NIST Special Publication 1019

Fire Dynamics Simulator (Version 4) User's Guide

Kevin McGrattan
Glenn Forney

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

NIST Special Publication 1019

Fire Dynamics Simulator (Version 4) User's Guide

Kevin McGrattan
Glenn Forney
Fire Research Division
Building and Fire Research Laboratory

in cooperation with
VTT Building and Transport, Finland

July 2004

U.S. Department of Commerce
Donald L. Evans, Secretary

Technology Administration
Phillip J. Bond, Under Secretary for Technology

National Institute of Standards and Technology
Arden L. Bement, Jr., Director

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

National Institute of Standards and Technology Special Publication 1019
Natl. Inst. Stand. Technol. Spec. Publ. 1019, 90 pages (July 2004)
CODEN: NSPUE2

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 2004

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

Preface

This guide describes how to use the Fire Dynamics Simulator (FDS). It does not provide the background theory. A companion document, called the FDS Technical Reference Guide [1], contains details about the governing equations, numerical methods and validation work. Although the User's Guide contains all the information necessary to perform fire simulations, the reader should also become familiar with some of the background theory in the Technical Reference Guide. The software and the User's Guide provide only limited guidance as to the proper prescription of input parameters.

The FDS User's Guide contains limited information on how to operate Smokeview, the companion visualization program for FDS. Its full capability is described in the "User's Guide for Smokeview Version 4" [2]. This guide also contains information on how Smokeview can be used to design FDS calculations, providing a short tutorial on the use of both models.

Disclaimer

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

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

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

Acknowledgments

The Fire Dynamics Simulator, in various forms, has been under development for almost 25 years. However, the publicly released software has only existed since 2000. While many individuals have contributed to the development of the model and its validation, a smaller group is actually responsible for writing the computer program. The FDS Technical Reference Guide contains an extensive list of model contributors. Here, however, we recognize those individuals who have done the actual programming.

Originally, the basic hydrodynamic solver was designed by Ronald Rehm and Howard Baum with programming help from Darcy Barnett, Dan Lozier and Hai Tang of the Computing and Applied Mathematics Laboratory (CAML) at NIST, and Dan Corley of the Building and Fire Research Laboratory (BFRL). Jim Sims of CAML developed the original visualization software. The direct pressure solver was written by Roland Sweet of the National Center for Atmospheric Research (NCAR), Boulder, Colorado. Kevin McGrattan expanded the basic program to include fire-specific routines, and he remains the custodian of the FDS source code. Glenn Forney developed the companion visualization program Smokeview and remains its custodian. Kuldeep Prasad added the multiple-mesh data structures, paving the way for parallel processing. William (Ruddy) Mell has added special routines to extend the model into areas such as microgravity combustion and wildland fire spread. Charles Bouldin devised the basic framework of the parallel version of the code.

Jason Floyd, a former NIST Post-Doc, wrote the mixture fraction and droplet evaporation routines. Simo Hostikka, a NIST guest researcher from VTT Building and Transport, Finland, wrote the radiation solver and the char pyrolysis routine. Although no longer at NIST, both continue to make significant contributions to the source code.

Contents

Preface **i**

Disclaimer **iii**

Acknowledgments **v**

1 Introduction **1**
1.1 Features of FDS . . . 1
1.2 What's New in FDS 4? . . . 2

2 Getting Started **3**
2.1 How to get FDS and Smokeview . . . 3
2.2 Computer Hardware Requirements . . . 3
2.3 Computer Operating System (OS) and Software Requirements . . . 4

3 Running FDS **5**
3.1 Creating the FDS Input Data File . . . 5
3.2 Starting an FDS Calculation . . . 5
3.2.1 Starting an FDS Calculation (Single Processor Version) . . . 6
3.2.2 Starting an FDS Calculation (Multiple Processor Version) . . . 6
3.3 Monitoring Progress . . . 8
3.4 Error Statements . . . 8
3.5 Reporting Bugs . . . 9

4 Setting up the Input File for FDS **11**
4.1 Preliminaries . . . 13
4.1.1 Naming the Job: The `HEAD` Namelist Group . . . 13
4.1.2 Setting Time Limits: The `TIME` Namelist Group . . . 13
4.2 The Numerical Grid . . . 14
4.2.1 Defining the Computational Domain: The `PDIM` Namelist Group . . . 14
4.2.2 Setting the Grid Size: The `GRID` Namelist Group . . . 14
4.2.3 Multiple Meshes and Parallel Processing . . . 15
4.3 Setting Global Parameters: The `MISC` Namelist Group . . . 17
4.4 Prescribing the Geometry and the Fire . . . 19
4.4.1 Prescribing Boundary Conditions: The `SURF` Namelist Group . . . 19
4.4.2 Combustion Parameters: The `REAC` Namelist Group . . . 24
4.4.3 Important Issues Related to Combustion . . . 25
4.4.4 Creating Obstructions: The `OBST` Namelist Group . . . 27

4.4.5 Creating Voids: The HOLE Namelist Group . . . 28
4.4.6 Designating Vents and Surfaces: The VENT Namelist Group . . . 29
4.4.7 Coloring Obstructions, Vents and Surfaces . . . 30
4.5 Lagrangian Particles: The PART Namelist Group . . . 32
4.6 Sprinklers and Detectors . . . 34
4.6.1 Specifying Sprinklers: The SPRK Namelist Group . . . 34
4.6.2 Specifying Heat Detectors: The HEAT Namelist Group . . . 36
4.7 Output Files . . . 38
4.7.1 Point Measurements: The THCP Namelist Group . . . 38
4.7.2 Animated Planar Slices: The SLCF Namelist Group . . . 39
4.7.3 Animated Boundary Quantities: The BNDF Namelist Group . . . 39
4.7.4 Animated Isosurfaces: The ISOF Namelist Group . . . 42
4.7.5 Static Data Dumps: The PL3D Namelist Group . . . 43
4.7.6 Extracting Numbers from the Output Data Files . . . 43

5 Special Features **47**
5.1 Stopping and Restarting Calculations . . . 47
5.2 Stretching the Grid: The TRNX, TRNY and/or TRNZ Namelist Groups . . . 48
5.3 Initial Conditions: The INIT Namelist Group . . . 49
5.4 Creating or Removing Obstructions; Opening or Closing Vents . . . 50
5.5 Extra Species . . . 51
5.6 Finite-Rate or Premixed Combustion . . . 52
5.7 Pyrolysis Models . . . 53
5.7.1 Thermoplastics . . . 53
5.7.2 Charring Fuels . . . 54
5.7.3 Liquid Fuels . . . 55
5.8 Burning Liquid Fuel Droplets . . . 55
5.9 Suppression by Water (Mixture Fraction Model Only) . . . 56
5.10 Visibility . . . 56
5.11 Layer Height and the Average Upper and Lower Layer Temperatures . . . 57
5.12 Leakage . . . 58
5.13 Fires and Flows in the Outdoors . . . 58
5.14 2D and Axially-Symmetric Calculations . . . 59
5.15 Restoring the Baroclinic Vorticity . . . 59
5.16 Fine-Tuning the Radiation Transport Model . . . 63
5.17 Defying Gravity . . . 63
5.18 Isothermal and Salt Water Simulations . . . 63
5.19 Non-rectangular Geometry . . . 64
5.20 Texture Mapping . . . 64

6 Conclusion **67**

References **69**

Appendices **69**

A Compiling the Source Code for FDS **71**
A.1 Serial Compilation 71
A.2 Parallel Compilation using Windows and MPICH 73

B Alphabetical List of Input Parameters **75**

C Output File Formats **85**
C.1 Diagnostic Output 85
C.2 Plot3D Data 86
C.3 Thermocouple Data 87
C.4 Sprinkler Data 87
C.5 Heat Release Rate 87
C.6 Gas Mass Data 88
C.7 Mixture Fraction State Relations 88
C.8 Slice Files 88
C.9 Boundary Files 89
C.10 Particle Data 89

Chapter 1

Introduction

Fire Dynamics Simulator (FDS) is a computational fluid dynamics (CFD) model of fire-driven fluid flow. The software described in this document 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 a companion document, called *Fire Dynamics Simulator (Version 4.0) – Technical Reference Guide* [1].

Smokeview is a visualization program that is used to display the results of an FDS simulation. Some examples of Smokeview are shown in this document. A detailed description can be found in a companion document, called *User's Guide for Smokeview Version 4* [2].

1.1 Features of FDS

Version 1 of FDS was publicly released in February 2000. Version 2 was publicly released in December 2001. 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 grid is fine enough. LES is the default mode of operation.

Combustion Model For most applications, FDS uses a mixture fraction combustion model. The mixture fraction is a conserved scalar quantity that is defined as the fraction of gas at a given point in the flow field that originated as fuel. The model assumes that combustion is mixing-controlled, and that the reaction of fuel and oxygen is infinitely fast. The mass fractions of all of the major reactants and products can be derived from the mixture fraction by means of "state relations," empirical expressions arrived at by a combination of simplified analysis and measurement.

Radiation Transport Radiative heat transfer is included in the model via the solution of the radiation transport equation for a non-scattering gray gas, and in some limited cases using a wide band model. The equation is solved using a technique similar to finite volume methods for convective transport, thus the name given to it is the Finite Volume Method (FVM). Using approximately 100 discrete angles,

the finite volume solver requires about 15 % of the total CPU time of a calculation, a modest cost given the complexity of radiation heat transfer. Water droplets can absorb thermal radiation. This is important in cases involving mist sprinklers, but also plays a role in all sprinkler cases. The absorption coefficients are based on Mie theory.

Geometry FDS approximates the governing equations on a rectilinear grid. The user prescribes rectangular obstructions that are forced to conform with the underlying grid.

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.

Boundary Conditions All solid surfaces are assigned thermal boundary conditions, plus information about the burning behavior of the material. Usually, material properties are stored in a database and invoked by name. 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 4?

FDS 4 has the same overall features as FDS 3, but there have been several refinements, re-organizations and bug fixes. Among the more important are:

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 3.2.2.

Multiple Meshes Improvements have been made to the multiple mesh capability allowing more flexibility in designing simulations. See Section 4.2.3 for details.

Holes One can now specify a cutout in much the same way as an obstruction. This is useful for carving doors and windows out of solid walls without having to break up the walls into pieces. See Section 4.4.5 for details.

Char Model A char model has been implemented in which a thin pyrolysis front is tracked inside a solid fuel. The front separates virgin fuel from char. Thermal properties for fuel and char must be provided by the user. See Section 5.7.2 for details.

Temperature-dependent Material Properties It is now possible to prescribe material properties of solids as a function of temperature. Note that this refinement has altered some of the pyrolysis conventions used in previous versions of FDS. Read Section 4.4.1 to see how old input files may be affected by the changes.

Lagrangian Particles The format of the input file has changed in regard to the treatment of Lagrangian particles, including sprinkler droplets and tracer particles. The underlying physical models are the same, but the book keeping within the code is different to accommodate on-going research at NIST. FDS 3 input files still run in FDS 4, but some of the effects have changed. See Section 4.5 for details about changes to particle parameters.

Layer Height A simple calculation of the layer (or interface) height has been added to FDS to enable users to compare FDS and zone model calculations or to present FDS results in a simplified manner. See Section 5.11 for details.

Chapter 2

Getting Started

Fire Dynamics Simulator (FDS) is a Fortran 90 computer program that solves the governing equations for thermally-induced fluid flow and fire. A detailed description of the equations and how they are solved numerically is described in Ref. [1]. The output of FDS is visualized using a computer program called Smokeview. The User's Guide for Smokeview is Ref. [2].

2.1 How to get FDS and Smokeview

All of the files associated with FDS and Smokeview are linked to the URL

`http://fire.nist.gov/fds`

Information about new versions, bug fixes, *etc.*, is found at the web site. Since FDS is not always backward compatible, the new executable name contains the version number **fds#.exe**. Users may want to retain copies of older FDS executables for the purpose of comparing new and old output. The graphics program Smokeview is backward compatible, and users are urged to replace the old Smokeview files with the new.

The FDS distribution consists of a self-extracting set-up program for Windows-based PCs. Unix, Linux and Mac users are directed to an anonymous FTP (File Transfer Protocol) site for source code, some compiled executables, Makefiles, *etc.* After downloading the set-up program to a PC, double-clicking on the icon walks the user through a series of steps as the program pieces get installed. The most important part of the installation is the creation of a directory (usually called **c:\nist\fds**) in which are installed the FDS and Smokeview executables, the Smokeview preference file **smokeview.ini** and a few directories containing sample cases, reference manuals, and supplemental data files. The set-up program also defines PATH variables and associates the **.smv** file extension to the Smokeview program so that one may either type Smokeview at any command line prompt or double click on any **.smv** file.

Users who have already downloaded earlier versions of FDS retain the same file structure as before, only now new files are distributed into various directories. To avoid naming conflicts, files associated with a particular version usually have that number somehow worked into the name.

2.2 Computer Hardware Requirements

FDS requires a relatively fast CPU and a substantial amount of random-access memory (RAM). For a Windows-based PC, the processor should be at least as fast as a 1 GHz Pentium III, with at least 512 MB RAM. Of course, more is better, and serious users ought to consider purchasing a computer with the fastest available CPU and largest amount of RAM. Plus, a large hard drive is needed to store the output of calculations. It is not unusual for a single calculation to generate on the order of 1 GB of output files. Most

computers now come with hard drives of at least 20 GB. For Unix-based workstations, the processor and memory should be at least as fast and as large as the PC specifications.

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 more important to obtain a fast graphics card than a fast CPU. If the computer is to run both FDS and Smokeview, then it is important to obtain a fast CPU as well.

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. Since most engineers use MS Windows, compiled versions of FDS and Smokeview are available for this OS. FDS/Smokeview run under any version of Windows except the initial release of Windows 95 that lacks the necessary libraries needed by Smokeview[1].

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

For those wishing to run FDS in parallel, MPI (Message Passing Interface) must be installed on each of the computers within the cluster. Information about installing MPI on a Windows PC is given in Appendix A. For other platforms, there are a variety of implementations of MPI that may be suitable. Consult the system administrator or hardware/software vendor.

[1]Some users of Windows ME have noticed trouble manipulating the Smokeview window. If given a choice, one ought to run under Windows 2000 or beyond.

Chapter 3

Running FDS

Running FDS is relatively simple. All of the parameters that describe a given fire scenario are typed into a text file that is referred to as the "data" or "input" file. In this document, the data file is designated as **job_name.data**, where "job_name" stands for any character string that helps to identify the simulation. All of the output files associated with the calculation have this common prefix.

In addition to the input file, there are often several external files containing input parameters for the simulation. One such file is referred to as the "database" file, for it contains parameters describing common materials and fuels. Usually, the database file is kept in a separate directory from that being used for the calculation. Files containing information about specific sprinklers are also stored along with the database file. The database and sprinkler files can be modified and/or moved to wherever appropriate.

It is suggested that a new user start with an existing data file, run it as is, and then make the appropriate changes to the input file for the desired scenario. By running a sample case, the user becomes familiar with the procedure, learn how to use Smokeview, and ensure that his/her computer is up to the task before embarking on learning how to create new input files.

3.1 Creating the FDS Input Data File

The input data file provides the program with parameters to describe the scenario under consideration. The parameters are organized into groups of related variables. For example, the group `SURF` contains parameters to describe the properties of solid surfaces. Each line of the input file contains parameters belonging to the same group. These lines are written as Fortran namelist formatted records. Each record starts with the character `&` followed immediately by the name of the namelist group (`HEAD, GRID, VENT`, *etc.*), followed by a list of the input parameters corresponding to that group, and finally terminated with a slash. Details about the input parameters can be found in Chapter 4.

3.2 Starting an FDS Calculation

There are two ways to run FDS – one way is to run on a single processor (CPU), the other way is to run on multiple CPUs. The single CPU executable (**fds#.exe**) works in a similar way to previous versions and is described presently. The parallel executable (**fds#_mpi.exe**) does not work in the traditional way; the differences are explained below. Note that the input file for both single and parallel processing is the same.

3.2.1 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.data** exists in some directory, run the program either in a DOS or Unix command prompt as follows:

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

```
fds4 < job_name.data
```

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 input file is read by FDS as standard input (indicated by the "<" sign), and diagnostic output is written out onto the screen. Unlike early versions of FDS, detailed diagnostic information is automatically written to a file **CHID.out**. Do not redirect the screen output to a file.

