delta \). The cross-term energy (i.e. \( \langle 2\phi\phi' \rangle \), where \( \phi' = \phi - \bar{\phi} \)) is ignored, but this does not affect the bounds of the measure. Further, it can be shown that this term is small if sufficient resolution is used. There is evidence to suggest that the requirements for scalar resolution may be somewhat more stringent than for the velocity field [24]. Therefore, currently the best advice is to keep the mean value of MSR less than 0.2.

Wavelet Error Measure

We begin by providing background on the simple Haar wavelet [27]. For a thorough and more sophisticated review of wavelet methods, the reader is referred to Schneider and Vasilyev [28].

Suppose the scalar function \( f(r) \) is sampled at discrete points \( r_j \), separated by a distance \( h \), giving values \( s_j \). Defining the unit step function over the interval \( [r_1, r_2] \) by

\[ \psi_{[r_1, r_2]}(r) = \begin{cases} 1 & \text{if } r_1 \leq r < r_2 \\ 0 & \text{otherwise} \end{cases} \]  

(14.37)

the simplest possible reconstruction of the signal is the step function approximation

\[ f(r) \approx \sum_j s_j \psi_{[r_j, r_j+h]}(r) \]  

(14.38)

By “viewing” the signal at a coarser resolution, say \( 2h \), an identical reconstruction of the function \( f \) over the interval \( [r_j, r_j + 2h] \) may be obtained from

\[ f_{[r_j, r_j+2h]}(r) = \sum_a^{r_j+s_{j+1}} a \psi_{[r_j, r_j+2h]}(r) + \sum_c^{r_j-s_{j+1}} c \psi_{[r_j, r_j+2h]}(r) \]  

(14.39)

where \( a \) is as the average coefficient and \( c \) is as the wavelet coefficient. The Haar mother wavelet (Figure 14.2) is identified as

\[ \psi_{[r_1, r_2]}(r) = \begin{cases} 1 & \text{if } r_1 \leq r < \frac{1}{2}(r_1 + r_2) \\ -1 & \text{if } \frac{1}{2}(r_1 + r_2) \leq r < r_2 \end{cases} \]  

(14.40)

The decomposition of the signal shown in (14.39) may be repeated at ever coarser resolutions. The result is a wavelet transform. The procedure is entirely analogous to the Fourier transform, but with compact support. If we look at a 1D signal with \( 2^m \) points, the repeated application of (14.39) results in an \( m \times m \) matrix of averages \( a \) with components \( a_{ij} \) and an \( m \times m \) wavelet coefficient matrix \( c \) with components \( c_{ij} \). Each row \( i \) of \( a \) may be reconstructed from the \( i+1 \) row of \( a \) and \( c \). Because of this and because small values of the wavelet coefficient matrix may be discarded, dramatic compression of the signal may be obtained.
Here we are interested in using the wavelet analysis to say something about the local level of error in grid resolution. Very simply, we ask what can be discerned from a sample of four data points along a line. Roughly speaking we might expect to see one of the four scenarios depicted in Figure 14.3. Within each plot window we also show the results of a Haar wavelet transform for that signal. Looking first at the two top plots, on the left we have a step function and on the right we have a straight line. Intuitively, we expect large error for the step function and small error for the line. The following error measure achieves this goal:

\[
WEM(x,t) = \max_{x,y,z} \left( \frac{|c_{11} + c_{12}| - |c_{21}|}{|a_{21}|} \right)
\]

(14.41)

Note that we have arbitrarily scaled the measure so that a step function leads to \(WEM = 1\). In practice the transform is performed in all coordinate directions and the max value is reported. The scalar value may be output to Smokeview at the desired time interval.

Looking now at the two plots on the bottom of Figure 14.3, the signal on the left, which may indicate spurious oscillations or unresolved turbulent motion, leads to \(WEM = 2\) (note that this limit differs from the upper bound of unity for MTR and MSR). Our measure therefore views this situation as the worst case in a sense. The signal to the lower right is indicative of an extremum, which actually is easily resolved by most centered spatial schemes and results again in \(WEM = 0\).

In [25], the time average of WEM was reported for LES of a nonreacting buoyant plume at three grid resolutions. From this study, the best advice currently is to maintain average values of WEM less than 0.5.
Figure 14.3: Averages and coefficients for local Haar wavelet transforms on four typical signals.
14.4 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 \texttt{fds2ascii.f90}, with a PC compiled version called \texttt{fds2ascii.exe}. To run the program, just type

\texttt{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 \texttt{CHID_fds2ascii.csv}. A typical command line session looks like this:

```
>> fds2ascii
    Enter Job ID string (CHID): bucket_test
    What type of file to parse?
      PL3D file? Enter 1
      SLCF file? Enter 2
      BNDF file? Enter 3
      3
    Enter Sampling Factor for Data?
      (1 for all data, 2 for every other point, etc.)
      1
    Limit the domain size? (y or n)
      y
    Enter min/max x, y and z
      -5 5 -5 5 0 1
      1 MESH 1, WALL TEMPERATURE
    Enter starting and ending time for averaging (s)
      35 36
    Enter orientation: (plus or minus 1, 2 or 3)
      3
    Enter number of variables
      1
    Enter boundary file index for variable 1
      1
    Enter output file name:
      bucket_test_fds2ascii.csv
    Writing to file... bucket_test_fds2ascii.csv
```

These commands tell \texttt{fds2ascii} that you want to convert (binary) boundary file data into a text file. You want to sample every data point within the specified volume, you want only those surfaces that point upwards (+3 orientation), you only want 1 variable (only one is listed anyway and its index is 1 – that is just the number used to list the available files). The data will be time-averaged, and it will be output to a file listed at the end of the session.
14.5 Summary of Frequently-Used Output Quantities

Table 14.1, spread over the following pages, summarizes the various Output Quantities. The column “File Type” lists the allowed output files for the quantities. “B” is for Boundary (BNDF), “D” is for Device (DEVC), “I” is for Iso-surface (ISOF), “P” is for Plot3D, “PA” for PArticle (PART), “S” is for Slice (SLCF). Be careful when specifying complicated quantities for Iso-surface or Plot3D files, as it requires computation in every gas phase cell.

For those output quantities that require a species name via SPEC_ID, the species implicitly defined when doing a mixture fraction calculation are as follows:

- fuel
- oxygen
- nitrogen
- water vapor
- carbon dioxide
- carbon monoxide
- hydrogen
- soot
- other

As an example of how to use the species names, suppose you want to calculate the integrated mass flux of carbon monoxide through a horizontal plane, like the total amount entrained in a fire plume. Use a “device” as follows:

```
&DEVC ID='CO_flow', XB=-5,5,-5,5,2,2, QUANTITY='MASS FLUX Z', SPEC_ID='carbon monoxide', STATISTICS='AREA INTEGRAL' /
```

Here, the ID is just a label in the output file. The use of lower case letters for the SPEC_ID indicates that you want to record the mass flow of the CO that is implicitly defined by the mixture fraction variables. This is the default combustion model in FDS. If you have defined a species explicitly via a SPEC line, you can specify that instead.

The format of certain output quantities changed, starting with FDS version 5.2, but the older names originating with version 5.0 will still be accepted. The new convention is that when an output quantity is related to a particular gas species or particle type, then you must specify the appropriate SPEC_ID or PART_ID on the same input line. Also note that the use of underscores in output quantity names has been eliminated – just remember that all output quantity names ought to be in single quotes.
Table 14.1: Summary of frequently used output quantities.

<table>
<thead>
<tr>
<th>QUANTITY</th>
<th>Symbol</th>
<th>Units</th>
<th>File Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSORPTION COEFFICIENT</td>
<td>κ</td>
<td>/m</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>ACTUATED SPRINKLERS</td>
<td></td>
<td>Number of actuated sprinklers</td>
<td>D</td>
</tr>
<tr>
<td>ADIABATIC SURFACE TEMPERATURE</td>
<td></td>
<td>See Section 14.3.7</td>
<td>B,D</td>
</tr>
<tr>
<td>AMPUAA**</td>
<td></td>
<td>See Section 14.3.6</td>
<td>B,D</td>
</tr>
<tr>
<td>AverAge SPECIFIC HEAT</td>
<td>κ_p</td>
<td>kJ/kg/K</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>Back WALL TEMPERATURE</td>
<td></td>
<td>See Section 14.2.2</td>
<td>B,D</td>
</tr>
<tr>
<td>Burning RATE</td>
<td>m_f</td>
<td>kg/m²/s</td>
<td>B,D</td>
</tr>
<tr>
<td>Chamber OBSCURATION</td>
<td></td>
<td>See Section 13.3.4</td>
<td></td>
</tr>
<tr>
<td>ConductIVITY</td>
<td>k</td>
<td>W/m/k</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>CONTROL</td>
<td></td>
<td>See Section 13.5</td>
<td>D</td>
</tr>
<tr>
<td>Convective HEAT FLUX</td>
<td>q''_c</td>
<td>kW/m²</td>
<td>B,D</td>
</tr>
<tr>
<td>CPUA**</td>
<td></td>
<td>See Section 14.3.6</td>
<td>B,D</td>
</tr>
<tr>
<td>CPU TIME</td>
<td></td>
<td>Elapsed CPU time</td>
<td>s, D</td>
</tr>
<tr>
<td>DENSITY</td>
<td>ρ or ρ_Yα with SPEC_ID</td>
<td>kg/m³</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>Divergence</td>
<td>V·u</td>
<td>s⁻¹</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>Droplet FLUX X**</td>
<td></td>
<td>See Section 14.3.6</td>
<td>P,S</td>
</tr>
<tr>
<td>Droplet FLUX Y**</td>
<td></td>
<td>See Section 14.3.6</td>
<td>P,S</td>
</tr>
<tr>
<td>Droplet FLUX Z**</td>
<td></td>
<td>See Section 14.3.6</td>
<td>P,S</td>
</tr>
<tr>
<td>Droplet DIAMETER</td>
<td>2r_d</td>
<td>μm</td>
<td>PA</td>
</tr>
<tr>
<td>Droplet VELOCITY</td>
<td></td>
<td>[u_d]</td>
<td></td>
</tr>
<tr>
<td>Droplet TEMPERATURE</td>
<td>T_d</td>
<td>°C</td>
<td>PA</td>
</tr>
<tr>
<td>Droplet MASS</td>
<td>m_d</td>
<td>kg</td>
<td>PA</td>
</tr>
<tr>
<td>Droplet AGE</td>
<td>t_d</td>
<td>s</td>
<td>PA</td>
</tr>
<tr>
<td>Enthalpy</td>
<td></td>
<td>Σ ρ_Yα∫₀ T' c_p,α(T') dT'</td>
<td>kJ/m³</td>
</tr>
<tr>
<td>Fed</td>
<td></td>
<td>See Section 14.3.9</td>
<td>D</td>
</tr>
<tr>
<td>Gauge HEAT FLUX</td>
<td></td>
<td>See Section 14.3.5</td>
<td>B,D</td>
</tr>
<tr>
<td>H</td>
<td></td>
<td>H =</td>
<td>u</td>
</tr>
<tr>
<td>heat FLOW</td>
<td></td>
<td>See Section 14.3.10</td>
<td>kW</td>
</tr>
<tr>
<td>Net heat flux</td>
<td></td>
<td>See Section 14.3.5</td>
<td>kW/m²</td>
</tr>
<tr>
<td>HRR</td>
<td></td>
<td>f q'''' dV</td>
<td>kW</td>
</tr>
<tr>
<td>HRRPVUV</td>
<td></td>
<td>q''''</td>
<td>kW/m³</td>
</tr>
<tr>
<td>incident heat flux</td>
<td></td>
<td>See Section 14.3.5</td>
<td>kW/m²</td>
</tr>
<tr>
<td>inside wall temperature</td>
<td></td>
<td>See Section 14.2.2</td>
<td>°C</td>
</tr>
<tr>
<td>iteration</td>
<td></td>
<td>Number of time steps</td>
<td>D</td>
</tr>
<tr>
<td>Layer height</td>
<td></td>
<td>See Section 14.3.3</td>
<td>m</td>
</tr>
<tr>
<td>link temperature</td>
<td></td>
<td>See Section 13.3.3</td>
<td>°C</td>
</tr>
<tr>
<td>lower temperature</td>
<td></td>
<td>See Section 14.3.3</td>
<td>°C</td>
</tr>
<tr>
<td>mass flow</td>
<td></td>
<td>See Section 14.3.10</td>
<td>kg/s</td>
</tr>
<tr>
<td>mass flux</td>
<td></td>
<td>Mass flux at solid surface</td>
<td>kg/m²/s</td>
</tr>
<tr>
<td>mass flux x*</td>
<td></td>
<td>ρu_Yα</td>
<td>kg/m²/s</td>
</tr>
<tr>
<td>mass flux y*</td>
<td></td>
<td>ρv_Yα</td>
<td>kg/m²/s</td>
</tr>
<tr>
<td>mass flux z*</td>
<td></td>
<td>ρw_Yα</td>
<td>kg/m²/s</td>
</tr>
</tbody>
</table>
Table 14.1: Summary of frequently used output quantities (continued).