Unix/Linux: Change directories to where the data file for the case is, then run the code by typing

```
fds4 < job_name.data
```

The input parameters are read in as standard input, and error statements and other diagnostics are written out to the screen. To run the job in the background:

```
fds4 < job_name.data > 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.

3.2.2 Starting an FDS Calculation (Multiple Processor Version)

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

FDS uses MPI (Message-Passing Interface) [3] to allow multiple computers to run a single FDS job. Actually, the job must be broken up into multiple meshes, and a processor is assigned to work on each mesh. Each processor runs an FDS job (called a thread) for its given mesh, and the MPI handles the transfer of information between meshes. There are different implementations of MPI, much like there are different Fortran and C compilers. Each implementation is essentially a library of subroutines called from FDS that transfer data from one thread to another across a fast network. The format of the subroutine calls has been widely accepted in the community, allowing different vendors and organizations the freedom to develop better software while working within an open framework.

The way FDS is executed in parallel depends on which implementation of MPI has been installed. To avoid any conflicts, it was decided to do away with the simple command prompt style of running the single

processor version of FDS. Instead, the parallel version of FDS looks for the name of the input file by opening a one-line text file called **fds.data** and reading the name of the input file on the first line. The file **fds.data** should contain only the name of the real input file **job_name.data** on the first line and nothing else. Note that all these file names are case-sensitive.

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

```
mpirun config.txt
```

where `config.txt` is a text file containing the name and location of the FDS executable, 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

```
exe \\machine1\nist\fds\fds4_mpi.exe
dir \\machine1\nist\fds\samples\
hosts
machine1 2
machine2 1
machine3 2
```

Note that all the computers must be able to access the executable and the working directory on `machine1`. This is achieved under Windows by "sharing." Under Unix/Linux, the process involves cross-mounting the file systems of the various machines. 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.

On the Linux cluster in the Building and Fire Research Lab at NIST, LAM/MPI, a free implemenation 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 fds4_mpi
```

where the 5 indicates that 5 processors are to be used. In this case, the executable **fds4_mpi** is located in the working directory. To make the process run in the background

```
mpirun -np 5 fds4_mpi > job_name.err &
```

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

In Appendix A, there is a detailed description of how one compiles and runs FDS in parallel under Windows using MPICH. For more information about LAM/MPI, visit the web site

```
http://www.lam-mpi.org/
```

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.

3.3 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 one can see how far along the program has progressed. At any time during a calculation, Smokeview can be run and the progress can be checked visually. To stop a calculation before its scheduled time, either kill the process, or preferably create a file in the same directory as the output files called **CHID.stop**. The existence of this file stops the program gracefully, causing it to dump out the latest flow variables for viewing in Smokeview.

Since calculations can be hours or days long, there is a restart feature in FDS. Details of how to use this feature are given in Section 5.1. Briefly, specify at the beginning of calculation how often a "restart" file should be saved. Should something happen to disrupt the calculation, like a power outage, the calculation can be restarted from the time the last restart file was saved.

3.4 Error Statements

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

Input File Error: 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 / is put at the end of the record. Make sure that the right type of information is being provided for each parameter, like whether one real number is expected, or several integers, or whatever. Make sure there are no non-ASCII characters being used, as can sometimes happen when text is cut and pasted from other applications or word-processing software. Make sure zeros are zeros and O's are O's. Make sure 1's are not !'s. Make sure apostrophes are used to designate character strings. Make sure the text file on a Unix/Linux machine was not created on a DOS machine, and *vice versa*. Make sure that all the parameters listed are still being used – new versions of FDS often drop or change parameters to force users to re-examine old input files.

Numerical Instability: It is possible that during an FDS calculation the flow velocity at some location in the domain can increase due to numerical error causing the time step size to decrease to a point where logic in the code decides that the results are unphysical and stops the calculation with an error message in the file **CHID.out**. In these cases, FDS ends by dumping out one final Plot3D file giving the user some means by which to see where the error is occurring within the computational domain. Usually, a numerical instability can be identified by fictitiously large velocity vectors emanating from a small region within the domain. Common causes of such instabilities are grid cells that have an aspect ratio larger than 2 to 1, high speed flow through a small opening, a sudden change in the heat release rate, or any number of sudden changes to the flow field. There are various ways to solve the problem, depending on the situation. Try to diagnose and fix the problem before reporting it. It is difficult for anyone but the originator of the input file to diagnose the problem.

Inadequate Computer Resources: The calculation might be using more RAM than the machine has, or the output files could have used up all the available disk space. In these situations, the computer may or may not produce an intelligible error message. Sometimes the computer is just unresponsive. It is the user's responsibility to ensure that the computer has adequate resources to do the calculation. Remember, there is no limit to how big or how long FDS calculations can be – it depends on the resources of the computer. For any new simulation, try running the case with a modest-sized grid, 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

Run-Time Error: 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, try to systematically simplify the input file until the error goes away. This process usually brings to light some feature of the calculation responsible for the problem and helps in the debugging.

Poisson Initialization: Sometimes at the very start of a calculation, an error appears stating that there is a problem with the "Poisson initialization." The equation for pressure in FDS is known as the Poisson equation. The Poisson solver consists of large system of linear equations that must be initialized at the start of the calculation. Most often, an error in the initialization step is due to a grid 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 `GRID` and Physical DIMension (`PDIM`) lines in the input file.

3.5 Reporting Bugs

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 bug reporting 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 grid 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 obvious possibilities, such as memory size, disk space, *etc.* If that does not solve the problem, report the bug 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. The input file should have no references to external databases. In this way, the developers can quickly run the problem input file and hopefully diagnose the problem.

Chapter 4

Setting up the Input File for FDS

The first step in performing a calculation is to generate a text input file that provides the program with all of the necessary information to describe the scenario under consideration. The most important inputs determine the physical size of the overall rectangular domain, the grid dimensions, and the additional geometrical features. Next, the fire and other boundary conditions must be specified. Finally, there are a number of parameters that customize the output files to capture the most important flow quantities. Input data is prescribed by writing a small file that uses the namelist formatted records. Each line of the file begins with the character `&` followed immediately by the name of the namelist group (`HEAD, GRID, VENT`, *etc.*), followed by a space or comma delimited list of the input parameters corresponding to that group. Each list is terminated with a slash. Note that the parameters listed are only those that need to be changed from the default. The structure of an input file is shown below.

```
&HEAD CHID='sample',TITLE='A Sample Input File' /
&GRID IBAR=24,JBAR=24,KBAR=48 /
&PDIM XBAR0=-.30,XBAR=0.30,YBAR0=-.30,YBAR=0.30,ZBAR=1.2 /
&TIME TWFIN=10. /
&MISC RADIATION=.FALSE. /
&SURF ID='burner',HRRPUA=1000. /
&OBST XB=-.20,0.20,-.20,0.20,0.00,0.05,SURF_IDS='burner','INERT','INERT' /
&VENT CB='XBAR' ,SURF_ID='OPEN' /
&VENT CB='ZBAR' ,SURF_ID='OPEN' /
&SLCF PBY=0.,QUANTITY='TEMPERATURE' /
&BNDF QUANTITY='HEAT_FLUX' /
```

The parameters in the input file can be integers (`IBAR=24`), real numbers (`XBAR=0.30`), groups of real numbers (`XB=-.20,0.20,...`), character strings (`CHID='sample'`), groups of character strings (`SURF_IDS = 'burner' 'INERT' 'INERT'`), or logicals (`RADIATION=.FALSE.`). A logical parameter is either `.TRUE.` or `.FALSE.` – the periods are a Fortran convention. Character strings that are listed in this User's Manual ought to be copied exactly as written – the code is case sensitive and underscores *do* matter. Also 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.

Input parameters can be separated by either a comma, space, or line break. Comments and notes can be written into the file so long as nothing comes between the ampersand `&` and the slash `/` except appropriate parameters corresponding to that particular namelist group. Note that FDS is case-sensitive. Copy exactly the parameter names from this manual and do not assume that the program understands if the case is changed.

Rarely does anyone actually write an input file from scratch. Usually, one takes a sample input file that has been distributed with the FDS release and modifies it appropriately. It is strongly encouraged 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 gridded coarsely so that the run times are less than an hour and corrections can easily be made without wasting too much time.

4.1 Preliminaries

The first few lines in the input file handle some custodial details like naming the job and establishing the time of the simulation. The name of the job is important because often a project involves numerous simulations in which case the names of the individual simulations can help organize the effort.

4.1.1 Naming the Job: The `HEAD` Namelist Group

The first thing to do when setting up an input file is to give the job a name. The namelist group `HEAD` contains two parameters. `CHID` is character string of 30 characters or less used to tag output files with a given character string. If, for example, `CHID='sample'`, it is convenient to name the input data file **sample.data** so that the input file can be associated with the output files. No periods or spaces are allowed in `CHID` because the output files are tagged with suffixes that are meaningful to certain computer operating systems. `TITLE` is a character string of 60 characters or less that describes the problem.

```
&HEAD CHID='sample',TITLE='A Sample Input File' /
```

4.1.2 Setting Time Limits: The `TIME` Namelist Group

`TIME` is the name of a group of parameters defining the time duration of the simulation and the initial time step used to advance the solution of the discretized equations. Usually, only the duration of the simulation is required on this line, via the parameter `TWFIN` (Time When FINished). The default is 1 s. If `TWFIN` is set to zero, only the set-up work is performed, allowing one to quickly check the geometry in Smokeview.

The initial time step size can also be prescribed with `DT`. This parameter is normally set automatically by dividing the size of a grid cell by the characteristic velocity of the flow. During the calculation, the time step is adjusted so that the CFL condition is satisfied. The default value of `DT` is $5\,(\delta x\,\delta y\,\delta z)^{\frac{1}{3}}/\sqrt{gH}$ s, where δx, δy, and δz are the dimensions of the smallest grid cell, H is the height of the computational domain, and g is the acceleration of gravity.

One additional parameter in the `TIME` group is `SYNCHRONIZE`, a logical flag (`.TRUE.` or `.FALSE.`) indicating that in a parallel 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 4.2.3.

4.2 The Numerical Grid

All FDS calculations must be performed within a domain that is made up of rectangular meshes, each with its own rectilinear grid. All obstructions, vents, *etc.* are forced to conform with the numerical grid(s). To establish a grid, first specify the overall physical dimensions of the rectangular grid via the `PDIM` namelist group. Second, specify the number of grid cells spanning each coordinate direction via the `GRID` namelist group. Finally, if desired, the grid cells can be stretched or shrunk in two of three coordinate directions via the `TRNX`, `TRNY`, and/or `TRNZ` groups (See Section 5.2). For cases in which more than one grid is used in the calculation, see Section 4.2.3 below for guidelines.

4.2.1 Defining the Computational Domain: The `PDIM` Namelist Group

`PDIM` is the name of the group of parameters defining the size of the physical domain. The coordinate system spanned by these dimensions conforms to the right hand rule (See Fig. 4.1). The physical domain is a single right parallelepiped, *i.e.* a box. The origin of the domain is the point `(XBAR0,YBAR0,ZBAR0)`, and the opposite corner of the domain is at the point `(XBAR,YBAR,ZBAR)`. By default, `XBAR0, YBAR0, ZBAR0` are zero, in which case the physical dimensions of the domain are given as `XBAR`, `YBAR` and `ZBAR` in units of meters. Unless otherwise directed, the domain is subdivided uniformly to form a grid of `IBAR` by `JBAR` by `KBAR` cells specified by the `GRID` namelist group. If it is desired that the grid cells not be uniform in size, then the namelist groups `TRNX`, `TRNY` and/or `TRNZ` may be used to alter the uniform gridding (See Section 5.2).

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

4.2.2 Setting the Grid Size: The `GRID` Namelist Group

The namelist group `GRID` contains the dimensions of the computational grid. The grid consists of `IBAR` cells in the x direction, `JBAR` cells in the y direction, and `KBAR` cells in the z direction. Usually, the z direction is assumed to be the vertical direction. The longer horizontal dimension should be taken as the x-direction. Note that it is best if the grid cells are close to cubes, that is, the length, width and height of the cells ought to be roughly the same. Also, because an important part of the calculation uses a Poisson solver based on Fast Fourier Transforms (FFTs), the dimensions of the grid should each be of the form $2^l 3^m 5^n$, where l, m and n are integers. For example, $64 = 2^6$, $72 = 2^3 3^2$ and $108 = 2^2 3^3$ are good grid dimensions. However, 37, 99 and 109 are not.

```
&GRID IBAR=64,JBAR=32,KBAR=32 /
```

Following 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				

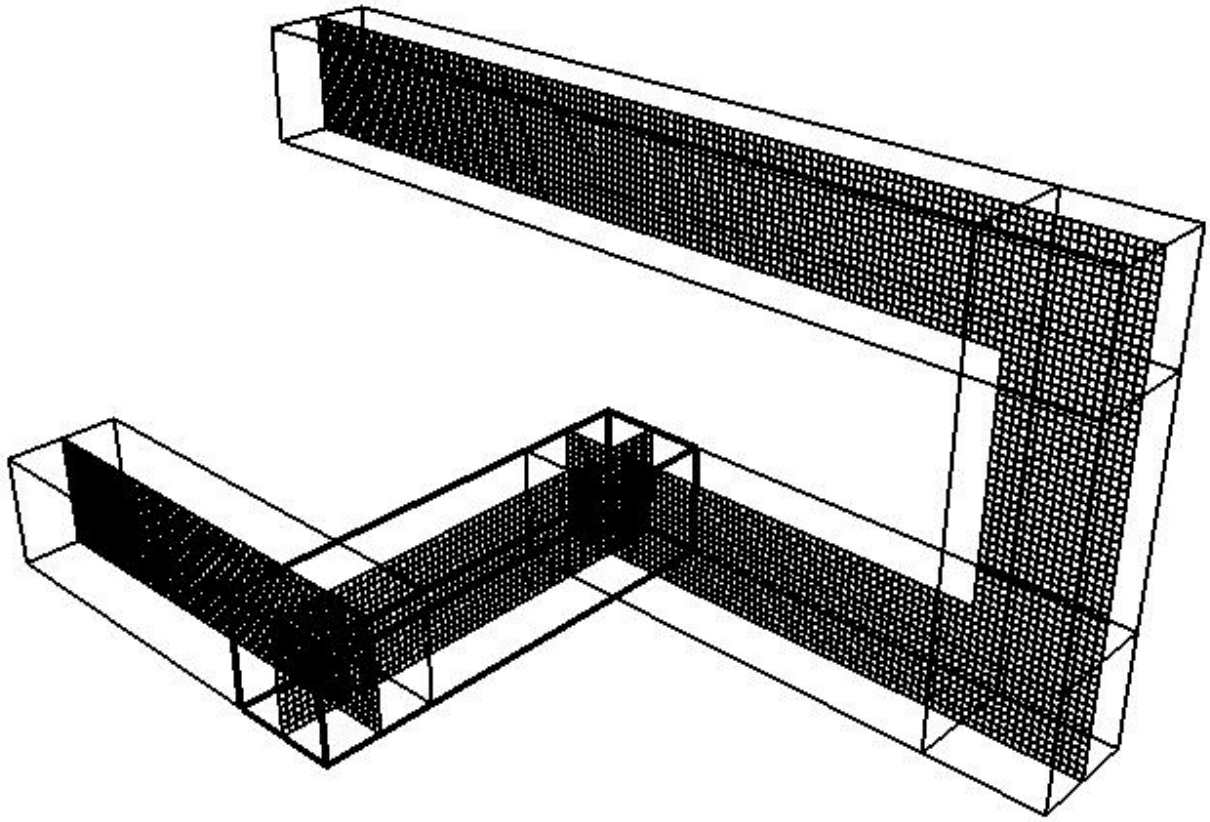


Figure 4.1: **An example of a multiple-mesh geometry.**

Note that `IBAR`, `JBAR` and `KBAR` should be at least 4, except for two-dimensional calculations, in which case `JBAR=1`.

4.2.3 Multiple Meshes and Parallel Processing

The term "multiple meshes" means that the computational domain consists of more than one rectangular mesh, usually connected although this is not required. In each mesh, the governing equations are 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. Also, to run FDS in parallel, one needs to break up the computational domain into multiple meshes so that each processor receives one mesh to work on. Whether the calculation is to be run on a single processor, or on multiple processors, the rules of prescribing multiple meshes are similar, with some issues to keep in mind. Here is a list of guidelines and warnings about the use of multiple meshes.

- If more than one mesh is used, there should be a `GRID` and `PDIM` 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 has precedence over a mesh listed second if the two meshes overlap. Meshes can overlap, abut, or not touch at all. In the last case, essentially two separate

calculations are performed with no communication at all between them. Obstructions and vents are entered in terms of the overall coordinate system and need not apply to any one particular mesh. Each mesh checks the coordinates of all the geometric entities and decides whether or not they are to be included.

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

- Information from other meshes is received only at the exterior boundary of a given mesh. This means that a mesh that is completely embedded within another receives information at its exterior boundary, but the larger mesh receives no information from the mesh embedded within. Essentially, the larger, usually coarser, mesh is doing its own simulation of the scenario and is not affected by the smaller, usually finer, mesh embedded within it. Details within the fine grid, especially related to fire growth and spread, may not be picked up by the coarse grid. In such cases, it is preferable to isolate the detailed fire behavior within one mesh, and position coarser meshes at the exterior boundary of the fine mesh. Then the fine and coarse meshes mutually exchange information.

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

- Be careful when using the shortcut convention of declaring an entire face of the domain to be an `OPEN` vent. Every grid takes on this attribute. See Section 4.4.6 for more details.

- If more than one mesh is used in a calculation, there can be no background pressure rise. Essentially, the different compartments are assumed to leak.

- In a parallel calculation, one can force the time steps in all meshes to be the same by setting

  ```
  SYNCHRONIZE=.TRUE.
  ```

  on the `TIME` line. 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 a select set of meshes. To do this, add `SYNCHRONIZE=.TRUE.` to the appropriate `GRID` lines. Do not then add `SYNCHRONIZE=.TRUE.` to the `TIME` line as it over-rides the `GRID` line settings.

- If an 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 may not account for it, in which case information is not transferred properly between meshes.

- When running a case with multiple meshes in parallel, the efficiency of the calculation can be checked as follows: (1) Set `SYNCHRONIZE=.TRUE.` on the `TIME` line, (2) Let the program run several hundred time steps, (3) Calculate the difference in wall clock time between two 100 iteration print outs in the file **CHID.out** (see Section C.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.

4.3 Setting Global Parameters: The `MISC` Namelist Group

`MISC` is the namelist group of miscellaneous input parameters. Only one `MISC` line should be entered in the data file. The `MISC` parameters vary in scope and degree of importance. The most important parameter in this category is the one that determines whether a Large Eddy Simulation (LES) calculation is to be performed, or whether a Direct Numerical Simulation (DNS) is to be performed. By default, an LES calculation is performed. If a DNS calculation is desired, enter `DNS=.TRUE.` on the `MISC` line. An example of a `MISC` line is

```
&MISC SURF_DEFAULT='CONCRETE',REACTION='METHANE',
      DATABASE='c:\nist\fds\database4\database4.data' /
```

This establishes that all bounding surfaces are to be made of `CONCRETE` unless otherwise indicated, that the combustion stoichiometry is for `METHANE`, and that the definition of `CONCRETE`, `METHANE`, and various other keywords throughout the input file are found in the file defined by `DATABASE`. Other inputs found on the `MISC` line include:

`DATABASE` A character string indicating the name of a file that contains information about surface materials and reaction parameters for various fuels. The `DATABASE` file does not need to be designated if none of its entries are to be used.

`DATABASE_DIRECTORY` A character string indicating the full path name of the directory where the database and sprinkler files are stored. If `DATABASE_DIRECTORY` is specified, there is no need to specify a `DATABASE` file, it is assumed to be **database4.data**.

`SURF_DEFAULT` Character string indicating which of the listed `SURF IDs` is to be considered the default. The default is `'INERT'`. `SURF` is a namelist group describing the properties of vents and surfaces, and is discussed in Section 4.4.1.

`REACTION` Character string indicating which of the listed groups of reaction (`REAC`) parameters are to be used. The default is `'PROPANE'`, meaning that unless `REACTION` is specified, it is assumed that the fuel is propane. See Section 4.4.2 for a description of reaction parameters.

`TMPA` Ambient temperature in degrees Celsius. (Default 20 °C)

`U0, V0 and W0` Initial values of velocity components in m/s. These can be used to prescribe an initial wind through the domain. (Default 0 m/s)

`TMPO` Temperature outside the computational domain, in degrees Celsius. (Default 20 °C)

`NFRAMES` Default number of output dumps per calculation. Thermocouple data, slice data, particle data, and boundary data is saved every `TWFIN/NFRAMES` unless otherwise specified with `DTSAM` on the `THCP`, `SLCF`, and `BNDF` namelist lines. (Default 1000)

`DTPAR` Time increment in seconds between Lagrangian particle insertions at solid surfaces. If more particles are desired, lower the input value of this parameter. (Default 0.05 s)

`DTSPAR` Time increment in seconds between droplet insertions at sprinklers. (Default 0.05 s)

`DTSAM_PART` Time increment between tracer particle (and sprinkler droplet) data dumps in seconds. These dumps add to a file called **CHID.part** which can be used to produce an animation of the flow field. (Default `TWFIN/NFRAMES`)

`NPPS` Number of particles per set. The maximum number of particles that can be output into the file **CHID.part** every `DTSAM` seconds. (Default 100000)

`MAXIMUM_DROPLETS` Maximum number of Lagrangian particles per mesh. (Default 500000)

4.4 Prescribing the Geometry and the Fire

Most of the 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. The following namelist groups describe how to prescribe the boundary conditions and the obstructions and vents to which the boundary conditions are assigned.

4.4.1 Prescribing Boundary Conditions: The SURF Namelist Group

SURF is the namelist group that defines boundary conditions for all solid surfaces or openings within or bounding the flow domain. The physical coordinates of obstructions or vents are listed in the OBST and VENT namelist groups below. Boundary conditions for the obstructions and vents are prescribed by referencing the appropriate SURF line(s) whose parameters are described presently.

The default boundary condition for all solid surfaces is that of a cold, inert wall. If only this boundary condition is needed, there is no need to add any SURF lines to the input file. If additional boundary conditions are desired, they are to be listed one boundary condition at a time. Each SURF line consists of an identification string ID='...' to allow references to it by an obstruction or vent. Thus, on each OBST and VENT line, the character string SURF_ID='...' indicates the ID of the SURF line containing the desired boundary condition parameters. If a particular SURF line is to be applied as the default boundary condition, CONCRETE for example, set SURF_DEFAULT='CONCRETE' on the MISC line.

Fire (Mixture Fraction Model) A fire is basically modeled as the ejection of pyrolyzed fuel from a solid surface or vent that burns when mixed with oxygen. This is the default mixture fraction model of combustion. Specify either a heat release rate per unit area **or** a heat of vaporization at the fuel surface. The stoichiometry of the reaction is set by the parameter REACTION on the MISC line. All of the species associated with the combustion process are accounted for by way of the mixture fraction variable and should not be explicitly prescribed. The exception to this rule is where a non-reacting gas is introduced into the domain that merely serves as a diluent, like CO_2 from an extinguisher or H_2O from evaporated sprinkler droplets (see Section 5.5 for details). If a finite rate combustion model is desired instead of the default mixture fraction model, see Section 5.6.

Following is a list of parameters that are prescribed on a SURF line to designate a fire using the mixture fraction approach.

HRRPUA Heat Release Rate Per Unit Area (kW/m^2). This parameter is used to control the burning rate of the fuel, as in the case of a prescribed fire using a gas burner. If all one desires is a fire of a given size, then HRRPUA is the only thing that need be set, 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 Boundary Conditions** below to learn how to ramp the heat release rate up and down.

HEAT_OF_VAPORIZATION (kJ/kg). This is an alternative to HRRPUA. This is the amount of energy required to vaporize a solid or liquid fuel once it has reached its ignition temperature TMPIGN. If it is desired that the burning rate of the fuel be dependent on heat feedback from the fire, use this parameter rather than HRRPUA. Do *not* specify HRRPUA and HEAT_OF_VAPORIZATION on the same SURF

line. They are mutually exclusive inputs. Also, if HEAT_OF_VAPORIZATION is specified for a given material, something else must serve as an ignition source to ignite the material.

BURN_AWAY If a burning object is to disappear from the calculation once it is exhausted of fuel, set BURN_AWAY=.TRUE.. Use this parameter cautiously. If an object has the potential of burning away, a significant amount of extra memory has to be set aside to store additional surface information as the rectangular block is eaten away. Note that in previous versions of FDS (3 and lower), the parameter DENSITY or SURFACE_DENSITY controlled both burn out and the removal of burnt objects. This is no longer true. If BURN_AWAY is prescribed as a SURF parameter, then a solid object with this SURF_ID disappears from a calculation as the mass of each of its grid cells is consumed. The mass of each grid cell is the volume of the grid cell multiplied by the DENSITY of the obstruction. BURN_AWAY can be applied to thermally-thin or thick materials, as long as a DENSITY is also prescribed and the obstruction is at least one grid cell wide. Note also that if BURN_AWAY is prescribed, the SURF should be applied to the entire object, not just a face of the object because it is unclear how to handle edges of solid obstructions that have different SURF_IDs on different faces. Also note that the amount of combustible fuel equals the DENSITY times the volume of the grid cell. If the volume of the obstruction changes because it has to conform to the uniform grid, FDS does **not** adjust the burning rate to account for this as it does with various quantities associated with areas, like HRRPUA.

Thermal Boundary Conditions: There are four types of thermal boundary conditions: fixed temperature solid surface, fixed heat flux solid surface, thermally-thick solid or thermally-thin sheet. For a given boundary condition (*i.e.* for the same SURF line), choose only one of these. For a solid surface of fixed temperature, set TMPWAL to be the surface temperature in units of °C. For a solid surface of fixed convective heat flux, set HEAT_FLUX to be the convective heat flux in units of kW/m^2. If HEAT_FLUX is positive, the wall heats up the surrounding gases. If HEAT_FLUX is negative, the wall cools the surrounding gases.

A solid surface that heats up due to radiative and convective heat transfer from the surrounding gas can be either thermally-thick or thermally-thin. For a thermally-thick solid, prescribe the thermal conductivity KS (W/m·K), DENSITY (kg/m^3), specific heat C_P (kJ/kg/K), and the thickness DELTA (m) of the material[1]. Both KS and C_P can be functions of temperature. DENSITY cannot be a function of temperature. See the discussion of RAMPs below for more details. The prescription of the thermal conductivity directs the code to perform a one-dimensional heat transfer calculation across the thickness of the material [2]. The thickness is *not* the same as the thickness of the entire wall, but rather the lining material that forms the outermost layer of the wall. See discussion below for more details.

For thermally-thin wall linings, prescribe C_DELTA_RHO, the product of the specific heat (kJ/kg/K), density (kg/m^3), and thickness (m) of the liner. A thermally-thin liner is assumed to be the same temperature throughout its width. These three parameters may be prescribed individually using C_P (kJ/kg/K), DELTA (m) and DENSITY (kg/m^3), in which case, C_P can be made temperature-dependent. Note that the absence of thermal conductivity directs the code to assume the material is thermally-thin instead of thick. If the thermal conductivity is prescribed, a thermally-thick calculation is performed.

Fixed temperature or fixed heat flux boundary conditions are easy to apply, but only of limited usefulness in real fire scenarios. In most cases, walls, ceilings and floors are made up of several layers of lining materials, the most important of which is the outermost layer. FDS only considers the thermal properties for

[1] In older versions of FDS, the thermal diffusivity ALPHA (m^2/s) was used to represent $k/\rho/c_p$. This parameter can still be used, but individual prescription of k, ρ and c_p is preferred.

[2] The default number of nodes used in the one-dimensional heat conduction calculation into a thermally-thick solid is 20. To change this parameter, WALL_POINTS can be added to the SURF line. Note that the nodes are not uniformly spaced, but rather are clustered so that the first cell in the solid is approximately 0.1 mm. This value may be changed by adding the parameter DX_SOLID to the SURF line. See the FDS Technical Reference Guide [1] for details about the heat transfer calculation.

this outermost layer. It is assumed that this layer backs up to an air gap at ambient temperature (like a sheet of gypsum board attached to wood studs), or it backs up to an insulated material in which case no heat is lost to the backing material, or it backs up to the room on the other side of the wall. By default, it is assumed that the wall liner backs up to an air gap. If the wall liner is assumed to back up against an insulating material, like a sheet of steel attached to a fiber insulating board, the expression `BACKING='INSULATED'` on the `SURF` line prevents any heat loss from the back side of the material. An example of where this might be used is in home furnishing. Recent work by Fleischmann and Chen [4] on the ignition properties of upholstery suggests that treating a fabric covered slab of polyurethane foam as thermally-thin produces a slightly better correlation than thermally-thick. If their thermally-thin data is used, the attribute `BACKING='INSULATED'` should be invoked. Finally, if it is desired that the heat transfer through the wall into the space behind the wall, the attribute `BACKING='EXPOSED'` should be listed. This feature only works if the wall is one grid cell thick, and if there is a non-zero volume of computational domain on the other side of the wall. Obviously, if the wall is an external boundary of the domain, the heat is lost to the void.

The emissivity of a solid surface may be set with `EMISSIVITY`, which is 1 by default. If the wall lining material is flammable, set its ignition temperature with `TMPIGN`, the temperature (°C) at which the material begins burning. This is only set if the wall liner is thermally-thick or thermally-thin. Heat fluxes to solid surfaces are both convective and radiative. If `TMPIGN` is set, a heat release rate per unit area `HRRPUA` or `HEAT_OF_VAPORIZATION` should also be given. (Default: `TMPIGN` is 5000 °C, *i.e.* no burning occurs unless this parameter is explicitly prescribed.)

The following are a few examples of `SURF` lines. These and several others are found in the `DATABASE` file.

```
&SURF ID          = 'CONCRETE'
      FYI         = 'Thermally-thick material'
      KS          = 1.0
      C_P         = 0.88
      DENSITY     = 2000.
      DELTA       = 0.2 /
&SURF ID          = 'UPHOLSTERY'
      FYI         = 'Assumed thermally-thin material'
      C_DELTA_RHO = 1.29
      BACKING     = 'INSULATED'
      TMPIGN      = 280.
      DENSITY     = 20.0
      HEAT_OF_VAPORIZATION=2500. /
&SURF ID          = 'SHEET METAL'
      FYI         = 'Thermally-thin material'
      C_DELTA_RHO = 4.7 /
```

Velocity Boundary Condition Velocity boundary conditions affect both the normal and tangential components of the velocity vector at boundaries. The normal component of velocity is controlled by the parameter `VEL`. If `VEL` is negative, the flow is entering the computational domain. If `VEL` is positive, the flow is exiting the domain. Sometimes it is desired that a given volume flux through a vent be prescribed rather than a velocity. If this is the case then `VOLUME_FLUX` can be prescribed instead of `VEL`. The units are m^3/s. If the flow is entering the computational domain, `VOLUME_FLUX` should be a negative number. Note that either `VEL` or `VOLUME_FLUX` should be prescribed, not both. The choice depends on whether an exact velocity is desired at a given vent, or whether the given volume flux is desired. The dimensions of the vent that are prescribed usually change because the prescribed vent dimensions are sometimes altered so that the vent

edges line up with grid cells. Also note that a SURF group with a VOLUME_FLUX prescribed should only be called by a VENT, not an OBST. Finally, note that if HRRPUA or HEAT_OF_VAPORIZATION is prescribed, no velocity should be prescribed. The combustible gases are ejected at a velocity computed by the code.

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

```
&SURF ID='BLOWER',VEL=-1.2,TMPWAL=50. /
```

The vent with SURF_ID='BLOWER' would blow 50 °C air at 1.2 m/s into the flow domain. Making VEL positive would suck air out, in which case TMPWAL would not be necessary.

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

In the case of a blowing vent (or even a solid surface), it is possible to prescribe both the normal and tangential components of the flow (or just the tangential). The normal component is specified with VEL as described above. The tangential is prescribed via a pair of real numbers VEL_T representing the desired tangential velocity components. For example, the line

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

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

Time Dependent Boundary Conditions 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 and TAU_V indicate that thermal or hydrodynamic quantities are to ramp up to their prescribed values in TAU seconds and remain there. TAU_Q is the characteristic ramp-up time of the heat release rate per unit area HRRPUA or wall temperature TMPWAL. TAU_V is the ramp-up time of the normal velocity at a surface VEL or the volume flux VOLUME_FLUX. If TAU_Q is positive, then the heat release rate ramps up like $\tanh(t/\tau)$. If negative, then the HRR ramps up like $(t/\tau)^2$. If the fire ramps up following a t^2 curve, it remains constant after TAU_Q seconds. These rules apply to TAU_V as well. The default value for all TAUs is 1 s. If something other than a tanh or t^2 ramp up is desired, then a user-defined burning history must be input. To do this, set RAMP_Q or RAMP_V equal to a character string designating the ramp function to use for that particular surface type, then somewhere in the input file generate lines of the form:

```
&RAMP ID='rampname1',T= 0.0,F=0.0 /
&RAMP ID='rampname1',T= 5.0,F=0.5 /
&RAMP ID='rampname1',T=10.0,F=0.7 /
                    .
```

```
                        .
                        .
 &RAMP ID='rampname2',T= 0.0,F=0.0 /
 &RAMP ID='rampname2',T=10.0,F=0.3 /
 &RAMP ID='rampname2',T=20.0,F=0.8 /
                        .
                        .
                        .
```

Here, T is the time, and F indicates the fraction of the heat release rate, wall temperature, velocity, mass fraction, *etc.*, to apply. Linear interpolation is used to fill in intermediate time points. Be sure that the prescribed function starts at T=0., the ignition time. Note that the TAUs and the RAMPs are mutually exclusive. For a given surface quantity, both cannot be prescribed.

As an example, the simple blowing vent from above can be controlled via the lines

```
&SURF ID='BLOWER',VEL=-1.2,TMPWAL=50.,
      RAMP_V='BLOWER RAMP',
      RAMP_Q='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_Q and RAMP_V, respectively.

Temperature-Dependent Boundary Conditions Certain thermal parameters like the specific heat of the solid (C_P) or the thermal conductivity of the solid (KS) can be made temperature-dependent using the RAMP convention. As an example, take the material properties for marinite, a wall material suitable for high temperature applications.