<table>
<thead>
<tr>
<th>QUANTITY</th>
<th>Symbol</th>
<th>Units</th>
<th>File Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASS FRACTION</td>
<td>$Y_\alpha$</td>
<td>kg/kg</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>MIXTURE FRACTION</td>
<td>$Z$</td>
<td>kg/kg</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>MPUA*</td>
<td></td>
<td>kg/m$^2$</td>
<td>B,D</td>
</tr>
<tr>
<td>MPUV**</td>
<td></td>
<td>kg/m$^3$</td>
<td>D,P,S</td>
</tr>
<tr>
<td>NORMAL VELOCITY</td>
<td></td>
<td>m/s</td>
<td>D,B</td>
</tr>
<tr>
<td>OPEN NOZZLES</td>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>OPTICAL DENSITY</td>
<td>$K/2.3$ (Section 14.3.2)</td>
<td>l/m</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>EXTINCTION COEFFICIENT</td>
<td>$K$ (Section 14.3.2)</td>
<td>l/m</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>PATH OBSCURATION</td>
<td></td>
<td>%</td>
<td>D</td>
</tr>
<tr>
<td>PRESSURE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRESSURE COEFFICIENT</td>
<td>$C_p$ (Section 14.3.12)</td>
<td>kW/m$^2$</td>
<td>B,D</td>
</tr>
<tr>
<td>PRESSURE ZONE</td>
<td></td>
<td></td>
<td>D,S</td>
</tr>
<tr>
<td>RADIATIVE HEAT FLUX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RADIATIVE HEAT FLUX GAS</td>
<td></td>
<td>kW/m$^2$</td>
<td>D</td>
</tr>
<tr>
<td>RADIOMETER</td>
<td></td>
<td>kW/m$^2$</td>
<td>B,D</td>
</tr>
<tr>
<td>RELATIVE HUMIDITY</td>
<td></td>
<td>%</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>SCALAR RESOLUTION****</td>
<td></td>
<td></td>
<td>S</td>
</tr>
<tr>
<td>SOOT VOLUME FRACTION</td>
<td>$\rho Y_s (Z)/\rho_s$</td>
<td>mol/mol</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>SPECIFIC ENTHALPY</td>
<td>$\Sigma Y_\alpha \int_T^0 c_{p,\alpha}(T') dT'$</td>
<td>kJ/kg</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>SPECIFIC HEAT</td>
<td>$c_p$</td>
<td>kJ/kg/K</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>SPRINKLER LINK TEMPERATURE</td>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>SOLID DENSITY</td>
<td></td>
<td>kg/m$^3$</td>
<td>D</td>
</tr>
<tr>
<td>SURFACE DENSITY</td>
<td></td>
<td>kg/m$^3$</td>
<td>B,D</td>
</tr>
<tr>
<td>TEMPERATURE</td>
<td>$T$ (Section 14.3.4)</td>
<td>°C</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>THERMOCOUPLE</td>
<td>$T_{TC}$ (Section 14.3.4)</td>
<td>°C</td>
<td>D</td>
</tr>
<tr>
<td>TIME</td>
<td>$t$ (Section 13.1)</td>
<td>s</td>
<td>D</td>
</tr>
<tr>
<td>TIME STEP</td>
<td>$\delta t$, Numerical time step</td>
<td>s</td>
<td>D</td>
</tr>
<tr>
<td>TURBULENCE RESOLUTION</td>
<td></td>
<td></td>
<td>D,S</td>
</tr>
<tr>
<td>U-VELOCITY</td>
<td>$u$</td>
<td>m/s</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>V-VELOCITY</td>
<td>$v$</td>
<td>m/s</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>W-VELOCITY</td>
<td>$w$</td>
<td>m/s</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>UPPER TEMPERATURE</td>
<td></td>
<td>°C</td>
<td>D</td>
</tr>
<tr>
<td>VELOCITY***</td>
<td>$\sqrt{u^2 + v^2 + w^2}$</td>
<td>m/s</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>VISCOSITY</td>
<td>$\mu$</td>
<td>kg/m/s</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>VISIBILITY</td>
<td>$S = C/K$ (Section 14.3.2)</td>
<td>m</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>VOLUME FLOW</td>
<td></td>
<td>m$^3$/s</td>
<td>D</td>
</tr>
<tr>
<td>VOLUME FRACTION****</td>
<td>$X_\alpha$</td>
<td>mol/mol</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>WALL CLOCK TIME</td>
<td></td>
<td></td>
<td>s</td>
</tr>
<tr>
<td>WALL TEMPERATURE</td>
<td>$T_w$</td>
<td>°C</td>
<td>B,D</td>
</tr>
<tr>
<td>WALL THICKNESS</td>
<td></td>
<td></td>
<td>B,D</td>
</tr>
<tr>
<td>WAVELET ERROR****</td>
<td></td>
<td></td>
<td>S</td>
</tr>
</tbody>
</table>
Quantity requires the specification of a gas species using SPEC_ID.

Quantity requires the specification of a particle name using PART_ID.

Add VELO_INDEX=1 to the input line if you want to multiply the velocity by the sign of $u$. Use the indices 2 and 3 for $v$ and $w$, respectively.

Quantity requires the specification of a gas species using SPEC_ID. Do not use for MIXTURE FRACTION.

Quantity requires specification of an additional scalar using QUANTITY2.
### 14.6 Summary of Infrequently-Used Output Quantities

Table 14.2 below lists some less often used output quantities. These are mainly used for diagnostic purposes. Explanations for most can be found in Volume 1 of the FDS Technical Reference Guide [1].

<table>
<thead>
<tr>
<th>QUANTITY</th>
<th>Symbol</th>
<th>Units</th>
<th>File Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td></td>
<td>µm</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>ADT</td>
<td></td>
<td>°C</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>DROPLET PHASE</td>
<td></td>
<td></td>
<td>PA</td>
</tr>
<tr>
<td>EMISSIVITY</td>
<td></td>
<td></td>
<td>B,D</td>
</tr>
<tr>
<td>F_X, F_Y, F_Z</td>
<td></td>
<td>m/s²</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>GAS TEMPERATURE</td>
<td></td>
<td>°C</td>
<td>B,D</td>
</tr>
<tr>
<td>HEAT TRANSFER COEFFICIENT</td>
<td></td>
<td></td>
<td>B,D</td>
</tr>
<tr>
<td>HRRPUL</td>
<td></td>
<td>kW/m²</td>
<td>D</td>
</tr>
<tr>
<td>INTEGRATED INTENSITY</td>
<td></td>
<td>kW/m²</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>KINETIC ENERGY</td>
<td></td>
<td>(m/s)²</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>MAXIMUM VELOCITY ERROR</td>
<td></td>
<td>s</td>
<td>D</td>
</tr>
<tr>
<td>MIXING TIME</td>
<td></td>
<td></td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>PDPA</td>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>PRESSURE ITERATIONS</td>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>RADIATION LOSS</td>
<td></td>
<td>kW/m³</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>STRAIN RATE X</td>
<td></td>
<td>l/s</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>STRAIN RATE Y</td>
<td></td>
<td>l/s</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>STRAIN RATE Z</td>
<td></td>
<td>l/s</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>VORTICITY X</td>
<td></td>
<td>l/s</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>VORTICITY Y</td>
<td></td>
<td>l/s</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>VORTICITY Z</td>
<td></td>
<td>l/s</td>
<td>D,I,P,S</td>
</tr>
<tr>
<td>PARTICLE RADIATION LOSS</td>
<td></td>
<td>kW/m³</td>
<td>D,I,P,S</td>
</tr>
</tbody>
</table>

Table 14.2: Summary of *infrequently* used output quantities.
14.7 Summary of HVAC Output Quantities

Table 14.3 summarizes the various Output Quantities for HVAC systems. Quantities for a duct require the specification of a DUCT_ID, and quantities for a node require the specification of a NODE_ID. Mass and volume fraction outputs also require the specification of a SPEC_ID.

<table>
<thead>
<tr>
<th>QUANTITY</th>
<th>Symbol</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUCT DENSITY</td>
<td>Density of the flow in a duct</td>
<td>kg/m³</td>
</tr>
<tr>
<td>DUCT MASS FLOW</td>
<td>Mass flow in a duct</td>
<td>kg/s</td>
</tr>
<tr>
<td>DUCT MASS FRACTION</td>
<td>Mass fraction of a species in a duct</td>
<td>kg/kg</td>
</tr>
<tr>
<td>DUCT TEMPERATURE</td>
<td>Temperature of the flow in a duct</td>
<td>°C</td>
</tr>
<tr>
<td>DUCT VELOCITY</td>
<td>Velocity of a duct</td>
<td>m/s</td>
</tr>
<tr>
<td>DUCT VOLUME FLOW</td>
<td>Volumetric flow in a duct</td>
<td>m³/s</td>
</tr>
<tr>
<td>DUCT VOLUME FRACTION</td>
<td>Volume fraction of a species in a duct</td>
<td>mol/mol</td>
</tr>
<tr>
<td>NODE DENSITY</td>
<td>Density of the flow through a node</td>
<td>kg/m³</td>
</tr>
<tr>
<td>NODE MASS FRACTION</td>
<td>Mass fraction of a species in a node</td>
<td>kg/kg</td>
</tr>
<tr>
<td>NODE Pressure</td>
<td>Pressure of a node</td>
<td>Pa</td>
</tr>
<tr>
<td>NODE TEMPERATURE</td>
<td>Temperature of the flow through a node</td>
<td>°C</td>
</tr>
<tr>
<td>NODE VOLUME FRACTION</td>
<td>Volume fraction of a species in a node</td>
<td>mol/mol</td>
</tr>
</tbody>
</table>
Chapter 15

Alphabetical List of Input Parameters

This Appendix lists all of the input parameters for FDS in separate tables grouped by Namelist, these tables
are in alphabetical order along with the parameters within them. This is intended to be used as a quick
reference and does not replace reading the detailed description of the parameters in the main body of this
guide. See Table 5.1 for a cross-reference of relevant sections and the tables in this Appendix. The reason
for this statement is that many of the listed parameters are mutually exclusive – specifying more than one
can cause the program to either fail or run in an unpredictable manner. Also, some of the parameters trigger
the code to work in a certain mode when specified. For example, specifying the thermal conductivity of a
solid surface triggers the code to assume the material to be thermally-thick, mandating that other properties
be specified as well. Simply prescribing as many properties as possible from a handbook is bad practice.
Only prescribe those parameters which are necessary to describe the desired scenario.
15.1 BNDF (Boundary File Parameters)

Table 15.1: For more information see Section 14.2.4.

<table>
<thead>
<tr>
<th>BNDF (Boundary File Parameters)</th>
<th>Logical</th>
<th>Do not do corner averaging</th>
<th>.FALSE.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CELL_CENTERED</td>
<td>Logical</td>
<td>Do not do corner averaging</td>
<td>.FALSE.</td>
</tr>
<tr>
<td>FYI</td>
<td>Character</td>
<td>Comment String (has no effect)</td>
<td></td>
</tr>
<tr>
<td>PART_ID</td>
<td>Character</td>
<td>Particle identifier (if needed)</td>
<td></td>
</tr>
<tr>
<td>PROP_ID</td>
<td>Character</td>
<td>Property identifier (if needed)</td>
<td></td>
</tr>
<tr>
<td>RECOUNT_DRIP</td>
<td>Logical</td>
<td>Adds mass to AWMPUA for each drop impact</td>
<td>.FALSE.</td>
</tr>
<tr>
<td>QUANTITY</td>
<td>Character</td>
<td>Quantity to visualize</td>
<td></td>
</tr>
<tr>
<td>SPEC_ID</td>
<td>Character</td>
<td>Species identifier (if needed)</td>
<td></td>
</tr>
</tbody>
</table>

15.2 CLIP (MIN/MAX Clipping Parameters)

Table 15.2: For more information see Section 6.7.

<table>
<thead>
<tr>
<th>CLIP (Specified Upper and Lower Limits)</th>
<th>Character</th>
<th>Comment String (has no effect)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FYI</td>
<td>Character</td>
<td>Comment String (has no effect)</td>
<td></td>
</tr>
<tr>
<td>MAXIMUM DENSITY</td>
<td>Real</td>
<td>Maximum Gas Density</td>
<td>kg/m³</td>
</tr>
<tr>
<td>MAXIMUM MASS FRACTION</td>
<td>Real Array</td>
<td>Maximum Gas Mass Fraction</td>
<td>kg/kg</td>
</tr>
<tr>
<td>MAXIMUM TEMPERATURE</td>
<td>Real</td>
<td>Maximum Gas Temperature</td>
<td>°C</td>
</tr>
<tr>
<td>MINIMUM DENSITY</td>
<td>Real</td>
<td>Minimum Gas Density</td>
<td>kg/m³</td>
</tr>
<tr>
<td>MINIMUM MASS FRACTION</td>
<td>Real Array</td>
<td>Minimum Gas Mass Fraction</td>
<td>kg/kg</td>
</tr>
<tr>
<td>MINIMUM TEMPERATURE</td>
<td>Real</td>
<td>Maximum Gas Temperature</td>
<td>°C</td>
</tr>
</tbody>
</table>

15.3 CTRL (Control Function Parameters)

Table 15.3: For more information see Section 13.5.

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

Table 15.4: For more information see Section 13.1.