```
&SURF ID='MARINITE'
      RGB = 0.70,0.70,0.70
      BACKING = 'EXPOSED'
      EMISSIVITY=0.8
      DENSITY = 737.
      RAMP_C_P='rampcp'
      RAMP_KS='rampks'
      DELTA=0.0254 /
&RAMP ID='rampks',T= 24.,F=0.13 /
&RAMP ID='rampks',T=149.,F=0.12 /
&RAMP ID='rampks',T=538.,F=0.12 /
&RAMP ID='rampcp',T= 93.,F=1.172 /
&RAMP ID='rampcp',T=205.,F=1.255 /
```

```
&RAMP ID='rampcp',T=316.,F=1.339 /
&RAMP ID='rampcp',T=425.,F=1.423 /
```

Notice that with temperature-dependent quantities, the RAMP parameter T now means Temperature, and F is the actual value of either C_P or KS. Neither C_P nor KS is given on the SURF line, but rather the RAMP names. One should prescribe either a constant value for C_P or KS or a RAMP name, but not both.

4.4.2 Combustion Parameters: The REAC Namelist Group

There are two ways of designating a fire: the first is to prescribe a Heat Release Rate Per Unit Area HRRPUA on a SURF line. The other is to prescribe a HEAT_OF_VAPORIZATION, in which case the burning rate of the fuel depends on the net heat feedback to the surface. In both cases, the mixture fraction combustion model is used. The REAC line is used for various parameters associated with the gas phase reaction of fuel and oxygen. Note that if only the fire's heat release rate is specified with HRRPUA, then these parameters usually do not require adjusting. However, if the burning behavior of a fuel is governed by its HEAT_OF_VAPORIZATION, care must be taken in selecting the appropriate reaction parameters. See Section 4.4.3 for additional details.

It is assumed that a single hydrocarbon fuel is being burned, with constant yields of CO and soot

$$\mathrm{C_xH_yO_z} + \nu_{\mathrm{O_2}}\ \mathrm{O_2} \rightarrow \nu_{\mathrm{CO_2}}\ \mathrm{CO_2} + \nu_{\mathrm{H_2O}}\ \mathrm{H_2O} + \nu_{\mathrm{CO}}\ \mathrm{CO} + \nu_{\mathrm{Soot}}\ \mathrm{Soot} \tag{4.1}$$

Specify the *ideal* stoichiometric coefficients for the fuel, O_2, CO_2 and H_2O, and yields for CO and soot. The yield of CO, for example, is defined as the fraction of the fuel mass that is converted into CO, and is denoted y_{CO}. Do not directly specify the stoichiometric coefficients for CO or soot because these numbers are usually reported in terms of mass yields. It is assumed that the CO and soot are created at the flame and transported with the combustion products. No growth, oxidation or after burning is assumed. Research continues to refine the handling of these products of incomplete combustion. For the moment, the ideal stoichiometric coefficients for O_2, CO_2 and H_2O are adjusted to account for the production of CO and soot at the start of a calculation. The actual stoichioimetric coefficients used in the calculations are

$$\begin{aligned}
\nu_{O_2} &= \left(\mathrm{x} - \frac{M_f}{2M_{CO}} y_{CO} - \frac{M_f}{M_C} y_s\right) + \frac{\mathrm{y}}{4} - \frac{\mathrm{z}}{2} \\
\nu_{CO_2} &= \mathrm{x} - \frac{M_f}{M_{CO}} y_{CO} - \frac{M_f}{M_C} y_s \\
\nu_{H_2O} &= \frac{\mathrm{y}}{2} \\
\nu_{CO} &= \frac{M_f}{M_{CO}} y_{CO} \\
\nu_{soot} &= \frac{M_f}{M_C} y_s
\end{aligned}$$

The following parameters may be prescribed on the REAC line. Note that only one REAC line may be used. Most often, one selects a reaction from the DATABASE via the REACTION parameter on the MISC line.

ID A character string naming the reaction. It is optional if the REAC line is included within the data input file, mandatory if the REAC line is included in the DATABASE file.

NU_O2, NU_H2O, NU_FUEL, NU_CO2 Ideal stoichiometric coefficients for the reaction of a hydrocarbon fuel. NU_FUEL is 1 by default. All numbers are positive. Default values are those of propane.

`MW_FUEL` Molecular weight of the fuel (g/mol). Default is for propane, 44 g/mol. If the fuel has nitrogen in it, add the parameter `FUEL_N2` to indicate how many molecules of N_2 there are. For example urethane is $C_3H_7NO_2$. For it, `FUEL_N2=0.5` because there is half of a N_2 molecule.

`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. (Default 0.01)

`CO_YIELD` The fraction of fuel mass converted into carbon monoxide, y_{CO}. Normally, this parameter need not be set, because by default y_{CO} is linked to the `SOOT_YIELD`, y_s, via the correlation developed by Köylü and Faeth

$$y_{CO} = \frac{12\,x}{M_f \nu_f}\; 0.0014 + 0.37\, y_s$$

where x is the number of carbon atoms in a fuel molecule, M_f is the molecular weight of the fuel, and ν_f is the stoichiometric coefficient of the fuel, assumed to be 1 here [5]. Note that this correlation refers to well-ventilated fires. Yields of CO and soot in underventilated fires is still a subject of active research.

`EPUMO2` Energy Per Unit Mass Oxygen, ΔH_{O_2} (kJ/kg). The amount of energy released per unit mass of oxygen consumed. (Default 13,100 kJ/kg). Note that the heat of combustion

$$\Delta H \approx \frac{\nu_{O_2}\, M_{O_2}}{\nu_f\, M_f}\;\; \Delta H_{O_2}$$

is modified slightly due to the presence of soot and CO.

`RADIATIVE_FRACTION` The fraction of energy released from the flame as thermal radiation. This parameter requires some interpretation. See Section 4.4.3 for details. (Default 0.35)

A few sample `REAC` lines are given here. More can be found in the `DATABASE` file.

```
&REAC ID='METHANE'
      MW_FUEL=16
      NU_O2=2.
      NU_CO2=1.
      NU_H2O=2.
      RADIATIVE_FRACTION=0.15
      SOOT_YIELD=0.01 /

&REAC ID='WOOD'
      SOOT_YIELD = 0.01
      NU_O2      = 3.7
      NU_CO2     = 3.4
      NU_H2O     = 3.1
      MW_FUEL    = 87.
      EPUMO2     = 8850. /
```

4.4.3 Important Issues Related to Combustion

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

Heat Release Rate: For coarsely gridded calculations, it is not possible to resolve the fire adequately, and as a result the heat release rate and flame height can be underestimated [6]. There are several ways to remedy the problem, one of which is to choose a different value of the mixture fraction when defining the flame sheet. The program has built into it a routine that redefines the stoichiometric value of the mixture fraction

$$\frac{Z_{f,eff}}{Z_f} = \min\left(1\,,\, C\,\frac{D^*}{\delta x}\right) \tag{4.2}$$

where Z_f is the ideal stoichiometric value of the mixture fraction, C is an empirical constant (not an input parameter), and D^* is the characteristic fire diameter

$$D^* = \left(\frac{\dot{Q}}{\rho_\infty\, c_p\, T_\infty\, \sqrt{g}}\right)^{\frac{2}{5}} \tag{4.3}$$

"Good" or "bad" resolution depends on both the size of the fire and the size of the grid cells. For relatively small values of D^* and/or large grid cells, the stoichiometric value of the mixture fraction is intentionally reduced, resulting in a better estimate of flame height and structure. In cases where D^* is relatively large and the cells relatively small, no adjustment need be made to the stoichiometric value of the mixture fraction. As always, a sufficiently resolved grid eliminates the need for these types of corrections, but in many instances it is difficult to both finely grid the region near the fire while at the same time saving enough cells for the rest of the space.

By default, the re-adjustment of the stoichiometric value of the mixture fraction is applied automatically. To turn it off, specify `AUTOMATIC_Z=.FALSE.` on the `MISC` line, in which case the ideal stoichiometric value of the mixture fraction is used to mark the flame zone. Also, minor adjustments can be made to the flame height by setting `Z_CONSTANT` to a value different than 1 on the `REAC` line. This parameter is essentially a normalized version of $1/C$ from Eq. (4.2). Reducing `Z_CONSTANT` shortens the flame length; increasing it lengthens the flame.

Radiative Fraction: The fraction of energy released from the fire as thermal radiation is a function of both the flame temperature and chemical composition, neither of which are well known in a large scale fire calculation. In calculations in which the grid cells are on the order of a centimeter and larger, the temperature near the flame surface cannot be relied upon when computing the source term in the radiation transport equation, especially because of the T^4 dependence. To compensate, if one prescribes a non-zero value of `RADIATIVE_FRACTION`, a grid cell cut by the flame radiates that fraction of the chemical energy being released into it. Some of that energy may be reabsorbed elsewhere, yielding a net radiative loss that is less than `RADIATIVE_FRACTION`, depending mainly on the size of the fire and the soot loading. If it is desired to use the radiation transport equation as is, then `RADIATIVE_FRACTION` ought to be set to zero, and the source term in the radiative transport equation is then based solely on the gas temperature and the chemical composition.

Heat of Combustion: The user does not usually prescribe the heat of combustion explicitly. Normally, the stoichiometric parameters listed on the `REAC` line are used to compute the heat of combustion. However, if a `HEAT_OF_VAPORIZATION` has been specified on a `SURF` line and the heat of combustion of the material differs from that specified by the governing `REACTION`, then add a `HEAT_OF_COMBUSTION` (kJ/kg) to the `SURF` line. In a mixture fraction based combustion model, it is assumed that there is only one fuel. However, in a realistic fire scenario, there may be many fuels originating from the various burning objects in the building. Specify the stoichiometry of the predominant reaction via the `REAC` namelist group. Often these parameters are stored in the `DATABASE` file, in which case simply prescribe a `REACTION` on the `MISC` line.

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.

Predicting the Burning Rate: Either prescribe the burning rate of the fuel via the parameter HRRPUA on the SURF line, or let the burning rate be predicted based on the fuel's thermal properties and its HEAT_OF_VAPORIZATION. This latter action introduces a fair amount of uncertainty into the calculation because the burning rate is a function of the energy fed back from the fire via convection and radiation, and the energy conducted through the solid or liquid fuel. The various phenomena are still subjects of active research, so be aware of the potential errors introduced into the calculation. One of the greatest sources of error is the radiative heat flux from the fire to the fuel surface. The error is due to a combination of insufficient grid resolution in the boundary layer, and uncertainty in the absorption coefficient and flame temperature. As a result, the heat flux to the fuel surface is often over-predicted. Until improvement can be made in the radiation calculation, prevent excess pyrolysis of fuel by prescribing a BURNING_RATE_MAX (kg/m^2/s) on the SURF line that limits the burning rate of the fuel to its measured maximum.

Gas Phase Fire Suppression: Modeling suppression of a fire due to the introduction of a suppression agent like CO_2 or water mist, or due to the exhaustion of oxygen within a compartment is challenging because the relevant physical mechanisms occur at length scales smaller than a single grid cell. Flames are extinguished due to lowered temperatures and dilution of the oxygen supply. With the mixture fraction model, only the dilution effect is accounted for because it is assumed that fuel and oxygen burn regardless of the temperature. However, a simple suppression algorithm has been implemented in FDS that attempts to guage whether or not a flame is viable near the stoichiometric mixture fraction surface. The Technical Reference Guide [1] contains more details about how the mechanism works. The only parameters the user 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 and 1427 °C, respectively. To turn off the suppression algorithm, set SUPPRESSION=.FALSE. on the MISC line.

4.4.4 Creating Obstructions: The OBST Namelist Group

OBST is the namelist group listing information about obstructions. Each OBST line contains the coordinates of a rectangular solid within the flow domain. This solid is defined by two points (x_1,y_1,z_1) and (x_2,y_2,z_2) that are entered on the OBST line in terms of the sextuplet XB=X1,X2,Y1,Y2,Z1,Z2. In addition to the coordinates, the boundary conditions for the obstruction can be specified with the parameter SURF_ID, which designates which SURF group to apply at the surface of the obstruction. If the obstruction has different boundary conditions for its top, sides and bottom, do not prescribe only one boundary condition with SURF_ID. Instead, use SURF_IDS, an array of three character strings specifying the boundary condition IDs for the top, sides and bottom of the obstruction, respectively. If the default boundary condition is desired, then SURF_ID(S) need not be set. However, if at least one of the surface conditions for an obstruction is the inert default, it can be referred to as 'INERT'. For example:

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

puts a fire on top of the obstruction. This is a simple way of prescribing a burner.
Some additional features of obstructions are as follows:

- In addition to SURF_ID and SURF_IDS, one 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 grid is coarse relative to the thickness of the barrier, the obstruction might be unnecessarily bulky if it is assumed to be one layer of grid cells thick. To remedy this, the logic of FDS has been changed so that all faces of an obstruction are shifted to the closest grid 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 thin or thick boundary conditions. This feature is still 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 grid cell thick. Thin sheet obstructions work fine as flow barriers, but other features are fragile and should be used with caution. Also note that unlike earlier versions of FDS, now obstructions that are too small relative to the underlying numerical grid are rejected. Be careful when testing cases on coarse grids.

- Obstructions may be created or removed during a simulation. See Section 5.4 for details.

- 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.

4.4.5 Creating Voids: The HOLE Namelist Group

It is often convenient to carve a hole out of an existing obstruction or set of obstructions. To do this, add lines of the form

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

Any solid grid 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. For example, if the OBST line denotes a wall 0.1 m thick:

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

and one wants to create a door, add this:

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

The extra centimeter added to the x coordinates of the hole make it clear that the hole is to punch through the entire obstruction. Students of American politics appreciate that it is not wise to leave hanging chads.

When a HOLE is created, the affected obstruction(s) are either rejected, or created or removed at predetermined times. For example, adding the parameter T_CREATE to the HOLE line creates the hole after the given number of seconds. Likewise, T_REMOVE re-fills the hole after the given number of seconds. Notice

that T_CREATE on a HOLE line has the opposite effect as T_CREATE on an OBST line. The same applies for T_REMOVE.

If the hole is to be created or removed in response to a HEAT detector, use the parameters HEAT_CREATE or HEAT_REMOVE. Each is a character string naming the appropriate HEAT detector. For example:

```
&HOLE XB=2.0,4.5,1.9,4.8,0.0,9.2,HEAT_CREATE='heat2' /
&HEAT XYZ=3.2,4.4,6.8,ACTIVATION_TEMPERATURE=74.,RTI=100.,LABEL='heat2' /
```

creates a hole when heat2 activates. Note that the old convention of using a negative value for T_CREATE or T_REMOVE does not work here. Also, if the HOLE line contains HEAT_CREATE='ALL', then any HEAT detector creates the hole.

If it is desired that the obstruction(s) to be cut out should have a different color than the original obstruction, set the real triplet RGB on the HOLE line. 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 (TWFIN=0 on the TIME line) or a short-time job to test the operation of T_CREATE and/or T_REMOVE.

Note that a HOLE cannot be used on a VENT or a mesh boundary.

4.4.6 Designating Vents and Surfaces: The VENT Namelist Group

Whereas the OBST group is used to prescribe obstructions within the computational domain, the VENT group is used to prescribe planes adjacent to obstructions or external walls. The vents are chosen in a similar manner to the obstructions, with the sextuplet XB denoting a plane abutting a solid surface. Two of the six coordinates must be the same, denoting a plane as opposed to a solid. An easy way to specify an entire external wall is to replace XB with CB, a character string whose value is one of the following: 'XBAR', 'XBAR0', 'YBAR', 'YBAR0', 'ZBAR' or 'ZBAR0' denoting the planes $x =$ XBAR, $x =$ XBAR0, $y =$ YBAR, $y =$ YBAR0, $z =$ ZBAR or $z =$ ZBAR0, 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 CB shortcut when doing a multiple mesh simulation, that is, when more than one rectangular mesh is used. The plane designated by the keyword CB is applied to all of the grids, 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 4.4.7.

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 pointing to a SURF line with the characteristics of the thermal and combustion properties of the fuel.

There are two reserved SURF_ID's that may be applied to a VENT. The first is 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. It is assumed here that the exterior boundary of the computational domain is a solid wall, and the OPEN vent is essentially an open door or window. The second reserved SURF_ID is for a symmetry plane, in which case the VENT has the attribute SURF_ID='MIRROR'. 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 "CB" notation is handy.

Vents to the outside of the computational domain (OPEN vents) may be opened or closed during a simulation. See Section 5.4 for details.

There is one exception to the rule that VENTs ought to be prescribed flush against a solid obstruction or external boundary. A VENT that is prescribed in the interior of the domain without any solid surface flush up against it can act as a fan. For example, the lines

```
&PDIM XBAR=1.0,YBAR=1.0,ZBAR=1.0 /
                .
                .
&SURF ID='BLOW',VEL=2.1,TAU_V=5.0 /
&VENT XB=0.50,0.50,0.25,0.75,0.25,0.75,SURF_ID='BLOW' /
```

create a plane within the unit cube domain that blows air at 2.1 m/s in the positive *x* direction, ramping up in about 5 s. In general, the value of VEL associated with a VENT that is not aligned with a solid surface blows gases in the coordinate direction normal to the VENT – positive if VEL is positive, negative if VEL is negative. Note that only velocity boundary conditions are appropriate on a SURF line associated with a free-standing VENT because there is no solid obstruction on which to impose thermal or material boundary conditions.

One final note for VENT: 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.

4.4.7 Coloring Obstructions, Vents and Surfaces

Surface and obstruction colors are prescribed in a number of ways. If one desires that all surfaces associated with a given SURF line be colored the same way, prescribe a triplet of real numbers called RGB on the SURF line. The three numbers may be between 0 and 1, indicating the amount of Red, Green and Blue that make up the color. Table 4.1 lists the RGB indices for a variety of colors. 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,