<table>
<thead>
<tr>
<th>DEVC (Device Parameters)</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BYPASS_FLOWRATE</td>
<td>Real</td>
<td>Aspiration smoke detector parameter</td>
<td>kg/s</td>
<td>0</td>
</tr>
<tr>
<td>CTRL_ID</td>
<td>Character</td>
<td>Associated CTRL line</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELAY</td>
<td>Real</td>
<td>Transport time for an aspiration detector</td>
<td>s</td>
<td>0</td>
</tr>
<tr>
<td>DEVC_ID</td>
<td>Character</td>
<td>Associated DEVC line for aspiration detector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DUCT_ID</td>
<td>Character</td>
<td>Associated HVAC line for a duct output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEPTH</td>
<td>Real</td>
<td>Depth into surface for internal wall temp</td>
<td>m</td>
<td>0</td>
</tr>
<tr>
<td>FLOWRATE</td>
<td>Real</td>
<td>Suction flowrate for an aspiration detector</td>
<td>kg/s</td>
<td>0</td>
</tr>
<tr>
<td>FYI</td>
<td>Character</td>
<td>Comment String (has no effect)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOR</td>
<td>Integer</td>
<td>Index of Orientation $(±1,±2,±3)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Character</td>
<td>Identifying label for output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INITIAL_STATE</td>
<td>Logical</td>
<td>Initial state of device</td>
<td>.FALSE.</td>
<td></td>
</tr>
<tr>
<td>LATCH</td>
<td>Logical</td>
<td>Device cannot change state multiple times</td>
<td>.TRUE.</td>
<td></td>
</tr>
<tr>
<td>MATL_ID</td>
<td>Character</td>
<td>Material identifier (if needed)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NODE_ID</td>
<td>Character</td>
<td>Associated HVAC line for a node output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORIENTATION</td>
<td>Real Triplet</td>
<td>Direction vector</td>
<td>0,0,-1</td>
<td></td>
</tr>
<tr>
<td>PART_ID</td>
<td>Character</td>
<td>Particle identifier (if needed)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POINTS</td>
<td>Integer</td>
<td>See Section 14.2.1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>PROP_ID</td>
<td>Character</td>
<td>Associated PROPERTY line</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUANTITY</td>
<td>Character</td>
<td>Name of Quantity to output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROTATION</td>
<td>Real Triplet</td>
<td>Rotation Angle</td>
<td>deg</td>
<td>0</td>
</tr>
<tr>
<td>SETPOINT</td>
<td>Real</td>
<td>Value at which device changes state</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPEC_ID</td>
<td>Character</td>
<td>Species identifier (if needed)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATISTICS</td>
<td>Character</td>
<td>See Section 14.3.10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SURF_ID</td>
<td>Character</td>
<td>See Section 14.3.10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME_AVERAGED</td>
<td>Logical</td>
<td>See Section 13.2</td>
<td>.TRUE.</td>
<td></td>
</tr>
<tr>
<td>TRIP_DIRECTION</td>
<td>Integer</td>
<td>Sign of derivative at first state change</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>VELO_INDEX</td>
<td>Integer</td>
<td>See Section 14.3.14</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>XB(6)</td>
<td>Real Sextuplet</td>
<td>Coordinates of non-point measurement</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>XYZ</td>
<td>Real Triplet</td>
<td>Physical coordinates</td>
<td>m</td>
<td></td>
</tr>
</tbody>
</table>

### 15.5 DUMP (Output Parameters)

Table 15.5: For more information see Section 14.1.

<table>
<thead>
<tr>
<th>DUMP (Output Parameters)</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>COLUMN_DUMP_LIMIT</td>
<td>Logical</td>
<td>Limit text output to 255 columns</td>
<td>.TRUE.</td>
<td></td>
</tr>
<tr>
<td>CTRL_COLUMN_LIMIT</td>
<td>Integer</td>
<td>Number of columns in CTRL file</td>
<td>254</td>
<td></td>
</tr>
<tr>
<td>DEVC_COLUMN_LIMIT</td>
<td>Integer</td>
<td>Number of columns in DEVC file</td>
<td>254</td>
<td></td>
</tr>
<tr>
<td>DT_BNDF</td>
<td>Real</td>
<td>Boundary dump interval</td>
<td>s</td>
<td>$2\Delta t/NFRAMES$</td>
</tr>
</tbody>
</table>
Table 15.5: Continued

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Units</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT_CTRL</td>
<td>Real</td>
<td>Control status dump interval</td>
<td>s</td>
<td>$\Delta t$/NFRAMES</td>
</tr>
<tr>
<td>DT_DEVCF</td>
<td>Real</td>
<td>Device output dump interval</td>
<td>s</td>
<td>$\Delta t$/NFRAMES</td>
</tr>
<tr>
<td>DT_DEVCF_LINE</td>
<td>Real</td>
<td>See Section 14.2.1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>DT_FLUSH</td>
<td>Real</td>
<td>See Section 14.1</td>
<td>s</td>
<td>$\Delta t$/NFRAMES</td>
</tr>
<tr>
<td>DT_HRR</td>
<td>Real</td>
<td>Heat release dump interval</td>
<td>s</td>
<td>$\Delta t$/NFRAMES</td>
</tr>
<tr>
<td>DT_ISO</td>
<td>Real</td>
<td>Iso-surface dump interval</td>
<td>s</td>
<td>$\Delta t$/NFRAMES</td>
</tr>
<tr>
<td>DT_MASS</td>
<td>Real</td>
<td>Mass diagnostic dump interval</td>
<td>s</td>
<td>$\Delta t$/NFRAMES</td>
</tr>
<tr>
<td>DT_PART</td>
<td>Real</td>
<td>Particle dump interval</td>
<td>s</td>
<td>$\Delta t$/NFRAMES</td>
</tr>
<tr>
<td>DT_PLOT3D</td>
<td>Real</td>
<td>PLOT3D dump interval</td>
<td>s</td>
<td>$\Delta t$/5</td>
</tr>
<tr>
<td>DT_PROF</td>
<td>Real</td>
<td>Profile dump interval</td>
<td>s</td>
<td>$\Delta t$/NFRAMES</td>
</tr>
<tr>
<td>DT_RESTART</td>
<td>Real</td>
<td>Restart core dump interval</td>
<td>s</td>
<td>1000000.</td>
</tr>
<tr>
<td>DT_SLCE</td>
<td>Real</td>
<td>Slice dump interval</td>
<td>s</td>
<td>$\Delta t$/NFRAMES</td>
</tr>
<tr>
<td>FLUSH_FILE_BUFFERS</td>
<td>Logical</td>
<td>See Section 14.1</td>
<td>.TRUE.</td>
<td></td>
</tr>
<tr>
<td>MASS_FILE</td>
<td>Logical</td>
<td>Flag for species MASS file</td>
<td>.FALSE.</td>
<td></td>
</tr>
<tr>
<td>MAXIMUM_DROPLETS</td>
<td>Integer</td>
<td>Max particles per mesh</td>
<td>500000</td>
<td></td>
</tr>
<tr>
<td>NFRAMES</td>
<td>Integer</td>
<td>Number of Frames of output data</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>PLOT3D_QUANTITY(5)</td>
<td>Char. Quint</td>
<td>See Section 14.2.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLOT3D_SPEC_ID(5)</td>
<td>Char. Quint</td>
<td>See Section 14.2.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLOT3D_VELO_INDEX</td>
<td>Integer Quint</td>
<td>See Section 14.3.14</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SMOKE3D</td>
<td>Logical</td>
<td>Flag for 3D Smoke Visualization</td>
<td>.TRUE.</td>
<td></td>
</tr>
<tr>
<td>SMOKE3D_QUANTITY</td>
<td>Character</td>
<td>See Section 14.2.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMOKE3D_SPEC_ID</td>
<td>Character</td>
<td>See Section 14.2.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATE_FILE</td>
<td>Logical</td>
<td>Flag for state relation file</td>
<td>.FALSE.</td>
<td></td>
</tr>
<tr>
<td>STATUS_FILES</td>
<td>Logical</td>
<td>Flag for status (notready) file</td>
<td>.FALSE.</td>
<td></td>
</tr>
<tr>
<td>WRITE_XYZ</td>
<td>Logical</td>
<td>Flag for writing PLOT3D .xyz file</td>
<td>.FALSE.</td>
<td></td>
</tr>
</tbody>
</table>

$\Delta t = T_{END} - T_{BEGIN}$

### 15.6 HEAD (Header Parameters)

Table 15.6: For more information see Section 6.1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHID</td>
<td>Character</td>
<td>Job Identification String</td>
<td>'output'</td>
</tr>
<tr>
<td>FYI</td>
<td>Character</td>
<td>Comment String (has no effect)</td>
<td></td>
</tr>
<tr>
<td>TITLE</td>
<td>Character</td>
<td>Title for job</td>
<td></td>
</tr>
</tbody>
</table>

### 15.7 HOLE (Obstruction Cutout Parameters)
Table 15.7: For more information see Section 7.3.

<table>
<thead>
<tr>
<th>HOLE (Obstruction Cutout Parameters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR</td>
</tr>
<tr>
<td>CTRL_ID</td>
</tr>
<tr>
<td>DEVC_ID</td>
</tr>
<tr>
<td>FYI</td>
</tr>
<tr>
<td>RGB(3)</td>
</tr>
<tr>
<td>TRANSPARENCY</td>
</tr>
<tr>
<td>XB(6)</td>
</tr>
</tbody>
</table>

### 15.8 HVAC (HVAC System Definition)

Table 15.8: For more information see Section 9.5.

<table>
<thead>
<tr>
<th>HVAC (HVAC System Definition)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMBIENT</td>
</tr>
<tr>
<td>AREA</td>
</tr>
<tr>
<td>CTRL_ID</td>
</tr>
<tr>
<td>DAMPER</td>
</tr>
<tr>
<td>DEVC_ID</td>
</tr>
<tr>
<td>DUCT_ID</td>
</tr>
<tr>
<td>DIAMETER</td>
</tr>
<tr>
<td>FAN_ID</td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>LENGTH</td>
</tr>
<tr>
<td>LOSS</td>
</tr>
<tr>
<td>MAX FLOW</td>
</tr>
<tr>
<td>MAX PRESSURE</td>
</tr>
<tr>
<td>NODE_ID</td>
</tr>
<tr>
<td>RAMP_ID</td>
</tr>
<tr>
<td>ROUGHNESS</td>
</tr>
<tr>
<td>TYPE_ID</td>
</tr>
<tr>
<td>VENT_ID</td>
</tr>
<tr>
<td>VOLUME FLOW</td>
</tr>
<tr>
<td>XYZ</td>
</tr>
</tbody>
</table>

### 15.9 INIT (Initial Conditions)

Table 15.9: For more information see Section 6.5.

<table>
<thead>
<tr>
<th>INIT (Initial Conditions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DENSITY</td>
</tr>
</tbody>
</table>
Table 15.9: Continued

<table>
<thead>
<tr>
<th>INIT (Initial Conditions)</th>
<th></th>
</tr>
</thead>
</table>
| MASS_FRACTION(II)        | Real Array | Initial value of species II | kg/kg | Ambient
| MASS_PER_TIME            | Real      | See Section 12.2.3          | kg/s  |
| MASS_PER_VOLUME          | Real      | See Section 12.2.3          | kg/m³ | 1
| MULT_ID                  | Character | See Section 7.2.2            |
| NUMBER_INITIAL_DROPLETS  | Integer   | See Section 12.2.3          |       | 0
| PART_ID                  | Character | See Section 12.2.3          |
| TEMPERATURE              | Real      | Initial value of temperature | °C   | TMPA
| XB(6)                    | Real Sextuplet | Coordinates | m  |

15.10 ISOF (Isosurface Parameters)

Table 15.10: For more information see Section 14.2.5.

<table>
<thead>
<tr>
<th>ISOF (Isosurface Parameters)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FYI</td>
<td>Character</td>
</tr>
<tr>
<td>QUANTITY</td>
<td>Character</td>
</tr>
<tr>
<td>SPEC_ID</td>
<td>Character</td>
</tr>
<tr>
<td>VALUE(I)</td>
<td>Real Array</td>
</tr>
<tr>
<td>VELO_INDEX</td>
<td>Integer</td>
</tr>
</tbody>
</table>

15.11 MATL (Material Properties)

Table 15.11: For more information see Section 8.3.

<table>
<thead>
<tr>
<th>MATL (Material Properties)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Real array</td>
</tr>
<tr>
<td>ABSORPTION_COEFFICIENT</td>
<td>Real</td>
</tr>
<tr>
<td>BOILING_TEMPERATURE</td>
<td>Real</td>
</tr>
<tr>
<td>CONDUCTIVITY</td>
<td>Real</td>
</tr>
<tr>
<td>CONDUCTIVITY_RAMP</td>
<td>Character</td>
</tr>
<tr>
<td>DENSITY</td>
<td>Real</td>
</tr>
<tr>
<td>E</td>
<td>Real array</td>
</tr>
<tr>
<td>EMISSIVITY</td>
<td>Real</td>
</tr>
<tr>
<td>FYI</td>
<td>Character</td>
</tr>
<tr>
<td>HEATING_RATE</td>
<td>Real array</td>
</tr>
<tr>
<td>HEAT_OF_COMBUSTION(:)</td>
<td>Real array</td>
</tr>
<tr>
<td>HEAT_OF_REACTION(:)</td>
<td>Real array</td>
</tr>
<tr>
<td>INITIAL_VAPOR_FLUX</td>
<td>Real</td>
</tr>
<tr>
<td>ID</td>
<td>Character</td>
</tr>
<tr>
<td>THRESHOLD_TEMPERATURE</td>
<td>Real array</td>
</tr>
</tbody>
</table>
### Table 15.11: Continued

<table>
<thead>
<tr>
<th>MATL (Material Properties)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N_REACTIONS</td>
</tr>
<tr>
<td>N_S</td>
</tr>
<tr>
<td>N_T</td>
</tr>
<tr>
<td>NU_FUEL</td>
</tr>
<tr>
<td>NU_GAS(:,II)</td>
</tr>
<tr>
<td>NU_RESIDUE</td>
</tr>
<tr>
<td>NU_WATER</td>
</tr>
<tr>
<td>PYROLYSIS_RANGE</td>
</tr>
<tr>
<td>REFERENCE_TEMPERATURE</td>
</tr>
<tr>
<td>RESIDUE</td>
</tr>
<tr>
<td>SPECIFIC_HEAT</td>
</tr>
<tr>
<td>SPECIFIC_HEAT_RAMP</td>
</tr>
</tbody>
</table>

### 15.12 MESH (Mesh Parameters)

Table 15.12: For more information see Section 6.3.