```
&SURF ID='UPHOLSTERY',...,RGB=0.0,1.0,0.0 /
```

causes the furnishings to be colored green in Smokeview. Fractions of "R", "G" and "B" can be used to create more life-like colors. 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 = 'RED', 'BLUE', 'BLACK', 'YELLOW', 'GREEN', 'MAGENTA', 'WHITE', 'CYAN' 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 triplet RGB on an OBST or VENT line to gain a wider color pallette. The use of RGB is preferable, especially to create colors that do not clash with the pastel colors used to show temperatures, concentrations, *etc.*

Table 4.1: RGB values for various colors.

Color	Red	Green	Blue
Red	1.0	0.0	0.0
Green	0.0	1.0	0.0
Blue	0.0	0.0	1.0
White	1.0	1.0	1.0
Black	0.0	0.0	0.0
Gray	0.5	0.5	0.5
Magenta	1.0	0.0	1.0
Yellow	1.0	1.0	0.0
Cyan	0.0	1.0	1.0
Tan	0.8	0.6	0.4
Orange	1.0	0.5	0.0

4.5 Lagrangian Particles: The PART Namelist Group

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

Properties of different types of Lagrangian particles are designated via one or more PART lines. Much like SURF lines contain the properties of a solid surface or vent, PART lines contain information about particles and droplets. Once a particular type of particle or droplet has been described using a PART line, then the name of that particle or droplet is invoked elsewhere in the input file. For example, an input file may have several PART lines that include the properties of different types of Lagrangian particles:

```
&PART ID='smoke',... /
&PART ID='water',... /
```

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

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

or the PART type can be invoked from a SPRK line to change the properties of the droplets ejected by a sprinkler, for example

```
&SPRK PART_ID='water',... /
```

The frequency at which Lagrangian particles are added to the simulation is controlled by the parameters DTPAR and DTSPAR which are prescribed on the MISC line. Regardless of type, particles at solid surfaces are inserted every DTPAR seconds and droplets from sprinklers every DTSPAR seconds. It is not possible to have different particle types inserted more often than others.

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.,... /
```

Other options for massless particles are

SAMPLING_FACTOR Sampling factor for the particle output file **CHID.part**. This parameter can be used to reduce the size of the particle output file used to animate the simulation. (Default 1)

QUANTITY A character string indicating which scalar quantity should be used to color the particles when viewed as an animation. The choices are 'TEMPERATURE' (°C), 'DIAMETER' (μm), or 'VELOCITY' (m/s). If the particles are to be one color, specify QUANTITY='RED', 'BLUE', 'BLACK', 'YELLOW', 'GREEN', 'MAGENTA', 'WHITE' or 'CYAN'. Note that if the particles are MASSLESS, it is not appropriate to color them according to their 'DIAMETER' or 'TEMPERATURE'. Unlike early versions of FDS, particles are no longer colored by gas phase quantities, but rather by properties of the particle itself. For example, 'TEMPERATURE' refers to the temperature of the particle itself (assuming it has mass) rather than the local gas temperature.

AGE Number of seconds the particle or droplet exists. Useful parameter to use when trying to reduce the number of droplets or particles in a simulation.

For Lagrangian particles that are not MASSLESS, the following additional parameters can be included on the PART line. Note that many of these parameters are used only for certain types of particles or droplets and need to be specified only if they are relevant.

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

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

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

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

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

`DIAMETER` Median volumetric diameter of droplets/particles, with the distribution assumed to be a combination of Rosin-Rammler and log-normal (Default 100 μm). The width of the distribution is controlled by the parameter `GAMMA_D`, default 2.4. See discussion of sprinkler data files in Section 4.6.1 for details. (Default 100 μm)

`STATIC` Logical parameter indicating whether particles move or just serve as obstructions or clutter. This parameter should only be used when `NUMBER_INITIAL_DROPLETS` is greater than 0. (Default `.FALSE.`)

`FUEL` Logical parameter indicating whether the liquid droplets evaporate into fuel gas and burn. (Default `.FALSE.`). If `FUEL=.TRUE.`, then also add `HEAT_OF_COMBUSTION` (kJ/kg) of the fuel.

`WATER` Logical parameter indicating whether the liquid droplets evaporate into `WATER VAPOR`, which is a separate species that is automatically added to the calculation. (Default `.FALSE.`)

`NUMBER_INITIAL_DROPLETS` Seed the domain with particles/droplets at the start of the simulation. (Default 0). 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`. 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.

`DROPLETS_PER_SECOND` Number of sprinkler droplets inserted every second per active sprinkler. Note that this parameter only affects sprinkler droplets. Changing this parameter does *not* change the flow rate, but rather the number of droplets used to represent the flow. (Default 1000)

4.6 Sprinklers and Detectors

Research to better characterize a sprinkler's spray distribution and droplet size has led to a need for a file format to record everything about a particular sprinkler in one file. Much of this is transparent to the user who simply specifies the physical coordinates of the sprinkler in the data file **job_name.data**. The data for the particular sprinkler is kept in a file called **sprinkler_name.spk**. Heat detector parameters are given within the input file itself. All one needs to do is specify the physical coordinates and an RTI within the data file **job_name.data**.

4.6.1 Specifying Sprinklers: The SPRK Namelist Group

Information about a given sprinkler is contained in a separate file, called, for example, **sprinkler_name.spk**[3]. A sample file is shown in Fig. 4.2. All the sprinkler characteristics are stored in this file. The user only has to provide lines of the form

```
&SPRK XYZ=3.0,5.6,2.3,MAKE='sprinkler_make',LABEL='doorway' /
```

in the input data file **job_name.data**. The physical coordinates of the sprinkler are given by a triplet of real numbers XYZ. The make of the sprinkler is given by a character string MAKE, for which there exists a file called **sprinkler_make.spk** containing thermal properties and other information about the sprinkler. The character string LABEL is merely a descriptor to identify the sprinkler in various output files.

There is an option to manually activate the sprinkler by setting T_ACTIVATE in seconds on the SPRK line itself. The sprinkler can be shut off by setting T_DEACTIVATE. Also, the (normally) downward-directed sprinkler spray can be redirected in some other direction. The triplet ORIENTATION can be added to a given SPRK line to change the direction of the spray. The default value of ORIENTATION is (0,0,-1), but if one were to prescribe

```
&SPRK XYZ=3.0,5.6,2.3,MAKE='sprinkler_make',ORIENTATION=1,0,0 /
```

the sprinkler would point in the positive x direction. If the designated sprinkler has a spray which is not axially-symmetric, another parameter called ROTATION can be used to rotate the spray through a given number of degrees. The default value is 0, but experiment with a single sprinkler flow calculation to ensure that the sprinkler is being re-oriented as desired.

In addition to the sprinkler parameters listed on the SPRK line, one may also want to adjust some parameters that control the number and frequency of droplet insertions. These parameters are typically entered under the namelist group PART. See Section 4.5 for details.

If the sprinkler system is of the dry-pipe variety, prescribe a time delay for the system by adding a DELAY (s) to the PIPE namelist line. The PIPE namelist group contains information about the sprinkler system as a whole. For the moment, it contains only two parameters, DELAY and PRESSURE. Note that only one PIPE line is read by FDS. If DELAY is set to be non-zero, then that amount of time has to elapse before water is allowed to flow from any activated sprinkler.

Be aware that sprinklers produce many droplets that need to be tracked in the calculation. To limit the burden, sprinkler droplets are given an AGE of 60 s, and also sprinkler droplets disappear when they hit the lower boundary of the computational domain, regardless of whether it is solid or not. To allow droplets to live longer than AGE, add PART_ID='whatever' to the SPRK line, where 'whatever' is the ID of a PART line which can be used to modify the properties of the water droplets. For example,

```
&SPRK XYZ=3.0,5.6,2.3,MAKE='sprinkler_make',PART_ID='my water droplets' /
&PART ID='my water droplets',WATER=.TRUE.,AGE=300.,QUANTITY='DIAMETER' /
```

[3]The MISC character string DATABASE_DIRECTORY can be used to point to a directory where sprinkler files are stored.

allows control over some of the properties of the water droplets from a sprinkler. The phrase `WATER=.TRUE.` is important to direct the program to evaporate the droplets into `WATER_VAPOR` without requiring one to explicitly add `WATER_VAPOR` to the calculation. The phrase `QUANTITY='DIAMETER'` directs Smokeview to color the water droplets according to their diameter.

To stop FDS from removing sprinkler droplets from the lower boundary of the computational domain, add the phrase `POROUS_FLOOR=.FALSE.` to the `MISC` line. Be aware, however, that droplets that land on the floor continue to move horizontally in randomly selected directions; bouncing off obstructions, and consuming CPU time.

Sprinkler properties listed in the file **sprinkler_name.spk** are

`RTI` Response Time Index of the sprinkler in units of $\sqrt{\mathrm{m \cdot s}}$. (Default 165.)

`C-FACTOR` C-Factor of sprinkler in units of $\sqrt{\mathrm{m/s}}$. (Default 0.)

`K-FACTOR` K-Factor of sprinkler in units of L/min/bar$^{\frac{1}{2}}$. (Default 166) The flow rate is given by $\dot{m}_w = K\sqrt{p}$ where $\dot{m}_w$ is the flow rate in L/min, K the K-factor in L/min/(bar)$^{\frac{1}{2}}$ and p the gauge pressure in bar[4]

`ACTIVATION_TEMPERATURE` Link activation temperature (°C). (Default 74 °C)

`OPERATING_PRESSURE` Sprinkler operating pressure in units of bar. This is the pressure at which the sprinkler was tested. If one desires to flow the sprinkler at a different pressure than that at which it was tested, a single line

```
&PIPE PRESSURE=1.5 /
```

should be added to the input data file, and then all sprinklers operate at this new pressure, 1.5 bar for example. The reason for the two pressures is that a slight adjustment is made to the droplet size distribution to account for the fact that the sprinkler operates at a different pressure than that at which it was tested. In the absence of any user-specified `PRESSURE`, the `OPERATING_PRESSURE` listed in the **.spk** file is used.

`OFFSET_DISTANCE` 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_DISTANCE` the droplets have completely broken up and are transported independently of each other. (Default 0.10 m)

`VELOCITY` Description of the initial droplet velocity distribution. There are two options of inputs here, designated by either a 1 or a 2 on the line immediately following the keyword `VELOCITY`. For case 1, the parameters: Minimum Spray Angle, Maximum Spray Angle, and Speed should be given values. The angles outline a conical spray pattern relative to the south pole of the sphere centered at the sprinkler with radius `OFFSET_DISTANCE`. For example, a Minimum Spray Angle of 20° and a Maximum Spray Angle of 80° directs the water droplets to leave the sprinkler through a conical region 20° north of the south pole and 10° south of the equator. The droplets are uniformly distributed within this belt. If more detailed information about the sprinkler spray is known, then the keyword `VELOCITY` is followed by a 2, indicating that the normal component of velocity of the droplets is given as a function of n_a azimuthal (latitudinal) points starting at the bottom of the sphere, and n_l longitudinal points starting at one of the frame arms. The integers n_a and n_l should be listed on the line following the "2", as in the example (Fig. 4.2). The case 2 `VELOCITY` data is applicable on a sphere centered about the sprinkler whose radius is the `OFFSET_DISTANCE`. This is typically the distance from the sprinkler itself where all the spray characteristic measurements have been made.

[4] 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$^{\frac{1}{2}}$.

`FLUX` Relative water mass flux (kg/m^2/s), entered in the same way as the case 2 for `VELOCITY`. If case 1 for `VELOCITY` is used, there is no need to even specify a `FLUX`.

`SIZE_DISTRIBUTION` Information about the droplet size distribution. The median volumetric droplet diameter can either be a global value, or it can be prescribed as a function of the solid angle. If the median diameter is independent of position, the key word `SIZE_DISTRIBUTION` is to be followed by the number 1 on the next line to indicate that the information given is a global average. The next line after that has the median volumetric diameter (800 μm in the sample case), plus the parameter γ from the Rosin-Rammler/log-normal distribution

$$F(d) = \begin{cases} \frac{1}{\sqrt{2\pi}} \int_0^d \frac{1}{\sigma d'} e^{-\frac{[\ln(d'/d_m)]^2}{2\sigma^2}} \, dd' & (d \le d_m) \\ 1 - e^{-0.693\left(\frac{d}{d_m}\right)^\gamma} & (d_m < d) \end{cases} \tag{4.4}$$

Note that the parameter σ is given the value $\sigma = 2/(\sqrt{2\pi}(\ln 2)\,\gamma) = 1.15/\gamma$ which ensures that the two functions are smoothly joined at $d = d_m$. The larger the value of γ, the narrower the droplet size is distributed about the median value. Its default value is 2.4.

If one has information about the median droplet size as a function of the sprinkler solid angle, that information can be entered as in the following example:

```
SIZE_DISTRIBUTION
2
4 1 2.4
700
600
500
400
```

Here, the number 2 is the code number directing FDS to read the next few lines in a certain way. The 4 indicates that there are 4 values of the median diameter from the south pole to the north pole of the sphere surrounding the sprinkler. For each latitudinal zone, there is only one value, meaning that this value is independent of longitude. The value of 700 μm is for the south pole to 45° S, 600 μm is for 45° S to the equator, 500 μm is for the equator to 45° N, and 400 μm is for 45° N to the north pole. The number 2.4 following the number of latitudinal and longitudinal points is the value of γ. Note that γ is not a function of position. Only the median diameter can be a function of position.

For more information about sprinklers, see the Technical Reference Guide [1].

4.6.2 Specifying Heat Detectors: The `HEAT` Namelist Group

`HEAT` is the namelist group used to specify heat detectors. A heat detector is designated in the data input file with a line(s) of the form

```
&HEAT XYZ=3.0,5.6,2.3,RTI=132.,ACTIVATION_TEMPERATURE=74.,LABEL='door' /
```

Like a sprinkler, the physical coordinates of the heat detector are given by a triplet of real numbers `XYZ`. Unlike a sprinkler, there is no external file with thermal properties for a heat detector. `RTI` is the Response Time Index in units of $\sqrt{\mathrm{m \cdot s}}$. `ACTIVATION_TEMPERATURE` is the link activation temperature in degrees C (Default 74 °C). `LABEL` is just a descriptor.

Note that a heat detector can be used to trigger an event to happen, like the opening of a `VENT` to the outside or the removal of an `OBST`struction. See Section 5.4 for details.

```
MANUFACTURER
Acme
MODEL
Splash2000
OPERATING_PRESSURE
0.50
K-FACTOR
80.
RTI
110.
C-FACTOR
0.
ACTIVATION_TEMPERATURE
74.
OFFSET_DISTANCE
0.10
SIZE_DISTRIBUTION
1
800 2.4
VELOCITY
2
60,24
6.3,6.4,6.5, ... (24 longitudinal zones, like time zones in this case)
     .
     .
     .
(60 azimuthal zones, south pole to north pole, 3 degrees per zone)
FLUX
2
60,24
11.9,13.0,13.5, ...
     .
     .
     .
```

Figure 4.2: **A sample sprinkler file. Note that the keywords should be written exactly as they are written here, and they should start in the very left column of each line. Note that this sample file includes a fairly detailed description of the spray pattern, requiring extensive measurements. In most cases, the** `VELOCITY` **and** `FLUX` **can be described with just a few parameters.**

4.7 Output Files

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 `THCP` is used to specify "thermocouple" or point measurements.

An animated isosurface (`ISOF`) is a three dimensional contour of a scalar quantity. For example, a 300 °C temperature isosurface shows where the gas temperature is 300 °C. By default, the stoichiometric value of the mixture fraction is visualized via an isosurface. Choose several other quantities to visualize via an isosurface.

In order to visualize the flow patterns better, one can 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 one to animate these quantities in time.

For static pictures of the flow field, use the `PL3D` (PLot3D file) namelist group. Plot3D format is used by many CFD programs as a simple way to store specified quantities over the entire grid 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.

The time increment between data dumps can be controlled. For `THCP`, `SLCF`, `ISOF` and `PART` files, data is written out to files every `TWFIN`/`NFRAMES` seconds. For `BNDF` files, data is written out every `TWFIN`/`NFRAMES`/2 seconds. The parameter `NFRAMES` may be specified on the `MISC` line. For the `PL3D` format, data is written out every `TWFIN`/5 seconds. To change the sampling rate of any output file type, the parameter `DTSAM` should be set on any one line of that particular group. For example, if one desires that `THCP` data be written out every second, then the string `DTSAM=1.` should be written on one of the `THCP` lines. This one entry dictates that all `THCP` data be written out every second. All of the other data files are updated according to the default increment, or according to another prescription of `DTSAM`.

4.7.1 Point Measurements: The `THCP` Namelist Group

`THCP` is the name of a group of parameters that can be used to record values of various quantities at a point as a function of time, much like a thermocouple or other point measurement. Each `THCP` line consists of the coordinates of the point at which the measurement is to be recorded, `XYZ`, and a quantity to record, `QUANTITY`. The quantities are either *gas phase* (see Table 4.2) or *solid phase* (see Table 4.3). Many of the gas phase quantities are self-explanatory. However, some require explanation. For example, `THERMOCOUPLE` is the temperature of the thermocouple itself, usually close to the gas temperature, but not always. It is determined by solving the following equation for T_{TC} iteratively [7]

$$\varepsilon_{TC}(\sigma T_{TC}^4 - U/4) + h(T_{TC} - T_g) = 0 \tag{4.5}$$

where ε_{TC} is the emissivity of the thermocouple, U is the integrated radiative intensity, T_g is the true gas temperature, and h is the heat transfer coefficient to a small sphere, $h = k_a \mathrm{Nu}/\mathrm{Pr}/d_{TC}$. 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.

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 grid does not always align with the input obstruction locations.

To help locate the appropriate surface, the parameter `IOR` *must* be included when designating a solid phase quantity. If the orientation of the solid surface is in the positive x direction `IOR=1`, negative x direction `IOR=-1`, positive y `IOR=2`, negative `IOR=-2`, positive z `IOR=3`, and negative z `IOR=-3`. There are still instances where FDS cannot determine which solid surface is being designated, in which case an error message appears in the diagnostic output file. Re-position the probe and try again. For example, the line

```
&THCP XYZ=0.7,0.9,2.1,QUANTITY='WALL_TEMPERATURE',IOR=-2,LABEL='whatever' /
```

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

In addition to point measurements, the `THCP` group can be used to report integrated quantities (See Table 4.4). For example, one often wants to know the mass flow out of a door or window. To report this, add the line

```
&THCP XB=0.3,0.5,2.1,2.5,3.0,3.0,QUANTITY='MASS FLOW',LABEL='whatever' /
```

Note that in this case, a plane is specified rather than a point. The sextuplet `XB` is used for this purpose. Notice when a flow is desired, two of the six coordinates need to be the same. Another `QUANTITY`, `HRR`, can be used to compute the total heat release rate within a subset of the domain. In this case, the sextuplet `XB` ought to define a volume rather than a plane. Specification of the plane or volume over which the integration is to take place can only be done using `XB` – avoid planes or volumes that cross multiple mesh boundaries. FDS has to decide which mesh to use in the integration, and it chooses the finest mesh overlapping the centroid of the designated plane or volume.

The thermocouples can be labeled with the character string `LABEL` for easier identification in the output file. The output file is a comma-delimited ASCII file called **CHID_tc.csv** which can be imported into most spread sheet software packages.

4.7.2 Animated Planar Slices: The `SLCF` Namelist Group

The `SLCF` ("slice file") namelist group allows one to record various gas phase quantities (Table 4.2) 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 region for which values of `QUANTITY` are to be recorded every `DTSAM` s. `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.

Slice file information is recorded in files labeled **CHID_*n*.sf**, where n is the index of the slice file. A Fortran 90 routine **fds2ascii** produces a test file from a line, plane or volume of data. See Section 4.7.6 for more details.

Figure 4.3 shows several snapshots of a vertical animated slice from Smokeview where the slice is colored according to gas temperature. Slice files are displayed by selecting the desired file from the `Load/Unload` menu in Smokeview.

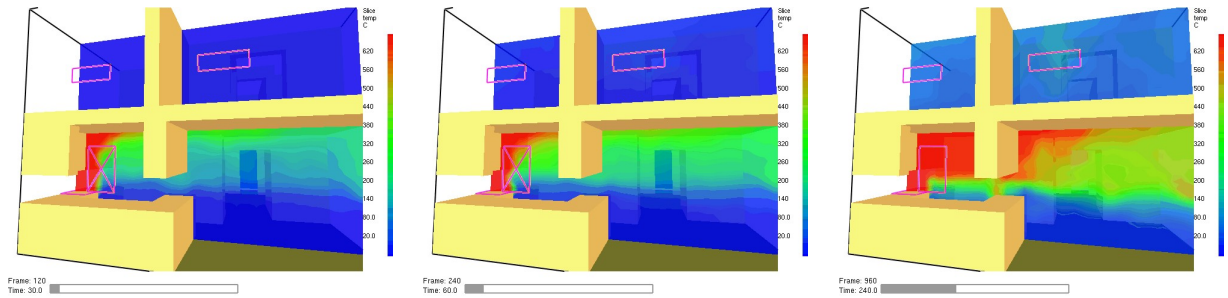
Animated vectors are displayed using data contained in two or more slice files. The direction and length of the vectors are determined from the U, V and/or W velocity slice files. The vector colors are determined from the file (such as temperature) selected from the `Load/Unload` menu. Figure 4.4 shows a sequence of vector slices corresponding to the shaded temperature contours found in Figure 4.3.

4.7.3 Animated Boundary Quantities: The `BNDF` Namelist Group

The `BNDF` ("boundary file") namelist group allows one to record surface quantities at all solid obstructions. The possible output quantities are listed in Table 4.3. As with the `SLCF` group, each quantity is prescribed

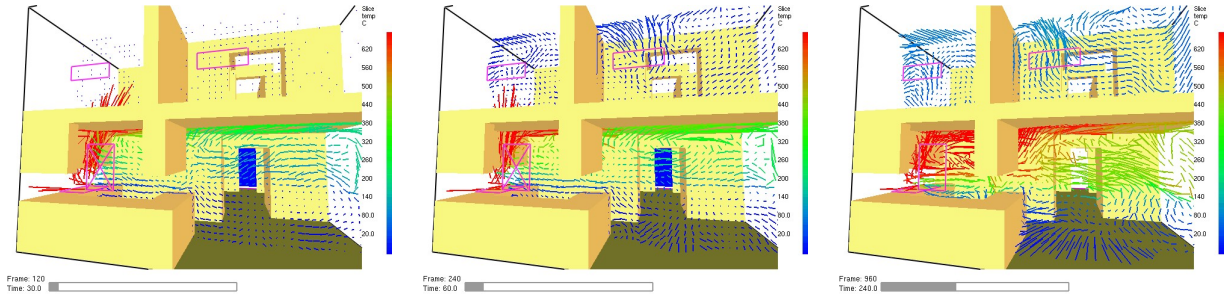
Quantity	Description	Symbol	Units
DENSITY	density	ρ	kg/m^3
TEMPERATURE	gas temperature	T	°C
THERMOCOUPLE	thermocouple temperature	T_{TC}	°C
U-VELOCITY	velocity component	u	m/s
V-VELOCITY	velocity component	v	m/s
W-VELOCITY	velocity component	w	m/s
VELOCITY	flow speed	$\sqrt{u^2+v^2+w^2}$	m/s
PRESSURE	perturbation pressure	$\tilde{p}$	Pa
H	total pressure divided by density	$H = \lvert\mathbf{u}\rvert^2/2+\tilde{p}/\rho$	(m/s)^2
HRRPUV	HRR Per Unit Volume	$\dot{q}'''$	kW/m^3
MIXTURE_FRACTION	mixture fraction	Z	kg/kg
DYNAMIC_VISCOSITY	dynamic viscosity	μ	kg/m/s
KINEMATIC_VISCOSITY	kinematic viscosity	$\nu = \mu/\rho$	m^2/s
DIVERGENCE	divergence	$\nabla\cdot\mathbf{u}$	s^{-1}
WMPUV	Water Mass PUV	m_w'''	kg/m^3
WATER VAPOR	water vapor mass frac.	Y_w	kg/kg
oxygen	O_2 volume fraction	$X_{O_2}(Z)$	mol/mol
oxygen mass fraction	O_2 mass fraction	$Y_{O_2}(Z)$	kg/kg
fuel	fuel volume fraction	$X_F(Z)$	mol/mol
nitrogen	N_2 volume fraction	$X_{N_2}(Z)$	mol/mol
water vapor	H_2O volume fraction	$X_{H_2O}(Z)$	mol/mol
carbon dioxide	CO_2 volume fraction	$X_{CO_2}(Z)$	mol/mol
carbon monoxide	CO volume fraction	$X_{CO}(Z)$	ppm
soot volume fraction	soot volume fraction	$\rho Y_s/\rho_s$	ppm
soot density	smoke particulate concentration	ρY_s	mg/m^3
extinction coefficient	light extinction coef.	$K = K_m\rho Y_s$	1/m
visibility	visibility distance	$S = C/K$	m
DROPLET_FLUX_X	water mass flux in x direction	$\dot{m}_w''$ (SLCF only)	kg/m^2/s
DROPLET_FLUX_Y	water mass flux in y direction	$\dot{m}_w''$ (SLCF only)	kg/m^2/s
DROPLET_FLUX_Z	water mass flux in z direction	$\dot{m}_w''$ (SLCF only)	kg/m^2/s

Table 4.2: **Gas Phase Output Quantities for THCP, SLCF or PL3D. Note that lower case quantities are appropriate only for calculations involving the mixture fraction. If individual species are listed via SPEC namelist lines, the Quantity for mass and volume fractions are [SPECIES_ID] and [SPECIES_ID]_VF. The quantities water vapor and WATER VAPOR denote the volume fraction of water vapor generated by combustion and the *mass* fraction of water vapor from evaporated sprinkler droplets, respectively. See Section 5.10 for a discussion of extinction coefficient and visibility.**



30 s 60 s 240 s

Figure 4.3: Slice file snapshots of shaded temperature contours in a vertical plane centered at the fire origin at t=(30, 60, 240) s after ignition. The data file for these contours was generated by adding the line
`&SLCF PBY=1.5,QUANTITY='TEMPERATURE' /`
to an FDS input file.



30 s 60 s 240 s

Figure 4.4: Vector slice file snapshots of shaded vector plots in a vertical plane centered at the fire origin at t=(30, 60, 240) s after ignition. The data files for these vector plots were generated by adding the line
`&SLCF PBY=1.5,QUANTITY='TEMPERATURE',VECTOR=.TRUE. /`
to an FDS input file.

Quantity	Description	Symbol	Units
RADIATIVE_FLUX	net radiative flux	$\dot{q}''_r$	kW/m^2
CONVECTIVE_FLUX	convective flux into solid	$\dot{q}''_c = h(T_g - T_w)$	kW/m^2
HEAT_FLUX	net heat flux into solid	$\dot{q}''_r + \dot{q}''_c$	kW/m^2
GAUGE_HEAT_FLUX	equivalent flux to cold wall	$\dot{q}''_r/\varepsilon + \dot{q}''_c + h(T_w - T_\infty) + \sigma(T_w^4 - T_\infty^4)$	kW/m^2
INCIDENT_HEAT_FLUX	incident heat flux	$\dot{q}''_r/\varepsilon + \dot{q}''_c + \sigma T_w^4$	kW/m^2
WALL_TEMPERATURE	wall temperature	T_w	C
INSIDE_WALL_TEMPERATURE	inner wall temperature	$T_s(x)$	C
BURNING_RATE	mass loss rate per unit area	$\dot{m}''_f$	kg/m^2/s
PRESSURE_COEFFICIENT	pressure coefficient	$C_p = (p - p_0)/(\frac{1}{2}\rho_\infty U^2)$	N/A
WMPUA	Water Mass Per Unit Area	$\dot{m}''_w$	kg/m^2
WCPUA	Water Cooling Per Unit Area	$\dot{q}''_w$	kW/m^2

Table 4.3: **Solid Phase Output Quantities for THCP and BNDF. Notes: (1) the difference between HEAT_FLUX and GAUGE_HEAT_FLUX. 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. It is more appropriate to use GAUGE_HEAT_FLUX when comparing predictions with experimental measurements. If the heat flux gauge used in an experiment has a temperature other than ambient, set GAUGE_TEMPERATURE on the MISC line. (2) If PRESSURE_COEFFICIENT is prescribed, one also needs to prescribe U via the parameter CHARACTERISTIC_VELOCITY on the MISC line. (3) To get an inner wall temperature, INSIDE_WALL_TEMPERATURE, the parameter DEPTH (in meters) must be provided on the same THCP line. Note that INSIDE_WALL_TEMPERATURE is allowed only as a THCP, not a BNDF, QUANTITY.**

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 and DTSAM (if desired).

Note that BNDF files can become very large, so caution should be used 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_BLOCK=.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_BLOCK=.TRUE. on the appropriate OBST line.

Figure 4.5 shows several snapshots of a boundary file animation where the surfaces are colored according to their temperature. Boundary files are displayed by selecting the desired file from the Smokeview Load/Unload menu.

4.7.4 Animated Isosurfaces: The ISOF Namelist Group

The ISOF ("ISOsurface File") namelist group allows one to save one or more values of a single gas phase quantity and render them as an animated sequence. The allowable quantities are DENSITY, TEMPERATURE, HRRPUV and MIXTURE_FRACTION. By default, the stoichiometric value of the mixture fraction is saved. For example, 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.

Figure 4.6 shows snapshots from an isosurface file animation at several time steps for a fire (mixture fraction level ≈ 0.05) and a smoke layer interface (mixture fraction level ≈ 0.001). These values are just for

Quantity	Description	Symbol	Units
VOLUME FLOW	volume flow rate	$\int \mathbf{u} \cdot d\mathbf{S}$	m^3/s
MASS FLOW	mass flow rate	$\int \rho \mathbf{u} \cdot d\mathbf{S}$	kg/s
HEAT FLOW	convective heat flow rate	$\int c_p \rho (T - T_\infty) \mathbf{u} \cdot d\mathbf{S}$	kW
HRR	heat release rate	$\int \dot{q}''' \, dV$	kW

Table 4.4: **Integrated Output Quantities for THCP only. These quantities differ from point measurements. See Section 4.7.1 for details.**

example. Smoke layers can be visualized with various values of the mixture fraction variable. Isosurface files are displayed by selecting the desired file from the Smokeview Load/Unload menu.