<table>
<thead>
<tr>
<th>MESH (Mesh Parameters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR</td>
</tr>
<tr>
<td>CYLINDRICAL</td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>IJK</td>
</tr>
<tr>
<td>FYI</td>
</tr>
<tr>
<td>MPI_PROCESS</td>
</tr>
<tr>
<td>RGB</td>
</tr>
<tr>
<td>MULT_ID</td>
</tr>
<tr>
<td>SYNCHRONIZE</td>
</tr>
<tr>
<td>XB(6)</td>
</tr>
</tbody>
</table>

### 15.13 MISC (Miscellaneous Parameters)

Table 15.13: For more information see Section 6.4.

<table>
<thead>
<tr>
<th>MISC (Miscellaneous Parameters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOW_UNDERSIDE_DROPLETS</td>
</tr>
<tr>
<td>ASSUMED_GAS_TEMPERATURE</td>
</tr>
<tr>
<td>BACKGROUND_SPECIES</td>
</tr>
<tr>
<td>BAROCLINIC</td>
</tr>
<tr>
<td>BNDF_DEFAULT</td>
</tr>
<tr>
<td>CFL_MAX</td>
</tr>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>----------------------</td>
</tr>
<tr>
<td>CFL_MIN</td>
</tr>
<tr>
<td>C_FORCED</td>
</tr>
<tr>
<td>C_HORIZONTAL</td>
</tr>
<tr>
<td>C_VERTICAL</td>
</tr>
<tr>
<td>CSMAG</td>
</tr>
<tr>
<td>CONDUCTIVITY</td>
</tr>
<tr>
<td>CO_PRODUCTION</td>
</tr>
<tr>
<td>DNS</td>
</tr>
<tr>
<td>FYI</td>
</tr>
<tr>
<td>GROUND_LEVEL</td>
</tr>
<tr>
<td>GVEC</td>
</tr>
<tr>
<td>h_EDDY</td>
</tr>
<tr>
<td>HUMIDITY</td>
</tr>
<tr>
<td>ISOThermal</td>
</tr>
<tr>
<td>LAPSE_RATE</td>
</tr>
<tr>
<td>LES</td>
</tr>
<tr>
<td>MW</td>
</tr>
<tr>
<td>NOISE</td>
</tr>
<tr>
<td>OVERWRITE</td>
</tr>
<tr>
<td>PR</td>
</tr>
<tr>
<td>P_INF</td>
</tr>
<tr>
<td>POROUS_FLOOR</td>
</tr>
<tr>
<td>RADIATION</td>
</tr>
<tr>
<td>RAMP_GX</td>
</tr>
<tr>
<td>RAMP_GY</td>
</tr>
<tr>
<td>RAMP_GZ</td>
</tr>
<tr>
<td>RESTART</td>
</tr>
<tr>
<td>RESTART_CHID</td>
</tr>
<tr>
<td>SC</td>
</tr>
<tr>
<td>SOLID_PHASE_ONLY</td>
</tr>
<tr>
<td>STRATIFICATION</td>
</tr>
<tr>
<td>SUPPRESSION</td>
</tr>
<tr>
<td>SURF_DEFAULT</td>
</tr>
<tr>
<td>TEXTURE_ORIGIN</td>
</tr>
<tr>
<td>THICKEN_OBSTENSIONS</td>
</tr>
<tr>
<td>TMPA</td>
</tr>
<tr>
<td>U0, V0, W0</td>
</tr>
<tr>
<td>VISCOSITY</td>
</tr>
<tr>
<td>VN_MAX</td>
</tr>
<tr>
<td>VN_MIN</td>
</tr>
</tbody>
</table>
15.14 **MULT** (Multiplier Function Parameters)

Table 15.14: For more information see Section 7.2.2.

| MULT (Multiplier Function Parameters) |
|-----------------------------|---------------------------------|------|
| DXB | Real Sextuplet | Spacing for all 6 coordinates | m 0. |
| DX | Real | Spacing in the x direction | m 0. |
| DY | Real | Spacing in the y direction | m 0. |
| DZ | Real | Spacing in the z direction | m 0. |
| DX0 | Real | Translation in the x direction | m 0. |
| DY0 | Real | Translation in the y direction | m 0. |
| DZ0 | Real | Translation in the z direction | m 0. |
| ID | Character | IDENTifier |
| I_LOWER | Integer | Lower array bound, x direction | 0 |
| I_UPPER | Integer | Upper array bound, x direction | 0 |
| J_LOWER | Integer | Lower array bound, y direction | 0 |
| J_UPPER | Integer | Upper array bound, y direction | 0 |
| K_LOWER | Integer | Lower array bound, z direction | 0 |
| K_UPPER | Integer | Upper array bound, z direction | 0 |
| N_LOWER | Integer | Lower sequence bound | 0 |
| N_UPPER | Integer | Upper sequence bound | 0 |

15.15 **OBST** (Obstruction Parameters)

Table 15.15: For more information see Section 7.2.

<table>
<thead>
<tr>
<th>OBST (Obstruction Parameters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOW_VENT</td>
</tr>
<tr>
<td>BNDF_FACE(-3:3)</td>
</tr>
<tr>
<td>BNDF_OBST</td>
</tr>
<tr>
<td>BULK_DENSITY</td>
</tr>
<tr>
<td>COLOR</td>
</tr>
<tr>
<td>CTRL_ID</td>
</tr>
<tr>
<td>DEVC_ID</td>
</tr>
<tr>
<td>FYI</td>
</tr>
<tr>
<td>MULT_ID</td>
</tr>
<tr>
<td>OUTLINE</td>
</tr>
<tr>
<td>PERMIT_HOLE</td>
</tr>
<tr>
<td>REMOVABLE</td>
</tr>
<tr>
<td>RGB(3)</td>
</tr>
<tr>
<td>SAWTOOTH</td>
</tr>
<tr>
<td>SURF_ID</td>
</tr>
<tr>
<td>SURF_IDS(3)</td>
</tr>
<tr>
<td>SURF_ID6(6)</td>
</tr>
</tbody>
</table>
Table 15.15: Continued

<table>
<thead>
<tr>
<th>OBST (Obstruction Parameters)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>THICKEN</td>
<td>Logical</td>
<td>Force at least one cell thick</td>
<td>.FALSE.</td>
</tr>
<tr>
<td>TEXTURE_ORIGIN(3)</td>
<td>Real Triplet</td>
<td>See Section 7.5.1</td>
<td>m (0,0,0,)</td>
</tr>
<tr>
<td>TRANSPARENCY</td>
<td>Real</td>
<td>Transparency indicator</td>
<td>1</td>
</tr>
<tr>
<td>XB(6)</td>
<td>Real Sextuplet</td>
<td>Min/Max Physical coordinates</td>
<td>m</td>
</tr>
</tbody>
</table>

15.16 PART (Lagrangian Particles/Droplets)

Table 15.16: For more information see Section 12.

<table>
<thead>
<tr>
<th>PART (Lagrangian Particles/Droplets)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AGE</td>
<td>Real</td>
<td>Droplet lifetime</td>
<td>s</td>
</tr>
<tr>
<td>COLOR</td>
<td>Character</td>
<td>Default color of droplets</td>
<td>’BLACK’</td>
</tr>
<tr>
<td>CTRL_ID</td>
<td>Character</td>
<td>Name of controller</td>
<td></td>
</tr>
<tr>
<td>DENSITY</td>
<td>Real</td>
<td>Droplet density</td>
<td>kg/m³</td>
</tr>
<tr>
<td>DEVC_ID</td>
<td>Character</td>
<td>Name of controlling device</td>
<td></td>
</tr>
<tr>
<td>DIAMETER</td>
<td>Real</td>
<td>Median Volumetric Diameter</td>
<td>µm</td>
</tr>
<tr>
<td>DRAG_LAW</td>
<td>Character</td>
<td>Geometry for drag correlation</td>
<td>’SPHERE’</td>
</tr>
<tr>
<td>EVAPORATE</td>
<td>Logical</td>
<td>Assume liquid evaporation</td>
<td>.TRUE.</td>
</tr>
<tr>
<td>FYI</td>
<td>Character</td>
<td>Comment String</td>
<td></td>
</tr>
<tr>
<td>FUEL</td>
<td>Logical</td>
<td>Liquid Fuel</td>
<td></td>
</tr>
<tr>
<td>GAMMA_D</td>
<td>Real</td>
<td>Size distribution parameter</td>
<td>2.4</td>
</tr>
<tr>
<td>HEAT_OF_COMBUSTION</td>
<td>Real</td>
<td>Heat of Combustion</td>
<td>kJ/kg</td>
</tr>
<tr>
<td>HEAT_OF_VAPORIZATION</td>
<td>Real</td>
<td>Latent Heat of Vaporization</td>
<td>kJ/kg</td>
</tr>
<tr>
<td>H_V_REFERENCE_TEMPERATURE</td>
<td>Real</td>
<td>See Section 12.3.1</td>
<td>°C</td>
</tr>
<tr>
<td>HORIZONTAL_VELOCITY</td>
<td>Real</td>
<td>Droplet speed, horizontal</td>
<td>m/s</td>
</tr>
<tr>
<td>ID</td>
<td>Character</td>
<td>Identifier</td>
<td></td>
</tr>
<tr>
<td>INITIAL_TEMPERATURE</td>
<td>Real</td>
<td>Initial Temperature</td>
<td>°C</td>
</tr>
<tr>
<td>MASSLESS</td>
<td>Logical</td>
<td>Massless tracers</td>
<td>.FALSE.</td>
</tr>
<tr>
<td>MAXIMUM_DIAMETER</td>
<td>Real</td>
<td>Break-up bound</td>
<td>µm</td>
</tr>
<tr>
<td>MINIMUM_DIAMETER</td>
<td>Real</td>
<td>Evaporation bound</td>
<td>µm</td>
</tr>
<tr>
<td>MELTING_TEMPERATURE</td>
<td>Real</td>
<td>Melting Temperature</td>
<td>°C</td>
</tr>
<tr>
<td>MONODISPERSE</td>
<td>Logical</td>
<td>Uniform droplet size</td>
<td>.FALSE.</td>
</tr>
<tr>
<td>PROP_ID</td>
<td>Character</td>
<td>Name of Property line</td>
<td></td>
</tr>
<tr>
<td>QUANTITIES(10)</td>
<td>Character</td>
<td>Quantities for coloring</td>
<td></td>
</tr>
<tr>
<td>RGB(3)</td>
<td>Integers</td>
<td>Color indices (0-255)</td>
<td></td>
</tr>
<tr>
<td>SAMPLING_FACTOR</td>
<td>Integer</td>
<td>Filter for output file</td>
<td>1</td>
</tr>
<tr>
<td>SIGMA_D</td>
<td>Real</td>
<td>Size distribution parameter</td>
<td></td>
</tr>
<tr>
<td>SPEC_ID</td>
<td>Character</td>
<td>Name of gas species</td>
<td></td>
</tr>
<tr>
<td>SPECIFIC_HEAT</td>
<td>Real</td>
<td>Droplet specific heat</td>
<td>kJ/kg/K</td>
</tr>
<tr>
<td>STATIC</td>
<td>Logical</td>
<td>Stationary Particles</td>
<td>.FALSE.</td>
</tr>
<tr>
<td>USER_DRAG_COEFFICIENT</td>
<td>Real</td>
<td>Constant drag coefficient</td>
<td>-1.</td>
</tr>
<tr>
<td>VERTICAL_VELOCITY</td>
<td>Real</td>
<td>Droplet speed, vertical</td>
<td>m/s</td>
</tr>
</tbody>
</table>
Table 15.16: Continued

<table>
<thead>
<tr>
<th>PART (Lagrangian Particles/Droplets)</th>
<th>WATER</th>
<th>Logical</th>
<th>Water Droplet</th>
<th>.FALSE.</th>
</tr>
</thead>
</table>

15.17 PRES (Pressure Solver Parameters)

Table 15.17: For more information see Section 6.6.

<table>
<thead>
<tr>
<th>PRES (Pressure Solver Parameters)</th>
<th>MAX_PRESSURE_ITERATIONS</th>
<th>Integer</th>
<th>See Section 6.6</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>VELOCITY_TOLERANCE</td>
<td></td>
<td>Real</td>
<td>See Section 6.6</td>
<td>m/s</td>
</tr>
</tbody>
</table>

15.18 PROF (Wall Profile Parameters)

Table 15.18: For more information see Section 14.2.2.