4.7.5 Static Data Dumps: The PL3D Namelist Group

PL3D is the namelist group that defines how often and what quantities are to be output into files of Plot3D format. At most one PL3D line should be listed in the input file. Five quantities from Table 4.2 are written out to a file at one instant in time every DTSAM s. The default specification is

```
&PL3D DTSAM=TWFIN/5,QUANTITIES='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 parameter QUANTITIES(1) and QUANTITIES(5).

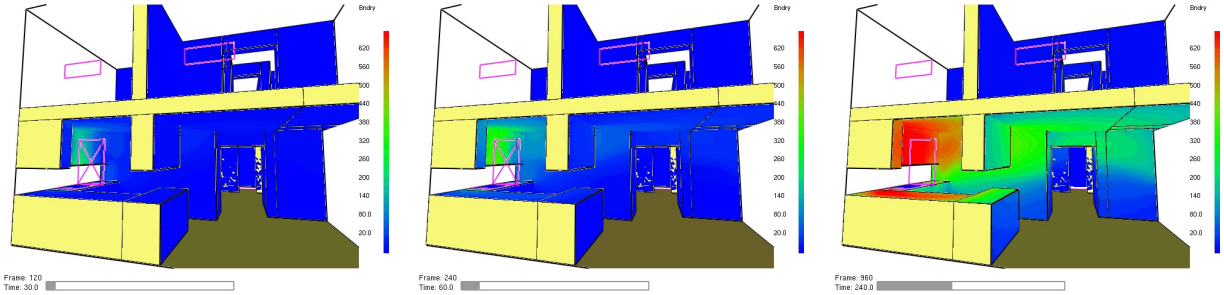
Data stored in Plot3D [8] files use a format developed by NASA and used by many CFD programs for representing simulation results. An FDS simulation creates a Plot3D file every DTSAM seconds. Plot3D data is visualized in three ways: as 2D contours, vector plots and isosurfaces. Figure 4.7a shows an example of a 2D Plot3D contour. 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. Figure 4.7b shows vectors in a doorway. Note the hot flow (red color) leaving the fire room in the upper part of the door and the cool flow (blue color) entering fire room in the lower part of the door. Figure 4.8 gives an example of isosurfaces for two different temperature levels. 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 one writes WRITE_XYZ=.TRUE. on the PL3D line, a file with suffix .xyz is written out. Smokeview does not require this file because the coordinate information can be obtained elsewhere.

4.7.6 Extracting Numbers from the Output Data Files

Often it is desired to present results of calculations in some form other than those offered by Smokeview. In this case, there is a short Fortran 90 program called **fds2ascii.f**, with a PC compiled version called **fds2ascii.exe**. To run the program, just type

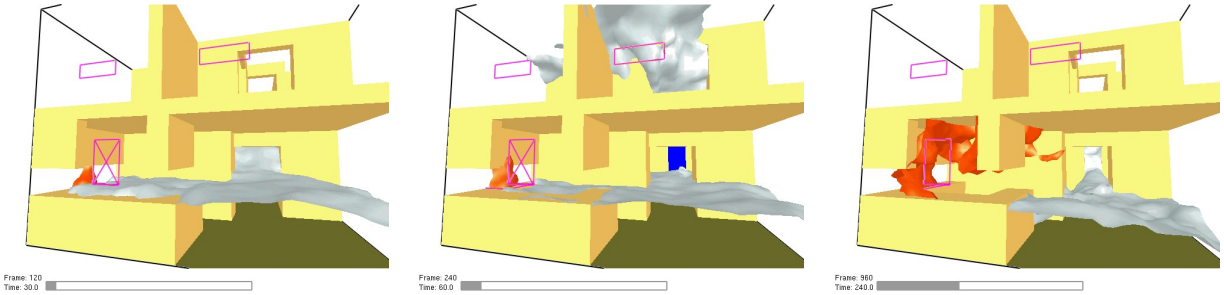
```
fds2ascii
```

at the command prompt. The user is asked a series of questions about which type of output file to process, what time interval to time average the data, and so forth. A single file is produced with the name **CHID_fds2ascii.csv**.



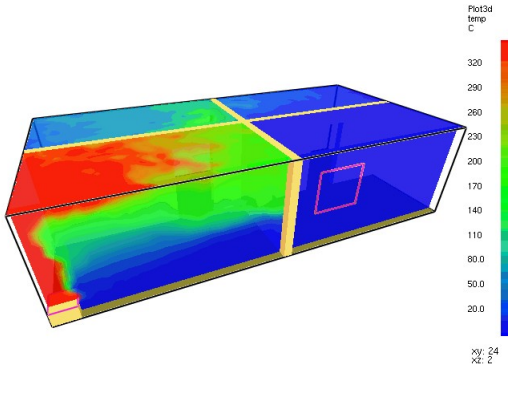
30 s 60 s 240

Figure 4.5: Boundary file snapshots of shaded wall temperatures at t=(30, 60, 240) s after ignition. The data file for these snapshots was generated by adding the line
`&BNDF QUANTITY='WALL_TEMPERATURE' /`
to an FDS input file.

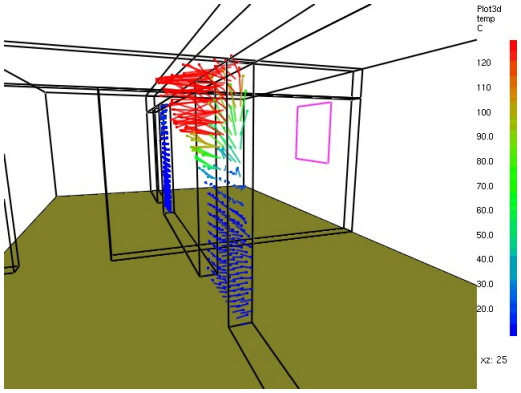


30 s 60 s 240 s

Figure 4.6: Isosurface file snapshots of mixture fraction levels at t=(30, 60, 240) s after ignition. The data file for these snapshots was generated by adding the line
`&ISOF QUANTITY='MIXTURE_FRACTION',VALUE(1)=0.05,VALUE(2)=0.001 /`
to an FDS input file.

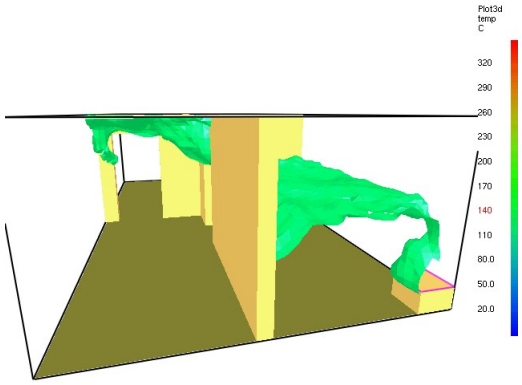


a) shaded 2D temperature contour plots in a vertical plane through the fire and a horizontal plane below the ceiling

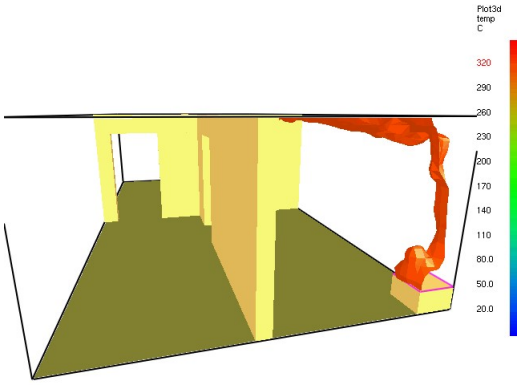


b) shaded temperature vector plot in a vertical plane through the doorway. The “a” key may be depressed to alter the vector sizes. The “s” key may be depressed to alter the number of vectors displayed.

Figure 4.7: Plot3D contour and vector plot examples.



a) isosurface at 140 °C



b) isosurface at 320 °C

Figure 4.8: Plot3D isocontour example.

Chapter 5

Special Features

The following sections describe some special features of the FDS model that are not usually invoked for engineering applications. Some of these features represent current research efforts and are to be considered fragile.

5.1 Stopping and Restarting Calculations

Normally, a simulation consists of a sequence of events starting from ambient conditions. However, there are occasions where the user 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 current working directory, and the phrase `RESTART=.TRUE.` needs to be added to the `MISC` line of the data file controlling the continued job. For example, suppose that the job whose `CHID` is "plume" is halted by the user who creates 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.data**, or whatever the user has chosen to name the input file. The existence of a restart file with the same `CHID` as the original job tells the code to save the new data in the same files as the old.

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 one wants 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 `DTCORE=50.` on the `MISC` line to save restart files every 50 s, for example. The default for `DTCORE` is infinity, meaning no restart files are created unless the user gracefully stops a job by creating a dummy file called **CHID.stop**.

Note that 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 at NIST, it should be considered a fragile construct. Examine the output to ensure that no sudden or unexpected events occur during the stop and restart.

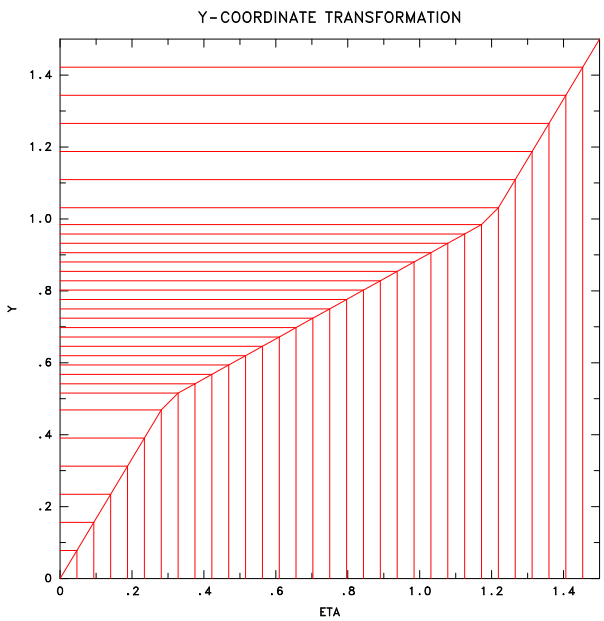


Figure 5.1: **Piecewise Linear Transformation.**

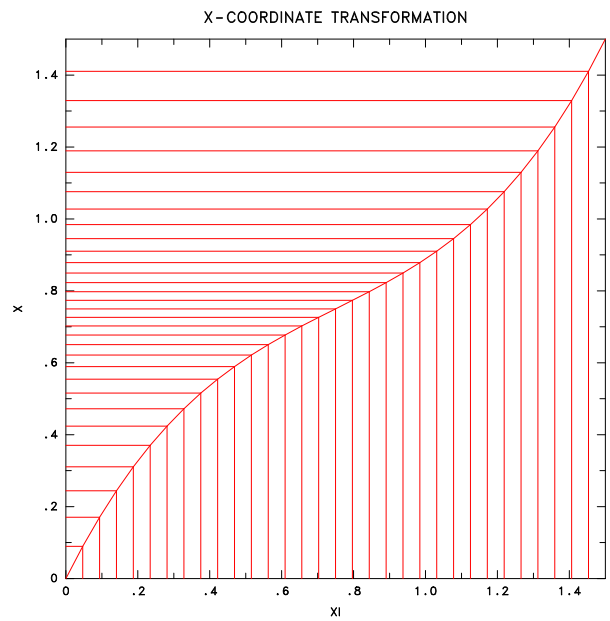


Figure 5.2: **Polynomial Transformation.**

5.2 Stretching the Grid: The TRNX, TRNY and/or TRNZ Namelist Groups

By default the grid 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 maps the uniformly-spaced computational grid to a non-uniformly spaced physical grid. Take the x direction as an example. A function $x = f(\xi)$ maps the uniformly-spaced Computational Coordinate (CC) ($\texttt{XBAR0} \leq \xi \leq \texttt{XBAR}$) to the Physical Coordinate (PC) ($\texttt{XBAR0} \leq x \leq \texttt{XBAR}$). The function has three mandatory constraints: it must be monotonic, it must map XBAR0 to XBAR0, and it must map XBAR to XBAR. The default transformation function is $f(\xi) = \xi$ (uniform grid). If uniform gridding is desired in the x direction, then nothing need be specified, and no &TRNX lines should be written. The same is true for the y and z directions.

Two types of transformation functions are allowed. The first, and simplest, is a piecewise linear function. Figure 5.1 gives an example of a piecewise linear transformation, in this case applied to the y coordinate direction. The graph indicates how the uniformly-spaced computational grid (horizontal axis) is mapped to the non-uniformly-spaced physical grid (vertical axis). In this case, the function is made up of straight line segments connecting user-prescribed points (CC,PC). Note that the points should be given in increasing order. The parameters used in this example are

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

The parameter CC refers to the Computational Coordinate, located on the horizontal axis; PC is the Physical Coordinate, located on the vertical axis. In this example, the uniformly-spaced grid cells between 0 and 0.3 on the horizontal axis are mapped to larger, but still uniformly-spaced, cells on the vertical axis between 0 and 0.5. Then the segment between 0.3 and 1.2 on the horizontal axis is mapped to a segment between 0.5 and 1.0 on the vertical axis. Finally, the segment between 1.2 and 1.5 on the horizontal axis is mapped to the segment between 1.0 and 1.5 on the vertical axis. The slopes of the line segments indicate whether the grid is being stretched (slopes greater than 1) or shrunk (slopes less than 1). The tricky part about this process is that the user usually has a desired shrinking/stretching strategy for the physical coordinate

(vertical axis), and must work backwards to determine what the corresponding points are in computational space (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

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

Figure 5.2 gives an example of a polynomial transformation. The parameters used in this example are

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

which correspond to the constraints $f(0.75) = 0.75$ and $\frac{df}{d\xi}(0.75) = 0.5$, or, in words, the function maps 0.75 into 0.75 and the slope of the function at $\xi = 0.75$ is 0.5 . Note that these constraints are optional. The two mandatory constraints $f(\texttt{XBAR0}) = \texttt{XBAR0}$ and $f(\texttt{XBAR}) = \texttt{XBAR}$ need not, and should not, be prescribed. The reason for this is that the mandatory constraints are already built into the linear system of equations that is solved to yield the coefficients of the *n*th order polynomial. In the above example, two optional constraints plus the two mandatory constraints yield a unique cubic polynomial $f(\xi) = c_0 + c_1\xi + c_2\xi^2 + c_3\xi^3$. More optional constraints lead to higher order polynomial functions. The monotonicity of the function is checked by the program and an error message is produced if it is not monotonic. Note that the transformation above is applied to the third mesh in a multiple mesh job.

The vertical and horizontal lines on Figs. 5.1 and 5.2 indicate how the uniformly spaced computational grid on the horizontal axis is transformed into a non-uniformly spaced physical grid on the vertical axis. The slope of the transformation function indicates the degree to which the computational cells is stretched (slope greater than 1) or shrunk (slope less than 1). Note that shrinking cells in one region necessarily leads to stretching cells elsewhere. When one or two coordinate directions are transformed, the aspect ratio of the grid cells in the 3D mesh 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 grid 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 grid cell, thus shrinking the grid in one or two directions may not necessarily lead to a better simulation if the third dimension is large. Also note that transformations, in general, reduce the efficiency of the computation, with two coordinate transformations impairing efficiency more than a transformation in one coordinate direction.

5.3 Initial Conditions: The `INIT` Namelist Group

Usually, an FDS calculation is initialized with constant ambient conditions; that is, the temperature, density and species mass fractions are fixed. However, there are some problems for which it is convenient to change the ambient conditions within some rectangular region of the domain. If so, add lines of the form

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

Here, within the region whose bounds are given by the sextuplet `XB`, the initial temperature shall be 30 °C instead of the ambient. This construct can also be used for `QUANTITY='DENSITY'` or `SPECIES_ID` where `SPECIES_ID` is a character string that is the name of a listed species.

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

A solid obstruction can be given an initial temperature via the parameter `TMPWAL0` on the `SURF` line.

5.4 Creating or Removing Obstructions; Opening or Closing Vents

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 entering a time via the parameter `T_REMOVE` or `T_CREATE` on the `OBST` line that defines the obstruction that is to be created or removed. For example, the line

```
&OBST XB=...,SURF_ID='whatever',T_CREATE=39. /
```

instructs the code to create the given obstruction 39 s after the start of the simulation. Likewise, the line

```
&OBST XB=...,SURF_ID='whatever',T_REMOVE=39. /
```

instructs the code to remove the obstruction after 39 s. To create an obstruction and then remove it, combine the instructions as follows

```
&OBST XB=...,SURF_ID='whatever',T_CREATE=30.,T_REMOVE=39. /
```

To remove an obstruction, then re-create one in its place, use 2 lines

```
&OBST XB=...,SURF_ID='whatever',T_REMOVE=30. /
&OBST XB=...,SURF_ID='whatever',T_CREATE=39. /
```

since the code simply sees this as two different obstructions. Experiment with these combinations using a simple case before trying a case to make sure that the code indeed is doing what is intended.

One additional feature is to time the removal or creation of an obstruction with a thermally-responsive device; in FDS the device being a `HEAT` detector. Assigning the character strings `HEAT_REMOVE` or `HEAT_CREATE` the name of a `HEAT` detector will direct FDS to remove or create the obstruction upon activation of the detector. For example, the lines

```
&OBST XB=...,SURF_ID='whatever',HEAT_REMOVE='det2' /
                    .
                    .
&HEAT XYZ=...,RTI=...,ACTIVATION_TEMPERATURE=87. ,LABEL='det1' /
&HEAT XYZ=...,RTI=...,ACTIVATION_TEMPERATURE=110.,LABEL='det2' /
```

causes the given obstruction to be removed when the specified `HEAT` detector activates. If either `HEAT_REMOVE` or `HEAT_CREATE` is set to `'ALL'`, then the desired action is triggered by the activation of *any* `HEAT` detector.

The removal and creation of an obstruction can be used to model the opening of a door that connects one part of the domain to another. If one wants to open or close a window or door to the outside of the computational domain (*i.e.* the atmosphere), a similar set of parameters can be used on the `VENT` line. For example, the line

```
&VENT XB=...,SURF_ID='OPEN',T_CLOSE=39. /
```

instructs the code to close a vent to the exterior of the domain at 39 s. The line

```
&VENT XB=...,SURF_ID='OPEN',HEAT_OPEN='det1' /
                    .
                    .
&HEAT XYZ=...,RTI=...,ACTIVATION_TEMPERATURE=87. ,LABEL='det1' /
&HEAT XYZ=...,RTI=...,ACTIVATION_TEMPERATURE=110.,LABEL='det2' /
```

instructs the code to open an initially closed vent to the outside when the first HEAT detector activates.

One additional feature associated with a VENT is to specify a time or a thermal device that controls when the VENT's SURF_ID is to be applied. For example, suppose it is desired that when the HEAT detector labelled 'det2', a fan should be turned on. Set