<table>
<thead>
<tr>
<th>PROF (Wall Profile Parameters)</th>
<th>IOR</th>
<th>Real</th>
<th>Orientation of wall surface</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Character</td>
<td>Identifier</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FYI</td>
<td>Character</td>
<td>Comment String (has no effect)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUANTITY</td>
<td>Character</td>
<td>Name of output quantity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XYZ</td>
<td>Real Triplet</td>
<td>Coordinates of wall surface</td>
<td>m</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

15.19 PROP (Device Properties)

Table 15.19: For more information see Section 13.3.

<table>
<thead>
<tr>
<th>PROP (Device Properties)</th>
<th>ACTIVATION_TEMPERATURE</th>
<th>Real</th>
<th>Threshold link temperature</th>
<th>°C</th>
<th>74</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVATION_OBSCURATION</td>
<td>Real</td>
<td>Threshold value of obscuration</td>
<td>%/m</td>
<td>3.28</td>
<td></td>
</tr>
<tr>
<td>ALPHA_C</td>
<td>Real</td>
<td>Smoke detector parameter</td>
<td>1.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALPHA_E</td>
<td>Real</td>
<td>Smoke detector parameter</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BETA_C</td>
<td>Real</td>
<td>Smoke detector parameter</td>
<td>1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BETA_E</td>
<td>Real</td>
<td>Smoke detector parameter</td>
<td>1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BEAD_DENSITY</td>
<td>Real</td>
<td>Density of TC bead</td>
<td>kg/m³</td>
<td>8908.</td>
<td></td>
</tr>
<tr>
<td>BEAD_DIAMETER</td>
<td>Real</td>
<td>Diameter of TC bead</td>
<td>m</td>
<td>0.001</td>
<td></td>
</tr>
<tr>
<td>BEAD_EMISSIVITY</td>
<td>Real</td>
<td>Emissivity of TC bead</td>
<td></td>
<td>0.85</td>
<td></td>
</tr>
<tr>
<td>BEAD_SPECIFIC_HEAT</td>
<td>Real</td>
<td>Specific Heat of TC bead</td>
<td>kJ/kg/K</td>
<td>0.44</td>
<td></td>
</tr>
<tr>
<td>CABLE_DIAMETER</td>
<td>Real</td>
<td>See Section 13.3.7</td>
<td>m</td>
<td>0.02</td>
<td></td>
</tr>
<tr>
<td>CABLE_FAILURE_TEMPERATURE</td>
<td>Real</td>
<td>See Section 13.3.7</td>
<td>°C</td>
<td>400</td>
<td></td>
</tr>
</tbody>
</table>
### Table 15.19: Continued

<table>
<thead>
<tr>
<th>PROP (Device Properties)</th>
<th>Type</th>
<th>Description</th>
<th>Unit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CABLE_JACKET_THICKNESS</td>
<td>Real</td>
<td>See Section 13.3.7</td>
<td>m</td>
<td>0.002</td>
</tr>
<tr>
<td>CABLE_MASS_PER_LENGTH</td>
<td>Real</td>
<td>See Section 13.3.7</td>
<td>kg/m</td>
<td>0.5</td>
</tr>
<tr>
<td>C_FACTOR</td>
<td>Real</td>
<td>Sprinkler activation parameter</td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>CHARACTERISTIC_VELOCITY</td>
<td>Real</td>
<td>See Section 14.3.12</td>
<td>m/s</td>
<td>1.0</td>
</tr>
<tr>
<td>DROPLET_VELOCITY</td>
<td>Real</td>
<td>Initial droplet velocity</td>
<td>m/s</td>
<td>0.0</td>
</tr>
<tr>
<td>DROPLETS_PER_SECOND</td>
<td>Integer</td>
<td>Drops per second per head</td>
<td></td>
<td>5000</td>
</tr>
<tr>
<td>DT_INSERT</td>
<td>Real</td>
<td>Time between particle insertions</td>
<td>s</td>
<td>0.01</td>
</tr>
<tr>
<td>FLOW_RATE</td>
<td>Real</td>
<td>Sprinkler or nozzle flow rate</td>
<td>L/min</td>
<td></td>
</tr>
<tr>
<td>FLOW_RAMP</td>
<td>Character</td>
<td>Time RAMP for flow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOW_TAU</td>
<td>Real</td>
<td>Time constant for flow</td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>GAUGE_TEMPERATURE</td>
<td>Real</td>
<td>See Section 14.3.5</td>
<td>°C</td>
<td>TMPA</td>
</tr>
<tr>
<td>ID</td>
<td>Character</td>
<td>IDentifier</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INITIAL_TEMPERATURE</td>
<td>Real</td>
<td>Initial link temperature</td>
<td>°C</td>
<td>TMPA</td>
</tr>
<tr>
<td>K_FACTOR</td>
<td>Real</td>
<td>Flow parameter</td>
<td>L/min/atm$^{1/2}$</td>
<td>1.0</td>
</tr>
<tr>
<td>LENGTH</td>
<td>Real</td>
<td>Smoke detector parameter</td>
<td></td>
<td>1.8</td>
</tr>
<tr>
<td>OFFSET</td>
<td>Real</td>
<td>Droplet offset distance</td>
<td>m</td>
<td>0.05</td>
</tr>
<tr>
<td>OPERATING_PRESSURE</td>
<td>Real</td>
<td>Sprinkler pipe pressure</td>
<td>atm</td>
<td>1.0</td>
</tr>
<tr>
<td>ORIFICE_DIAMETER</td>
<td>Real</td>
<td>Nozzle orifice diameter</td>
<td>m</td>
<td>0.0</td>
</tr>
<tr>
<td>PART_ID</td>
<td>Character</td>
<td>Name of associated PART line</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDPA_END</td>
<td>Real</td>
<td>See Section 14.3.6</td>
<td>s</td>
<td>T_END</td>
</tr>
<tr>
<td>PDPA_M</td>
<td>Integer</td>
<td>See Section 14.3.6</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>PDPA_M</td>
<td>Integer</td>
<td>See Section 14.3.6</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>PDPA_RADIUS</td>
<td>Real</td>
<td>See Section 14.3.6</td>
<td>m</td>
<td>0.0</td>
</tr>
<tr>
<td>PDPA_START</td>
<td>Real</td>
<td>See Section 14.3.6</td>
<td>s</td>
<td>0.0</td>
</tr>
<tr>
<td>PRESSURE_RAMP</td>
<td>Character</td>
<td>Pressure RAMP for sprinklers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUANTITY</td>
<td>Character</td>
<td>Name of associated output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTI</td>
<td>Real</td>
<td>Response Time Index</td>
<td>√m/s</td>
<td>100.0</td>
</tr>
<tr>
<td>SMOKEVIEW_ID</td>
<td>Char. Array</td>
<td>Name(s) of drawn object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMOKEVIEW_PARAMETERS</td>
<td>Char. Array</td>
<td>Misc. parameters for drawing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPEC_ID</td>
<td>Character</td>
<td>See Section 13.3.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPRAY_ANGLE(2)</td>
<td>Real</td>
<td>Cone angles for water spray</td>
<td>deg</td>
<td>60.0,75.0</td>
</tr>
<tr>
<td>SPRAY_PATTERN_TABLE</td>
<td>Character</td>
<td>TABL for spray pattern</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 15.20 RADI (Radiation Parameters)

Table 15.20: For more information see Section 11.3.

<table>
<thead>
<tr>
<th>RADI (Radiation Parameters)</th>
<th>Type</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANGLE_INCREMENT</td>
<td>Integer</td>
<td>Number of angles skipped per update</td>
<td>5</td>
</tr>
<tr>
<td>CH4_BANDS</td>
<td>Logical</td>
<td>Include extra fuel bands</td>
<td>.FALSE.</td>
</tr>
<tr>
<td>KAPPA0</td>
<td>Real</td>
<td>Constant absorption coefficient</td>
<td>1/m</td>
</tr>
<tr>
<td>NMIEANG</td>
<td>Integer</td>
<td>Number of polar angles</td>
<td>15</td>
</tr>
</tbody>
</table>

196
Table 15.20: Continued

<table>
<thead>
<tr>
<th>RADI (Radiation Parameters)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NUMBER_RADIATIONANGLES</strong></td>
<td>Integer</td>
<td>Number of solid angles</td>
<td>104</td>
</tr>
<tr>
<td><strong>PATH_LENGTH</strong></td>
<td>Real</td>
<td>Path length for radiation calc.</td>
<td>m</td>
</tr>
<tr>
<td><strong>RADIATIVE_FRACTION</strong></td>
<td>Real</td>
<td>Radiative Loss Fraction</td>
<td>0.35</td>
</tr>
<tr>
<td><strong>RADTMP</strong></td>
<td>Real</td>
<td>Assumed radiative source temp.</td>
<td>°C</td>
</tr>
<tr>
<td><strong>TIME_STEP_INCREMENT</strong></td>
<td>Integer</td>
<td>Number time steps skipped</td>
<td>3</td>
</tr>
<tr>
<td><strong>WIDE_BAND_MODEL</strong></td>
<td>Logical</td>
<td>Non-gray gas assumption</td>
<td>.FALSE.</td>
</tr>
</tbody>
</table>

15.21 RAMP (Ramp Function Parameters)

Table 15.21: For more information see Section 10.

<table>
<thead>
<tr>
<th>RAMP (Ramp Function Parameters)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DEVCI_ID</strong></td>
<td>Character</td>
<td>See Section 13.6</td>
<td></td>
</tr>
<tr>
<td><strong>F</strong></td>
<td>Real</td>
<td>Function value</td>
<td></td>
</tr>
<tr>
<td><strong>FYI</strong></td>
<td>Character</td>
<td>Comment String (has no effect)</td>
<td></td>
</tr>
<tr>
<td><strong>ID</strong></td>
<td>Character</td>
<td>Identifier</td>
<td></td>
</tr>
<tr>
<td><strong>T</strong></td>
<td>Real</td>
<td>Time (or Temperature)</td>
<td>s (or °C)</td>
</tr>
<tr>
<td><strong>X</strong></td>
<td>Real</td>
<td>x-coordinate (gravity only)</td>
<td>m</td>
</tr>
</tbody>
</table>

15.22 REAC (Reaction Parameters)

Table 15.22: For more information see Section 11.1.

<table>
<thead>
<tr>
<th>REAC (Reaction Parameters)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BOF</strong></td>
<td>Real</td>
<td>Pre-exponential Factor (Finite Rate)</td>
<td>cm³/mol/s</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>Real</td>
<td>Carbon atoms in fuel</td>
<td></td>
</tr>
<tr>
<td><strong>C_EDC</strong></td>
<td>Real</td>
<td>See Section 11.1.5</td>
<td></td>
</tr>
<tr>
<td><strong>CO_YIELD</strong></td>
<td>Real</td>
<td>Fraction of CO from the fuel</td>
<td>kg/kg</td>
</tr>
<tr>
<td><strong>CRITICAL_FLAME_TEMPERATURE</strong></td>
<td>Real</td>
<td>Suppression criterion</td>
<td>°C</td>
</tr>
<tr>
<td><strong>E</strong></td>
<td>Real</td>
<td>Activation Energy (Finite Rate)</td>
<td>kJ/kmol</td>
</tr>
<tr>
<td><strong>EDDY_DISSIPATION</strong></td>
<td>Logical</td>
<td>See Section 11.1.5</td>
<td>.TRUE.</td>
</tr>
<tr>
<td><strong>EPUMO2</strong></td>
<td>Real</td>
<td>Energy per Unit Mass Oxygen</td>
<td>kJ/kg</td>
</tr>
<tr>
<td><strong>FUEL</strong></td>
<td>Character</td>
<td>Name of Fuel (Finite Rate)</td>
<td>ETHYLENE</td>
</tr>
<tr>
<td><strong>FYI</strong></td>
<td>Character</td>
<td>Comment String (has no effect)</td>
<td></td>
</tr>
<tr>
<td><strong>H</strong></td>
<td>Real</td>
<td>Hydrogen atoms in fuel</td>
<td></td>
</tr>
<tr>
<td><strong>H2_YIELD</strong></td>
<td>Real</td>
<td>Fraction of H₂ from the fuel</td>
<td>kg/kg</td>
</tr>
<tr>
<td><strong>HEAT_OF_COMBUSTION</strong></td>
<td>Real</td>
<td>Energy per Unit Mass Fuel</td>
<td>kJ/kg</td>
</tr>
<tr>
<td><strong>HRRPUA_SHEET</strong></td>
<td>Real</td>
<td>See Section 11.1.5</td>
<td>kW/m²</td>
</tr>
<tr>
<td><strong>HRRPUV_AVERAGE</strong></td>
<td>Real</td>
<td>See Section 11.1.5</td>
<td>kW/m³</td>
</tr>
<tr>
<td><strong>ID</strong></td>
<td>Character</td>
<td>Identifier</td>
<td></td>
</tr>
</tbody>
</table>
Table 15.22: Continued

<table>
<thead>
<tr>
<th>REAC (Reaction Parameters)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IDEAL</td>
<td>Logical</td>
<td>Adjust for minor product yields</td>
<td>.FALSE.</td>
</tr>
<tr>
<td>MASS_EXTINCTION_COEFFICIENT</td>
<td>Real</td>
<td>Visibility parameter</td>
<td>m²/kg</td>
</tr>
<tr>
<td>MAXIMUM_VISIBILITY</td>
<td>Real</td>
<td>Visibility parameter</td>
<td>m</td>
</tr>
<tr>
<td>MW_OTHER</td>
<td>Real</td>
<td>Molecular Weight of OTHER</td>
<td>g/mol</td>
</tr>
<tr>
<td>N</td>
<td>Real</td>
<td>Nitrogen atoms in the fuel</td>
<td>0</td>
</tr>
<tr>
<td>N_S(N)</td>
<td>Real</td>
<td>Arrhenius Exponents (Finite Rate)</td>
<td></td>
</tr>
<tr>
<td>NU(N)</td>
<td>Real</td>
<td>Reaction stoichiometry (Finite Rate)</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>Real</td>
<td>Oxygen atoms in the fuel</td>
<td>0</td>
</tr>
<tr>
<td>OTHER</td>
<td>Real</td>
<td>Other atoms in the fuel</td>
<td>0</td>
</tr>
<tr>
<td>OXIDIZER</td>
<td>Character</td>
<td>Name of Oxidizer (Finite Rate)</td>
<td></td>
</tr>
<tr>
<td>SOOT_YIELD</td>
<td>Real</td>
<td>Fraction of soot from the fuel</td>
<td>kg/kg</td>
</tr>
<tr>
<td>SOOT_H_FRACTION</td>
<td>Real</td>
<td>Atom fraction of hydrogen in soot</td>
<td>0.1</td>
</tr>
<tr>
<td>VISIBILITY_FACTOR</td>
<td>Real</td>
<td>Visibility parameter</td>
<td>3</td>
</tr>
<tr>
<td>X_O2_LL</td>
<td>Real</td>
<td>Lower Oxygen Limit</td>
<td>mol/mol</td>
</tr>
<tr>
<td>Y_F_INLET</td>
<td>Real</td>
<td>Mass Frac. of Fuel in Burner</td>
<td>kg/kg</td>
</tr>
<tr>
<td>Y_F_LFL</td>
<td>Real</td>
<td>Lower Fuel limit (mass fraction)</td>
<td>kg/kg</td>
</tr>
<tr>
<td>Y_O2_INFTY</td>
<td>Real</td>
<td>Ambient oxygen mass fraction</td>
<td>kg/kg</td>
</tr>
</tbody>
</table>

15.23 SLCF (Slice File Parameters)

Table 15.23: For more information see Section 14.2.3.

<table>
<thead>
<tr>
<th>SLCF (Slice File Parameters)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CELL_CENTERED</td>
<td>Logical</td>
<td>Show raw data with no averaging</td>
<td>.FALSE.</td>
</tr>
<tr>
<td>FYI</td>
<td>Character</td>
<td>Comment String (has no effect)</td>
<td></td>
</tr>
<tr>
<td>MESH_NUMBER</td>
<td>Integer</td>
<td>Save only slices in this mesh</td>
<td></td>
</tr>
<tr>
<td>PART_ID</td>
<td>Character</td>
<td>Particle identifier (if needed)</td>
<td></td>
</tr>
<tr>
<td>PBX</td>
<td>Real</td>
<td>x-plane to save slice file</td>
<td></td>
</tr>
<tr>
<td>PBY</td>
<td>Real</td>
<td>y-plane to save slice file</td>
<td></td>
</tr>
<tr>
<td>PBZ</td>
<td>Real</td>
<td>z-plane to save slice file</td>
<td></td>
</tr>
<tr>
<td>QUANTITY</td>
<td>Character</td>
<td>Name of Quantity to display</td>
<td></td>
</tr>
<tr>
<td>SPEC_ID</td>
<td>Character</td>
<td>Species identifier (if needed)</td>
<td></td>
</tr>
<tr>
<td>VECTOR</td>
<td>Logical</td>
<td>Include flow vectors</td>
<td>.FALSE.</td>
</tr>
<tr>
<td>VELO_INDEX</td>
<td>Integer</td>
<td>See Section 14.3.14</td>
<td>0</td>
</tr>
<tr>
<td>XB(6)</td>
<td>Real Sextuplet</td>
<td>Min/Max coordinates of region to save</td>
<td>m</td>
</tr>
</tbody>
</table>

15.24 SPEC (Species Parameters)
### Table 15.24: For more information see Section 11.2.

<table>
<thead>
<tr>
<th>SPEC (Species Parameters)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSORBING</td>
<td>Logical</td>
<td>Absorbs radiation</td>
</tr>
<tr>
<td>CONDUCTIVITY</td>
<td>Real</td>
<td>Conductivity, $k$</td>
</tr>
<tr>
<td>DIFFUSIVITY</td>
<td>Real</td>
<td>Diffusivity, $D$</td>
</tr>
<tr>
<td>EPSILONKLJ</td>
<td>Real</td>
<td>Leonard-Jones Parameter</td>
</tr>
<tr>
<td>FORMULA</td>
<td>Character</td>
<td>Smokeview label</td>
</tr>
<tr>
<td>FYI</td>
<td>Character</td>
<td>Comment String</td>
</tr>
<tr>
<td>ID</td>
<td>Character</td>
<td>Name of species</td>
</tr>
<tr>
<td>MASS_EXTINCTION_COEFFICIENT</td>
<td>Real</td>
<td>See Section 13.3.4</td>
</tr>
<tr>
<td>MASS_FRACTION_0</td>
<td>Real</td>
<td>Initial mass fraction</td>
</tr>
<tr>
<td>MW</td>
<td>Real</td>
<td>Molecular Weight</td>
</tr>
<tr>
<td>REFERENCE_TEMPERATURE</td>
<td>Real</td>
<td>See Section 11.2.2</td>
</tr>
<tr>
<td>SPECIFIC_ENTHALPY</td>
<td>Real</td>
<td>Specific Enthalpy</td>
</tr>
<tr>
<td>SPECIFIC_HEAT</td>
<td>Real</td>
<td>Specific Heat</td>
</tr>
<tr>
<td>SIGMALJ</td>
<td>Real</td>
<td>Leonard-Jones Parameter</td>
</tr>
<tr>
<td>VISCOSITY</td>
<td>Real</td>
<td>Dynamic Viscosity, $\mu$</td>
</tr>
</tbody>
</table>

### 15.25 SURF (Surface Properties)

Table 15.25: For more information see Section 7.1.

<table>
<thead>
<tr>
<th>SURF (Surface Properties)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ADIABATIC</td>
<td>Logical</td>
<td>Adiabatic thermal BC</td>
</tr>
<tr>
<td>BACKING</td>
<td>Character</td>
<td>Back boundary condition</td>
</tr>
<tr>
<td>BURN_AWAY</td>
<td>Logical</td>
<td>See Section 8.4.6</td>
</tr>
<tr>
<td>CELL_SIZE_FACTOR</td>
<td>Real</td>
<td>See Section 8.3.7</td>
</tr>
<tr>
<td>COLOR</td>
<td>Character</td>
<td>Surface Color</td>
</tr>
<tr>
<td>CONVECTIVE_HEAT_FLUX</td>
<td>Real</td>
<td>Heat flux at surface</td>
</tr>
<tr>
<td>DUCT_PATH</td>
<td>Int. Pair</td>
<td>Pressure Zones for fans</td>
</tr>
<tr>
<td>E_COEFFICIENT</td>
<td>Real</td>
<td>Extinguishing coefficient</td>
</tr>
<tr>
<td>EMISSIVITY</td>
<td>Real</td>
<td>Emissivity (front)</td>
</tr>
<tr>
<td>EMISSIVITY_BACK</td>
<td>Real</td>
<td>Emissivity (back)</td>
</tr>
<tr>
<td>EXTERNAL_FLUX</td>
<td>Real</td>
<td>Heat flux to surface</td>
</tr>
<tr>
<td>FREE_SLIP</td>
<td>Logical</td>
<td>See Section 9.6</td>
</tr>
<tr>
<td>FYI</td>
<td>Character</td>
<td>Comment String</td>
</tr>
<tr>
<td>GEOMETRY</td>
<td>Character</td>
<td>Geometry type</td>
</tr>
<tr>
<td>H_FIXED</td>
<td>Real</td>
<td>See Section 8.2.2</td>
</tr>
<tr>
<td>HEAT_OF_VAPORIZATION</td>
<td>Real</td>
<td>For specified HRR only</td>
</tr>
<tr>
<td>HRRPUA</td>
<td>Real</td>
<td>HRR Per Unit Area</td>
</tr>
<tr>
<td>ID</td>
<td>Character</td>
<td>Identifier</td>
</tr>
<tr>
<td>IGNITION_TEMPERATURE</td>
<td>Real</td>
<td>Ignition temperature</td>
</tr>
<tr>
<td>LAYER_DIVIDE</td>
<td>Real</td>
<td>See Section 8.3.5</td>
</tr>
<tr>
<td>LEAK_PATH</td>
<td>Int. Pair</td>
<td>Pressure Zones for leakage</td>
</tr>
</tbody>
</table>
## Table 15.25: Continued

<table>
<thead>
<tr>
<th>SURF (Surface Properties)</th>
<th>MASS_FLUX(I)</th>
<th>Real Array</th>
<th>For species I</th>
<th>kg/m² s</th>
<th>0.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASS_FLUX_TOTAL</td>
<td>Real</td>
<td>Total Mass Flux</td>
<td>kg/m² s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MASS_FRACTION(I)</td>
<td>Real Array</td>
<td>For species I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MATL_ID</td>
<td>Char. Array</td>
<td>(Layer,Component)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MATL_MASS_FRACTION</td>
<td>Real Array</td>
<td>(Layer,Component)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX_PRESSURE</td>
<td>Real</td>
<td>Max over-pressure for fan</td>
<td>Pa</td>
<td>1.E12</td>
<td></td>
</tr>
<tr>
<td>MLRPUA</td>
<td>Real</td>
<td>Mass loss rate per unit area</td>
<td>kg/m²s</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>NET_HEAT_FLUX</td>
<td>Real</td>
<td>Net flux at surface</td>
<td>kW/m²</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>NO_SLIP</td>
<td>Logical</td>
<td>See Section 9.6</td>
<td>.FALSE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NPPC</td>
<td>Integer</td>
<td>Number of particles per cell</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PARTICLE_MASS_FLUX</td>
<td>Real</td>
<td>See Section 12.2.1</td>
<td>kg/m² s</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>PART_ID</td>
<td>Character</td>
<td>Lagrangian Particle ID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POROUS</td>
<td>Logical</td>
<td>See Section 9.1</td>
<td>.FALSE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLE</td>
<td>Real</td>
<td>Atmospheric profile exp.</td>
<td>m/s</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>PROFILE</td>
<td>Character</td>
<td>Name of velocity profile</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMP_EF</td>
<td>Character</td>
<td>Ramp ID for external flux</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMP_MF(I)</td>
<td>Character</td>
<td>Ramp ID for species I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMP_Q</td>
<td>Character</td>
<td>Ramp ID for HRR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMP_T</td>
<td>Character</td>
<td>Ramp ID for temp.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMP_V</td>
<td>Character</td>
<td>Ramp ID for velocity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RGB(3)</td>
<td>Int. Triplet</td>
<td>Color indices (0-255)</td>
<td>255,204,102</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROUGHNESS</td>
<td>Real</td>
<td>Wall roughness height</td>
<td>m</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>SPREAD_RATE</td>
<td>Real</td>
<td>See Section 8.4.2</td>
<td>m/s</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>STRETCH_FACTOR</td>
<td>Real</td>
<td>See Section 8.3.7</td>
<td>2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SURFACE_DENSITY</td>
<td>Real</td>
<td>See Section 8.4.6</td>
<td>kg/m²</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>TAU_EF</td>
<td>Real</td>
<td>Ramp time for external flux</td>
<td>s</td>
<td>1.</td>
<td></td>
</tr>
<tr>
<td>TAU_MF(I)</td>
<td>Real</td>
<td>Ramp time for species I</td>
<td>s</td>
<td>1.</td>
<td></td>
</tr>
<tr>
<td>TAU_Q</td>
<td>Real</td>
<td>Ramp time for HRR</td>
<td>s</td>
<td>1.</td>
<td></td>
</tr>
<tr>
<td>TAU_T</td>
<td>Real</td>
<td>Ramp time for temp.</td>
<td>s</td>
<td>1.</td>
<td></td>
</tr>
<tr>
<td>TAU_V</td>
<td>Real</td>
<td>Ramp time for velocity</td>
<td>s</td>
<td>1.</td>
<td></td>
</tr>
<tr>
<td>TEXTURE_HEIGHT</td>
<td>Real</td>
<td>Height of texture image</td>
<td>m</td>
<td>1.</td>
<td></td>
</tr>
<tr>
<td>TEXTURE_MAP</td>
<td>Character</td>
<td>Name of texture map file</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEXTURE_WIDTH</td>
<td>Real</td>
<td>Width of texture image</td>
<td>m</td>
<td>1.</td>
<td></td>
</tr>
<tr>
<td>THICKNESS(IL)</td>
<td>Real Array</td>
<td>Thickness of Layer IL</td>
<td>m</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>TEMP_BACK</td>
<td>Real</td>
<td>Back surface temperature BC</td>
<td>°C</td>
<td>20.</td>
<td></td>
</tr>
<tr>
<td>TEMP_FRONT</td>
<td>Real</td>
<td>Front surface temperature</td>
<td>°C</td>
<td>20.</td>
<td></td>
</tr>
<tr>
<td>TEMP_INNER</td>
<td>Real</td>
<td>Initial solid temperature</td>
<td>°C</td>
<td>20.</td>
<td></td>
</tr>
<tr>
<td>TRANSPARENCY</td>
<td>Real</td>
<td>Transparency of obstruction</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VEL</td>
<td>Real</td>
<td>Normal velocity</td>
<td>m/s</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>VEL_T</td>
<td>Real Pair</td>
<td>Tangential velocity comps.</td>
<td>m/s</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>VOLUME_FLUX</td>
<td>Real</td>
<td>Normal velocity x vent area</td>
<td>m³/s</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>XYZ(3)</td>
<td>Real Triplet</td>
<td>See Section 8.4.2</td>
<td>m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z0</td>
<td>Real</td>
<td>Atmospheric profile origin</td>
<td>m</td>
<td>10.</td>
<td></td>
</tr>
</tbody>
</table>
15.26 **TABLE (Table Parameters)**

Table 15.26: For more information see Section 10.

<table>
<thead>
<tr>
<th>ID</th>
<th>Character</th>
<th>IDentifier</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FYI</td>
<td>Character</td>
<td>Comment String (has no effect)</td>
<td></td>
</tr>
<tr>
<td>TABLE_DATA</td>
<td>Real Array</td>
<td>Data for one row of the table</td>
<td></td>
</tr>
</tbody>
</table>

15.27 **TIME (Time Parameters)**

Table 15.27: For more information see Section 6.2.

<table>
<thead>
<tr>
<th>DT</th>
<th>Real</th>
<th>Initial time step</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>FYI</td>
<td>Character</td>
<td>Comment String (has no effect)</td>
<td></td>
</tr>
<tr>
<td>LOCK_TIME_STEP</td>
<td>Logical</td>
<td>Do not allow time step changes</td>
<td>.FALSE.</td>
</tr>
<tr>
<td>RESTRICT_TIME_STEP</td>
<td>Logical</td>
<td>Do not allow time step to exceed initial</td>
<td>.TRUE.</td>
</tr>
<tr>
<td>SYNCHRONIZE</td>
<td>Logical</td>
<td>Sync time step of multiple meshes</td>
<td>.TRUE.</td>
</tr>
<tr>
<td>T_BEGIN</td>
<td>Real</td>
<td>Starting time for calculation</td>
<td>s 0.</td>
</tr>
<tr>
<td>T_END (TWFIN)</td>
<td>Real</td>
<td>Ending time for calculation</td>
<td>s 1</td>
</tr>
<tr>
<td>TIME_SHRINK_FACTOR</td>
<td>Real</td>
<td>See Section 6.2.3</td>
<td>1.</td>
</tr>
<tr>
<td>WALLINCREMENT</td>
<td>Integer</td>
<td>Time steps between 1D wall solution updates</td>
<td>2</td>
</tr>
</tbody>
</table>

15.28 **TRNX, TRNY, TRNZ (MESH Transformations)**

Table 15.28: For more information see Section 6.3.5.

<table>
<thead>
<tr>
<th>CC</th>
<th>Real</th>
<th>Computational coordinate</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>FYI</td>
<td>Character</td>
<td>Comment String (has no effect)</td>
<td></td>
</tr>
<tr>
<td>IDERIV</td>
<td>Integer</td>
<td>Order of polynomial transformation</td>
<td></td>
</tr>
<tr>
<td>MESH_NUMBER</td>
<td>Integer</td>
<td>Number of mesh to transform</td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>Real</td>
<td>Physical coordinate or derivative</td>
<td></td>
</tr>
</tbody>
</table>

15.29 **VENT (Vent Parameters)**

201
### Table 15.29: For more information see Section 7.4.

<table>
<thead>
<tr>
<th>VENT (Vent Parameters)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR</td>
<td>Character</td>
<td>See Section 7.5</td>
</tr>
<tr>
<td>CTRL_ID</td>
<td>Character</td>
<td>ID of Control Function</td>
</tr>
<tr>
<td>DEVC_ID</td>
<td>Character</td>
<td>ID of Controlling Device</td>
</tr>
<tr>
<td>DYNAMIC_PRESSURE</td>
<td>Real</td>
<td>See Section 9.3</td>
</tr>
<tr>
<td>FYI</td>
<td>Character</td>
<td>Comment String (has no effect)</td>
</tr>
<tr>
<td>IOR</td>
<td>Integer</td>
<td>Orientation Index</td>
</tr>
<tr>
<td>MASS_FRACTION(N)</td>
<td>Real Array</td>
<td>Mass Fraction of species N at OPEN vent</td>
</tr>
<tr>
<td>MB</td>
<td>Character</td>
<td>Mesh Boundary</td>
</tr>
<tr>
<td>OUTLINE</td>
<td>Logical</td>
<td>Draw vent as outline .FALSE.</td>
</tr>
<tr>
<td>PBX, PBY, PBZ</td>
<td>Real</td>
<td>Coordinate Plane</td>
</tr>
<tr>
<td>PRESSURE_RAMP</td>
<td>Character</td>
<td>See Section 9.3</td>
</tr>
<tr>
<td>RGB(3)</td>
<td>Integer Triplet</td>
<td>See Section 7.5</td>
</tr>
<tr>
<td>SPREAD_RATE</td>
<td>Real</td>
<td>See Section 8.4.2</td>
</tr>
<tr>
<td>SURF_ID</td>
<td>Character</td>
<td>Associated Surface 'INERT'</td>
</tr>
<tr>
<td>TEXTURE_ORIGIN(3)</td>
<td>Real Triplet</td>
<td>See Section 7.5.1</td>
</tr>
<tr>
<td>TMP_EXTERIOR</td>
<td>Real</td>
<td>Temperature at OPEN vent °C</td>
</tr>
<tr>
<td>TRANSPARENCY</td>
<td>Real</td>
<td>Transparency indicator 1.0</td>
</tr>
<tr>
<td>XB(6)</td>
<td>Real Sextuplet</td>
<td>Min/Max physical coordinates m</td>
</tr>
<tr>
<td>XYZ(3)</td>
<td>Real Triplet</td>
<td>See Section 8.4.2</td>
</tr>
</tbody>
</table>

### Table 15.30: For more information see Section 9.7.

<table>
<thead>
<tr>
<th>ZONE (Pressure Zone Parameters)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ZONE (Pressure Zone Parameters)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Character</td>
<td>Identifier</td>
</tr>
<tr>
<td>LEAK_AREA(N)</td>
<td>Real</td>
<td>Leakage area to pressure zone N m² 0</td>
</tr>
<tr>
<td>XB(6)</td>
<td>Real Sextuplet</td>
<td>Coordinates of Zone m</td>
</tr>
</tbody>
</table>
Chapter 16

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.

16.1 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.

16.2 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 13.1 and 13.5 for details.

16.3 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
The surface is still specified the same way as before, for example:

```verbatim
&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 7.1.

### 16.4 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:

```verbatim
&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.

```verbatim
&REAC ID = 'PROPANE'
SOOT\_YIELD = 0.01
C = 3.
H = 8. /
```
16.5 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

FDS and Smokeview Development Tools
Chapter 17

The FDS/Smokeview Repository

For those interested in obtaining the FDS and Smokeview source codes, either for development work or simply to compile on a particular platform, it is strongly suggested that you download onto your computer the entire FDS/Smokeview “Repository.” All project documents are maintained using the online utility Google Code Project Hosting, a free service offered by Google to support software development for open source applications. Google Code uses the Subversion (SVN) revision management system. Under this system a centralized repository containing all project files resides on a Google Code server. Subversion uses a single integer that identifies the version of the entire repository rather than of a specific file (i.e. anytime a change is made to the repository all files are incremented in version number). A record of version number when a specific file was last changed is maintained.

As an open source program, any individual can obtain a copy of the repository or retrieve specific versions repository. Only the FDS and Smokeview developers can commit changes to the repository.

The current location of the FDS repository is http://fds-smv.googlecode.com/svn/trunk/. The repository contains the following files:

1. FDS and Smokeview source code files
2. FDS and Smokeview documentation
3. Input files for software testing (Examples), verification testing, and validation testing
4. Experimental data files used for validation testing
5. Scripts and post-processing utilities used for software testing
6. Web pages and wikis

The wikis are particularly useful in describing the details of how you go about working with the Repository assets.
Chapter 18

Compiling FDS

This section describes what you need to know if you want to compile the FDS source code yourself. It is not a step by step guide, more detailed instructions can be found on Developer section of the web site at http://fire.nist.gov/fds.

If a compiled version of FDS exists for the machine on which the calculation is to be run and no changes have been made to the original source code, there is no need to re-compile the code. For example, the file 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 mpis.f90
• The parallel version of FDS uses **mpip.f90** instead of **mpis.f90**, plus additional MPI libraries need to be installed. More details on MPI can be found at the web site, along with links to the necessary organizations who have developed free MPI libraries.

### Table 18.1: Source Code Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isob.c</td>
<td>C Routine for computing isosurfaces and 3D smoke</td>
</tr>
<tr>
<td>prec.f90</td>
<td>Specification of numerical precision</td>
</tr>
<tr>
<td>smvv.f90</td>
<td>Interfaces for C routines used for Smokeview output</td>
</tr>
<tr>
<td>devc.f90</td>
<td>Derived type definitions and constants for devices’</td>
</tr>
<tr>
<td>type.f90</td>
<td>Derived type definitions</td>
</tr>
<tr>
<td>mesh.f90</td>
<td>Arrays and constants associated with each mesh</td>
</tr>
<tr>
<td>cons.f90</td>
<td>Global arrays and constants</td>
</tr>
<tr>
<td>func.f90</td>
<td>Global functions and subroutines</td>
</tr>
<tr>
<td>irad.f90</td>
<td>Functions needed for radiation solver, including RadCal</td>
</tr>
<tr>
<td>ieva.f90</td>
<td>Support routines for evac.f90</td>
</tr>
<tr>
<td>evac.f90</td>
<td>Egress computations (future capability)</td>
</tr>
<tr>
<td>pois.f90</td>
<td>Poisson (pressure) solver</td>
</tr>
<tr>
<td>radi.f90</td>
<td>Radiation solver</td>
</tr>
<tr>
<td>part.f90</td>
<td>Lagrangian particle transport and sprinkler activation</td>
</tr>
<tr>
<td>ctrl.f90</td>
<td>Definitions and routines for control functions</td>
</tr>
<tr>
<td>dump.f90</td>
<td>Output data dumps into files</td>
</tr>
<tr>
<td>read.f90</td>
<td>Read input parameters</td>
</tr>
<tr>
<td>mass.f90</td>
<td>Mass equation(s) and thermal boundary conditions</td>
</tr>
<tr>
<td>wall.f90</td>
<td>Wall boundary conditions</td>
</tr>
<tr>
<td>fire.f90</td>
<td>Combustion routines</td>
</tr>
<tr>
<td>pres.f90</td>
<td>Spatial discretization of pressure (Poisson) equation</td>
</tr>
<tr>
<td>divg.f90</td>
<td>Compute the flow divergence</td>
</tr>
<tr>
<td>init.f90</td>
<td>Initialize variables and Poisson solver</td>
</tr>
<tr>
<td>turb.f90</td>
<td>Experimental routines, mostly involving the turbulence model</td>
</tr>
<tr>
<td>velo.f90</td>
<td>Momentum equations</td>
</tr>
<tr>
<td>vege.f90</td>
<td>Experimental vegetation model</td>
</tr>
<tr>
<td>mpis.f90</td>
<td>&quot;Dummy&quot; Fortran/MPI bindings for non-MPI compilation</td>
</tr>
<tr>
<td>mpip.f90</td>
<td>MPI &quot;include&quot; statement for MPI compilation</td>
</tr>
<tr>
<td>main.f90</td>
<td>Main program for both serial and parallel versions</td>
</tr>
</tbody>
</table>
Chapter 19

Output File Formats

The output from the code consists of the file CHID.out, plus various data files that are described below. Most of these output files are written out by the routine dump.f, and can easily be modified to accommodate various plotting packages.

19.1 Diagnostic Output

The file CHID.out consists of a list of the input parameters, and an accounting of various important quantities, including CPU usage. Typically, diagnostic information is printed out every 100 time steps.

```
Iteration  8300  May 16, 2003  08:37:53
----------------------------------------------
Mesh  1, Cycle 3427
CPU/step:  2.272 s, Total CPU: 2.15 hr
Time step:  0.03373 s, Total time: 128.86 s
Max CFL number: 0.86E+00 at (21, 9, 80)
Max divergence: 0.24E+01 at (25, 30, 22)
Min divergence: -.39E+01 at (26, 18, 31)
Number of Sprinkler Droplets: 615
Total Heat Release Rate: 7560.777 kW
Radiation Loss to Boundaries: 6776.244 kW
Mesh 2, Cycle 2914
CPU/step:  1.887 s, Total CPU: 1.53 hr
Time step:  0.03045 s, Total time: 128.87 s
Max CFL number: 0.96E+00 at (21, 29, 42)
Max divergence: 0.20E+01 at (22, 20, 22)
Min divergence: -.60E+01 at (7, 26, 48)
Number of Sprinkler Droplets: 301
```

The Iteration number indicates how many time steps the code has run, whereas the Cycle number for a given mesh indicates how many time steps have been taken on that mesh. The date and time (wall clock time) are on the line starting with the word Iteration. The quantity CPU/step is the amount of CPU time required to complete a time step for that mesh; Total CPU is the amount of CPU time elapsed since the start of the run; Time step is the time step size for the given mesh; Total time is the time of the simulation;
Max/Min divergence is the max/min value of the function $\nabla \cdot \mathbf{u}$ and is used as a diagnostic when the flow is incompressible (i.e. no heating); and Max CFL number is the maximum value of the CFL number. The Radiation Loss to Boundaries is the amount of energy that is being radiated to the boundaries. As compartments heat up, the energy lost to the boundaries can grow to be an appreciable fraction of the Total Heat Release Rate. Finally, Number of Tracer Particles indicates how many passive particles are being tracked at that time.

Following the completion of a successful run, a summary of the CPU usage per subroutine is listed. This is useful in determining where most of the computational effort is being placed.

### 19.2 Heat Release Rate and Related Quantities

The heat release rate of the fire, plus other global energy-related quantities, are automatically written into a text file called CHID_hrr.csv. The format of the file is as follows

$s, kW, kW, kW, kW, kW, kg/s, atm, atm$

FDS_HRR_Time, HRR, RAD_LOSS, CONV_LOSS, COND_LOSS, BURN_RATE, ZONE_01, ZONE_02, ...

0.0000000E+000, 0.0000000E+000, ...

3.5353338E-001, 0.0000000E+000, ...

...  

HRR is the total heat release rate, RAD_LOSS is the amount of thermal radiation lost to the boundaries, CONV_LOSS is the rate of energy that is flowing out of (positive) or into (negative) the computational domain, COND_LOSS is the rate of energy that is being conducted into (positive) or out of (negative) the solid surfaces, BURN_RATE is the total mass loss rate of fuel, and ZONE_01, etc., are the background pressures of the various pressure ZONES. Note that the reported BURN_RATE is not adjusted to account for the possibility that each individual material might have a different heat of combustion.

Details of the integrated energy quantities can be found in Section 14.3.1. The background pressure is discussed in Section 9.7.

### 19.3 Device Output Data

Data associated with particular devices (link temperatures, smoke obscuration, thermocouples, etc.) specified in the input file under the namelist group DEVC is output in comma delimited format in a file called CHID_devc.csv. The format of the file is as follows

$s$, UNITS(1), UNITS(2), ..., UNITS(N.DEVC)

FDS Time, ID(1), ID(2), ..., ID(N.DEVC)

T(1), VAL(1,1), VAL(2,1), ..., VAL(N.DEVC,1)

T(2), VAL(1,2), VAL(2,2), ..., VAL(N.DEVC,2)

...

where N.DEVC is the number of devices, ID(I) is the user-defined ID of the Ith device, UNITS(I) the units, T(J) the time of the Jth dump, and VAL(I,J) the value at the Ith device at the Jth time. The files can be imported into Microsoft Excel or almost any other spread sheet program. If the number of columns exceeds 256, the file will automatically be split into smaller files.
19.4 Control Output Data

Data associated with particular control functions specified in the input file under the namelist group **CTRL** is output in comma delimited format in a file called **CHID_ctrl.csv**. The format of the file is as follows

s,status,status,status,status
FDS_Time,ID(1),ID(2),...,ID(N_CTRL)
0.00000E+000,-001, 001, ...
1.11803E-001,-001,-001, ...
...

where **N_CTRL** is the number of controllers, **ID(I)** is the user-defined ID of the **I**th control function, and plus or minus 1’s represent the state -1 = .FALSE. and +1 = .TRUE. of the **I**th control function at the particular time. The files can be imported into Microsoft Excel or almost any other spread sheet program. If the number of columns exceeds 256, the file will automatically be split into smaller files.

19.5 Gas Mass Data

The total mass of the various gas species at any instant in time is reported in the comma delimited file **CHID_mass.csv**. The file consists of several columns, the first column containing the time in seconds, the second contains the total mass of all the gas species in the computational domain in units of kg, the next lines contain the total mass of the individual species.

You must specifically ask that this file be generated, as it can potentially cost a fair amount of CPU time to generate. Set **MASS_FILE=.TRUE.** on the **DUMP** line to create this output file.

19.6 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.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 gas species in the case where complete combustion has occurred.

You must specifically ask that this file be generated. Set **STATE_FILE=.TRUE.** on the **DUMP** line to create this output file. If you forget to do this, you can easily re-run the case (remember to rename it something else!) for a few time steps. This file is created before the time iterations even begin.

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=I1,I2),J=J1,J2),K=K1,K2)
```
QUANTITY, SHORT_NAME and UNITS are character strings of length 30. The sextuplet \((I_1, I_2, J_1, J_2, K_1, K_2)\) 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 \((0, 0, 0, 0, 0, 0)\).

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 14.4 for more details about the program `fds2ascii`.

### 19.8 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`

\[
\begin{align*}
&\text{WRITE}(LU13) \quad (I_{BAR}+1, J_{BAR}+1, K_{BAR}+1) \\
&\text{WRITE}(LU13) \quad (\{(X(I)), I=0, I_{BAR}\}, \{J=0, J_{BAR}\}, \{K=0, K_{BAR}\}), \\
&\text{WRITE}(LU13) \quad (\{(Y(J)), I=0, I_{BAR}\}, \{J=0, J_{BAR}\}, \{K=0, K_{BAR}\}), \\
&\text{WRITE}(LU13) \quad (\{(Z(K)), I=0, I_{BAR}\}, \{J=0, J_{BAR}\}, \{K=0, K_{BAR}\}), \\
&\text{WRITE}(LU13) \quad (I_{BLK}(I, J, K), I=0, I_{BAR}), \{J=0, J_{BAR}\}, \{K=0, K_{BAR}\})
\end{align*}
\]

where \(X, Y\) and \(Z\) are the coordinates of the cell corners, and \(I_{BLK}\) 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 \(I_{BLK}\) is 0. Otherwise, \(I_{BLK}\) 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

\[
\begin{align*}
&\text{WRITE}(LU14) \quad (I_{BAR}+1, J_{BAR}+1, K_{BAR}+1) \\
&\text{WRITE}(LU14) \quad \text{ZERO}, \text{ZERO}, \text{ZERO}, \text{ZERO} \\
&\text{WRITE}(LU14) \quad (\{(QQ(I, J, K, N)), I=0, I_{BAR}\}, \{J=0, J_{BAR}\}, \{K=0, K_{BAR}\}, N=1, 5)
\end{align*}
\]

The five channels \(N=1, 5\) are by default the temperature \(({}^\circ\text{C})\), the \(u, v\) and \(w\) components of the velocity (m/s), and the heat release rate per unit volume (kW/m\(^3\)). Alternate variables can be specified with the input parameter `PLOT3D_QUANTITY(1:5)` on the DUMP line. Note that the data is interpolated at cell corners, thus the dimensions of the Plot3D data sets are one larger than the dimensions of the computational mesh.

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.

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

The boundary files defined under the namelist group BNDF are named CHID_n.bf \((n=0001,0002,...)\), and are written out unformatted. These files are written out from dump.f with the following lines:

```fortran
WRITE(LUBF) QUANTITY
WRITE(LUBF) SHORT_NAME
WRITE(LUBF) UNITS
WRITE(LUBF) NPATCH
WRITE(LUBF) I1,I2,J1,J2,K1,K2,IOR,NB,NM
WRITE(LUBF) I1,I2,J1,J2,K1,K2,IOR,NB,NM
  .
  .
WRITE(LUBF) TIME
WRITE(LUBF) (((QQ(I,J,K),I=11,I2),J=J1,J2),K=K1,K2)
WRITE(LUBF) (((QQ(I,J,K),I=11,I2),J=J1,J2),K=K1,K2)
  .
  .
WRITE(LUBF) TIME
WRITE(LUBF) (((QQ(I,J,K),I=11,I2),J=J1,J2),K=K1,K2)
WRITE(LUBF) (((QQ(I,J,K),I=11,I2),J=J1,J2),K=K1,K2)
  .
  .
```

`QUANTITY`, `SHORT_NAME` and `UNITS` are character strings of lengths 60, 30 and 30, respectively. `NPATCH` is the number of planes (or “patches”) that make up the solid boundaries plus the external walls. The sextuplet \((I1,I2,J1,J2,K1,K2)\) defines the cell nodes of each patch. `IOR` is an integer indicating the orientation of the patch \((\pm 1, \pm 2, \pm 3)\). You do not prescribe these. `NB` is the number of the boundary (zero for external walls) and `NM` is the number of the mesh. Note that the data is planar, thus one pair of cell nodes is the same. Presently, Smokeview is the only program available to view the boundary files.

19.10 Particle Data

Coordinates and specified quantities related to tracer particles, sprinkler droplets, and other Lagrangian particles are written to a FORTRAN unformatted (binary) file called CHID.prt5. Note that the format of this file has changed from previous versions (4 and below). The file consists of some header material, followed by particle data output every \(DT\_PART\) seconds. The time increment \(DT\_PART\) is specified on the DUMP line. It is \(T\_END/NFRAMES\) by default. The header materials is written by the following FORTRAN code in the file called dump.f90.

```fortran
WRITE(LUPF) ONE_INTEGER ! Integer 1 to check Endian-ness
WRITE(LUPF) NINT(VERSION*100.) ! FDS version number
WRITE(LUPF) N_PART ! Number of PARTicle classes
DO N=1,N_PART
  PC => PARTICLE\_CLASS(N)
  WRITE(LUPF) PC%N_QUANTITIES,ZERO_INTEGER ! ZERO_INTEGER is a place holder
  DO NN=1,PC\_N\_QUANTITIES
    WRITE(LUPF) CDATA(PC\_QUANTITIES_INDEX(NN)) ! 30 character output quantity
    WRITE(LUPF) UDATA(PC\_QUANTITIES_INDEX(NN)) ! 30 character output units
  ENDDO
ENDDO
ENDDO
```
Note that the initial printout of the number 1 is used by Smokeview to determine the Endian-ness of the file. The Endian-ness has to do with the particular way real numbers are written into a binary file. The version number is used to distinguish new versus old file formats. The parameter N_PART is not the number of particles, but rather the number of particle classes corresponding to the PART namelist groups in the input file. Every DT_PART seconds the coordinates of the particles and droplets are output as 4 byte reals:

```fortran
WRITE(LUPF) REAL(T,FB) ! Write out the time T as a 4 byte real
DO N=1,N_PART
    WRITE(LUPF) NPLIM ! Number of particles in the PART class
    WRITE(LUPF) (XP(I),I=1,NPLIM),(YP(I),I=1,NPLIM),(ZP(I),I=1,NPLIM)
    WRITE(LUPF) (TA(I),I=1,NPLIM) ! Integer "tag" for each particle
    IF (PC%N_QUANTITIES > 0) WRITE(LUPF) ((QP(I,NN),I=1,NPLIM),NN=1,PC%N_QUANTITIES)
ENDDO
```

The particle “tag” is used by Smokeview to keep track of individual particles and droplets for the purpose of drawing streamlines. It is also useful when parsing the file. The quantity data, QP(I,NN), is used by Smokeview to color the particles and droplets. Note that it is now possible with the new format to color the particles and droplets with several different quantities.

### 19.11 Profile Files

The profile files defined under the namelist group PROF are named CHID_prof_.csv (nn = 01, 02...), and are written out formatted. These files are written out from dump.f with the following line:

```fortran
WRITE(LU_PROF) T,NWP+1,(X_S(I),I=0,NWP),(Q(I),I=0,NWP)
```

After the time T, the number of node points is given and then the node coordinates. These are written out at every time step because the wall thickness and the local solid phase mesh may change over time due to the solid phase reactions. Array Q contains the values of the output quantity, which may be wall temperature, density or component density.

### 19.12 3-D Smoke File Format

3-D smoke files contain alpha values used by Smokeview to draw semi-transparent planes representing smoke and fire. FDS outputs 3-D smoke data at fixed time intervals, but unlike other output data, the file format is C, not Fortran. Note that char’s are one byte, and “int’s” and float’s are four bytes. A pseudo-code representation of the 3-D smoke file is given by:

```plaintext
endian flag (int)
is1, is2, js1, js2, ks1, ks2 (6*int)
version (int)
for each time:
    time (float)
    chars_uncompressed, chars_compressed (2*int)
    compressed_data (chars_compressed*char)
end time
```

The endian flag is an integer one. Smokeview uses this number to determine whether the computer creating the 3-D smoke file and the computer viewing the 3-D smoke file use the same or different byte swap (endian) conventions for storing floating point numbers. The opacity data is compressed using run-length encoding (RLE).
**19.13 Isosurface File Format**

Iso-surface files are used to store one or more surfaces where the specified QUANTITY is a specified value. FDS outputs iso-surface data at fixed time intervals. As with 3-D smoke, the file format is C, not Fortran. Note that char’s are one byte, short’s are two bytes and “int’s” and float’s are four bytes. A *pseudo-code* representation of the iso-surface file is given by:

```
version  (int)
len1,len2,len3  (3*int)
label1,label2,label3  ((len1+len2+len3+4)*char)
nlevels  (int)
level_1, level_2, ..., level_nlevels  (nlevels*float)
for each time:
    time  (float)
    for each level
        nvertices  (int)
        ntriangles  (int)
        vertices_1, ..., vertex_nvertices  (3*short*nvertices)
        triangles_1, ..., triangle_ntriangles  (3*(byte/short/float)*ntriangles)
    end level
end time
```

The length of each triangles_i node is one byte if the number of triangles, ntriangles, is between zero and 255 (inclusive), two bytes if ntriangles is between 256 and 65536 (inclusive) and four bytes if ntriangles is greater than or equal to 65536.
Bibliography