```
&SURF ID='FAN',VOLUME_FLUX=5. /
&VENT XB=...,SURF_ID='FAN',HEAT_ACTIVATE='det2' /
```

If it is desired that the fan be turned off, use the parameter HEAT_DEACTIVATE in the very same way. Note that T_ACTIVATE and T_DEACTIVATE are equivalent to T_OPEN and T_CLOSE. Also note that none of these control parameters should be applied to a MIRROR.

5.5 Extra Species

Normally when one specifies a fire via either HRRPUA or HEAT_OF_VAPORIZATION, the mixture fraction combustion model is applied. A single scalar variable, Z, represents the state of the combustion process from pure fuel ($Z = 1$) to pure air ($Z = 0$). The major reactants and products of combustion – fuel, O_2, CO_2, H_2O, N_2, CO and soot – are all pre-tabulated functions of the mixture fraction, Z. In other words, the value of Z in any given grid cell determines the mass fraction of all the gases listed. The user-defined stoichiometry listed under the REAC namelist group is used to generate the table associating the mass fractions with Z. One 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 a second scalar transport equation[1]. In fact, there does not need to be any fire at all – the FDS code can be used to transport a mixture of non-reacting ideal gases.

The namelist group SPEC is used to specify each additional species. Each SPEC line should include at the very least the name of the species via a character string called (ID). Next, if the ambient (initial) mass fraction of the gas is something other than 0, then the parameter MASS_FRACTION_0 is used to specify it. Several gases that can be included in a calculation are listed in Table 5.1. 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 5.1, its molecular weight MW must be specified in units of g/mol. In addition, if a DNS calculation is being performed, either the Lennard-Jones potential parameters σ (SIGMALJ) and ε/k (EPSILONKLJ) should be specified; or the VISCOSITY (kg/m/s), THERMAL_CONDUCTIVITY (W/m/K), and DIFFUSION_COEFFICIENT (m^2/s) between the given species and the background species should be specified.

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

There are three types of boundary conditions for the listed species. Reference to a given species is via its place in the input file. For example, the second listed species is N=2. If a simple no-flux condition is desired at a solid wall, do not set anything. If the mass fraction of the Nth species is to be some value at a forced flow boundary, set MASS_FRACTION(N) equal to the desired mass fraction on the appropriate SURF line. If the mass flux of the Nth species is desired, set MASS_FLUX(N) instead of MASS_FRACTION(N). If MASS_FLUX(N) is set, no VEL should be set. It is automatically calculated based on the mass flux. The mass flux is in units of kg/m^2/s.

[1] Often an extra gas introduced into a calculation is the same as a product of combustion, like water vapor from a sprinkler or carbon dioxide from an extinguisher. These gases are tracked separately, thus water vapor generated by the combustion is tracked via the mixture fraction variable and water vapor generated by evaporating sprinkler droplets is tracked via its own transport equation. In the case of sprinklers, do not specify WATER VAPOR as an extra species – it is done automatically.

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 (see Time Dependent Boundary Conditions in Section 4.4.1).

As an example, the lines

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

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

5.6 Finite-Rate or Premixed Combustion

By default, FDS assumes that the fire is essentially an infinitely-fast reaction between fuel and oxygen, and this reaction is not dependent on the surrounding gas temperature. It also assumes that the reaction zone is an infinitely thin sheet with fuel on one side and oxygen on the other. If a finite-rate or a pre-mixed reaction is desired, the following steps must be taken:

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: one may attempt to use the finite-rate reaction scheme in an LES calculation, but because the temperature in a large scale calculation is smeared out over a grid 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. Each SPEC line should include the name of the species (ID) and its ambient (initial) mass fraction, MASS_FRACTION_0. Several gases that can be included in a calculation are listed in Table 5.1. The physical properties of these gases are known and need not be specified. However, if a desired gas is not included in Table 5.1, its molecular weight MW must be specified in units of g/mol. In addition, if a DNS calculation is being performed, either the Lennard-Jones potential parameters σ (SIGMALJ) and ε/k (EPSILONKLJ) should be specified; or the VISCOSITY (kg/m/s), THERMAL_CONDUCTIVITY (W/m/K), and DIFFUSION_COEFFICIENT (m^2/s) between the given species and the background species should be specified. If the listed species is to be consumed or generated by the finite-rate reaction, its stoichiometric coefficient, NU, needs to be specified. Note that the background species cannot participate in the reaction except as a diluent.

Table 5.1: **Optional Gas Species [9]**

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

4. Read Section 5.5 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.

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

   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.

   XNO Exponent for oxygen concentration in one-step chemical reaction. (Default 1.)

   XNF Exponent for fuel concentration in one-step chemical reaction. (Default 1.)

   DELTAH The effective heat of combustion for one-step chemical reaction in units of kJ/kg. (Default 40,000 kJ/kg)

5.7 Pyrolysis Models

Fuel for a fire can be either a gas, solid or liquid. Gaseous fuels are easist to model because there is no phase change. In fact, specifying HRRPUA on the SURF line is equivalent to defining a gas burner with a specified flow rate. If this is all that is desired, skip this section. Otherwise, predicting the burning rate of solids and liquids requires more information to define the physical properties of the fuel.

5.7.1 Thermoplastics

By default, a solid burning object is assumed to be a homogenous material that burns at the surface. The alternative is to specify PHASE='CHAR' or PHASE='LIQUID' to direct the code to model the material as a char former or liquid, respectively (see below). For a thermoplastic material, specify the thermal properties of the material. The material can be regarded as thermally-thick or thin. In the former case, specify the thickness DELTA (m), DENSITY (kg/m^3), thermal conductivity KS or RAMP_KS (W/m/K), specific heat C_P or RAMP_C_P (kJ/kg/K). For a thermally-thin material, specify the product of the specific heat, thickness and density, C_DELTA_RHO (kJ/m^2/K); or specify separately the thickness DELTA (m), DENSITY (kg/m^3), and specific heat C_P or RAMP_C_P (kJ/kg/K). The default thickness is 0.1 m and the default density is 450 kg/m^3. Do not specify the thermal conductivity if the material is to be regarded as thermally-thin.

The burning rate of a thermoplastic is given by the expression

$$\dot{m}'' = A\,\rho_s\,e^{-E/RT} \tag{5.1}$$

The density of the material ρ_s is either directly prescribed via the parameter DENSITY, or is inferred from the other parameters. It is preferable to prescribe DENSITY explicitly. A is prescribed under the name A, with units of m/s. E, the activation energy, is prescribed via E in units of kJ/kmol. Remember that 1 kcal is 4.184 kJ, and be careful with factors of 1000. A and E are not readily accessible for most real fuels. However, if they are known, prescribe both. Avoid prescribing just one because they act as a pair. If A and E are not known, which is usually the case, prescribe MASS_FLUX_CRITICAL (kg/m^2/s) and TMPIGN (°C). This directs the code to choose A and E so that the fuel burns at the rate MASS_FLUX_CRITICAL when its surface reaches the temperature TMPIGN. Note that the temperature is read in using degrees Celsius, but is then converted to degrees Kelvin within the program. The default value of MASS_FLUX_CRITICAL is 0.02 kg/m^2/s. The default TMPIGN is 5000 °C (in other words, no burning occurs).

Remember to include the HEAT_OF_VAPORIZATION (kJ/kg) if the solid is to be regarded as a thermoplastic.

5.7.2 Charring Fuels

By default, a burning object is assumed to be a homogenous solid that burns at the surface. However, many materials burn internally, leaving char in the wake of a pyrolysis front that progresses into the material. To specify this type of burning behavior, set PHASE='CHAR' on the SURF line, then specify material properties for both the virgin material and the char. For the virgin material, the properties are those of a thermally-thick solid: DELTA (m), DENSITY (kg/m^3), KS or RAMP_KS (W/m/K), C_P or RAMP_C_P (kJ/kg/K). For the char, the parameters are similar: CHAR_DENSITY, KS_CHAR or RAMP_KS_CHAR, C_P_CHAR or RAMP_C_P_CHAR.

The burning behavior is described using similar parameters as a thermoplastic material:

$$\dot{m}'' = A\,(\rho_s - \rho_c)\,e^{-E/RT} \tag{5.2}$$

The density of the virgin material ρ_s and the density of the char ρ_c are both directly prescribed via the parameters DENSITY and CHAR_DENSITY. A is prescribed under the name A, with units of m/s. E, the activation energy, is prescribed via E in units of kJ/kmol. These parameters are not readily accessible for most real fuels. An alternative is to prescribe MASS_FLUX_CRITICAL (kg/m^2/s) and TMPIGN (°C). This directs the code to choose A and E so that the fuel burns at the rate MASS_FLUX_CRITICAL when its surface reaches the temperature TMPIGN. Note that the temperature is read in using degrees Celsius, but is then converted to degrees Kelvin within the program. The default value of MASS_FLUX_CRITICAL is 0.02 kg/m^2/s. The default TMPIGN is 5000 °C (in other words, no burning occurs).

Remember to include the HEAT_OF_VAPORIZATION (kJ/kg) if the solid is to be regarded as a charring fuel. Note that the HEAT_OF_VAPORIZATION refers to the gasification of the virgin material at the pyrolysis front. It is not an "effective" value that is often used to model the effect of the char shielding the virgin material from the heat flux at the surface.

One additional parameter describing a charring material is MOISTURE_FRACTION, giving the mass fraction of water in the virgin material. An entry for a charring material is given here, *only as an example*:

```
&SURF ID='SPRUCE'
      PHASE = 'CHAR'
      MOISTURE_FRACTION = 0.05
      DELTA=0.01
```

```
      TMPIGN=360.0
      HEAT_OF_VAPORIZATION=500.
      DENSITY = 450.
      RAMP_KS = 'KS',
      RAMP_C_P = 'CPV',
      RAMP_C_P_CHAR = 'CPC',
      RAMP_KS_CHAR = 'KSC',
      CHAR_DENSITY = 120.
      WALL_POINTS = 30
      BACKING = 'INSULATED'/
&RAMP ID = 'KS',  T = 20.,   F = 0.13  /
&RAMP ID = 'KS',  T = 500.,  F = 0.29  /
&RAMP ID = 'KSC', T = 20.,   F = 0.077 /
&RAMP ID = 'KSC', T = 900.,  F = 0.16  /
&RAMP ID = 'CPV', T = 20.,   F = 1.2   /
&RAMP ID = 'CPV', T = 500.,  F = 3.0   /
&RAMP ID = 'CPC', T = 20.,   F = 0.68  /
&RAMP ID = 'CPC', T = 400.,  F = 1.5   /
&RAMP ID = 'CPC', T = 900.,  F = 1.8   /
```

5.7.3 Liquid Fuels

For a liquid fuel, specify `PHASE='LIQUID'` on the `SURF` line. The thermal parameters are the same as those of a thermally-thick solid. The evaporation rate of the fuel is governed by the Clausius-Clapeyron equation (see FDS Technical Reference Guide for details). The only drawback of this approach is that the fuel gases burn regardless of any ignition source. Thus, if `PHASE='LIQUID'` is specified, the fuel begins burning at once. An example of a liquid fuel is

```
&SURF ID='METHANOL'
      HEAT_OF_VAPORIZATION=1101.
      PHASE='LIQUID'
      DELTA=0.01
      KS=0.20
      DENSITY=900.
      C_P=2.5
      TMPIGN=65. /
```

Note that `TMPIGN` is assumed to be the boiling temperature of the liquid. 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.

5.8 Burning Liquid Fuel Droplets

The evaporization of water droplets from sprinklers has been generalized so that a liquid fuel spray nozzle can be modeled. Fuel evaporation is triggered by the inclusion of the phrase `FUEL=.TRUE.` on the appropriate `PART` line. The spray nozzle characteristics are specified in the same way as those for a sprinkler. Fuel properties are specified as follows:

```
&PART ID='heptane',FUEL=.TRUE.,
      VAPORIZATION_TEMPERATURE=98.0,
      HEAT_OF_VAPORIZATION=316.0,
      SPECIFIC_HEAT=2.25,
      DENSITY=688.0,... /
```

The vaporization (boiling) temperature of the liquid fuel is in degrees Celsius, the heat of vaporization is in units of kJ/kg, the specific heat is in units of kJ/kg/K, and the density is in units of kg/m^3. `FUEL=.TRUE.` automatically invokes a mixture fraction calculation in which fuel from the evaporating fuel droplets is burned when it has diluted to the appropriate stoichiometric value. Note that this construct is fragile and subject to grid dependence. If the grid cells are too coarse, the evaporating fuel is diluted to such a degree that it never burns. Proper resolution depends on the type of fuel and the amount of fuel being ejected from the nozzle.

5.9 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 grid cell. In the gas phase, flames are extinguished due to lowered temperatures and dilution of the oxygen supply. See Section 4.4.3 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 fuel has been assigned a `HEAT_OF_VAPORIZATION`, there is no need to set any additional suppression parameters. It is assumed that water impinging on the fuel surface takes energy away from the pyrolysis process and thereby reduces the burning rate of the fuel. If the surface has been assigned a `HRRPUA` (Heat Release Rate Per Unit Area), a parameter needs to be specified that governs the suppression of the fire by water. An empirical way to account for fire suppression by water is to characterize the reduction of the pyrolysis rate in terms of an exponential function. The local mass loss rate of the fuel is expressed in the form

$$\dot{m}''_f(t) = \dot{m}''_{f,0}(t)\, e^{-\int k(t)\, dt} \tag{5.3}$$

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

$$k(t) = \texttt{E_COEFFICIENT}\, m''_w(t) \quad \mathrm{s}^{-1} \tag{5.4}$$

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.

Another parameter used in water suppression calculations is the fuel `POROSITY` which is the fraction of water that does not cascade down the side of the fuel package, either because of absorption or because it is trapped within the interior of the fuel. Both `POROSITY` and `E_COEFFICIENT` are 0 by default and both are prescribed on the `SURF` line.

5.10 Visibility

If one is 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 [10]. The intensity of monochromatic light passing a distance L through smoke is attenuated according to

$$I/I_0 = e^{-KL} \tag{5.5}$$

The light extinction coefficient, K, is a product of the density of smoke particulate, ρY_s, and a mass specific extinction coefficient that is fuel dependent

$$K = K_m \, \rho \, Y_s \tag{5.6}$$

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

$$S = C/K \tag{5.7}$$

where C is a nondimensional constant characteristic of the type of object being viewed through the smoke, *i.e.* $C = 8$ for a light-emitting sign and $C = 3$ for a light-reflecting sign [10]. Since K varies from point to point in the domain, the visibility S does as well. Keep in mind that FDS can only track smoke whose production rate and composition are specified. Predicting either is beyond the capability of the present version of the model.

Three parameter control smoke production and visibility; each parameter is input on the `REAC` line. The first parameter is `SOOT_YIELD`, which is the fraction of fuel mass that is converted to soot. The second parameter is called the `MASS_EXTINCTION_COEFFICIENT`, and it is the K_m in Eq. (5.6). The default value is 7600 m^2/kg, a value suggested for flaming combustion of wood and plastics. The third parameter is called the `VISIBILITY_FACTOR`, the constant C in Eq. (5.7). It is 3 by default.

The slice, Plot3D or thermocouple output quantity `extinction coefficient` is K. The visibility S is output via the keyword `visibility`. Note that each is tied to the mixture fraction formulation of combustion.

5.11 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 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. Conservation of energy dictates that

$$(H - z_{int}) \, T_u + z_{int} \, T_l = \int_0^H T(z) \, dz = I_1 \tag{5.8}$$

and conservation of mass (assuming a perfect gas) dictates that

$$(H - z_{int}) \, \frac{1}{T_u} + z_{int} \, \frac{1}{T_l} = \int_0^H \frac{1}{T(z)} \, dz = I_2 \tag{5.9}$$

Solving for z_{int}:

$$z_{int} = \frac{T_l (I_1 I_2 - H^2)}{I_1 + I_2 T_l^2 - 2 T_l H} \tag{5.10}$$

Letting T_l be the temperature in the lowest grid cell and using Simpson's Rule to perform the numerical integration of I_1 and I_2, T_u can be defined as the average upper layer temperature via

$$(H - z_{int})\, T_u = \int_{z_{int}}^{H} T(z)\, dz \tag{5.11}$$

Further discustion of this procedure can be found in Ref. [11].

The quantities `LAYER HEIGHT`, `UPPER TEMPERATURE` and `LOWER TEMPERATURE` can be displayed in Smokeview as slice (`SLCF`) files or in a spread sheet file via "thermocouple" (`THCP`) entries in the input file. For example, the entry

```
&THCP XYZ=2.0,3.0,0.5,QUANTITY='LAYER HEIGHT',LABEL='whatever' /
```

produces a time history of the smoke layer height at $x = 2$ and $y = 3$. The coordinate $z = 0.5$ is not used and can be anything so long as it is within the range of z defined on the `PDIM` line. Otherwise, the `THCP` line is ignored. Normally, the vertical integrations are carried out from the lowest grid cell to the highest ($1 \leq k \leq$ `KBAR`). To change this, add the parameters `K_LOW` and `K_HIGH` to the `THCP` or `SLCF` line. Note that for any one mesh, there can only be one set of limits.

If multiple meshes are being used, the `THCP` is assigned to the mesh corresponding to the `XYZ` coordinates. If two or more meshes overlap at this point, control goes to the mesh with the lowest index, that is, the mesh listed first in the input file. In the case of a slice file, add the parameter `MESH_NUMBER` to the `SLCF` line to indicate directly which mesh to use in the integration if there is any possible chance of confusion. This also reduces the computation time a bit since the unwanted meshes do not generate undesired slice files.

5.12 Leakage

For a limited class of scenarios, it is possible to prescribe leakage in an FDS calculation. Three criteria must be met. First, the boundary of the computational domain must coincide with the exterior walls of the compartment. Second, there must be no `OPEN` (passive) vents to the exterior. Third, only one mesh must be used. If these three criteria are satisfied, then prescribe `LEAK_AREA` on the `MISC` line, and assign to at least one of the wall materials (regardless of whether or not it is on the exterior boundary) the attribute `LEAKING=.TRUE.` on its `SURF` line. The program assigns the leakage uniformly to the `LEAKING` surfaces. The `LEAK_AREA` (m^2) is based on the formula

$$A_L = \dot{V}\, \sqrt{\rho_\infty / 2(p_0 - p_\infty)} \tag{5.12}$$

where $\dot{V}$ is the airflow rate (m^3/s) at a given background pressure p_0 (Pa). ρ_∞ is the ambient density (kg/m^3). Note that the discharge coefficient normally seen in this type of formula is assumed to be 1.

5.13 Fires and Flows in the Outdoors

Simulating a fire in the outdoors is not much different than a fire indoors, but there are several useful parameters that can easily be invoked. First, it is easy to prescribe the wind with a realistic increase in velocity with altitude. By default the velocity (wind) profile at any vent is a top hat, but the parameter `PROFILE` on the `SURF` line can yield other profiles. For example, `PROFILE='PARABOLIC'` produces a parabolic profile with `VEL` being the maximum velocity, and `'ATMOSPHERIC'` produces a typical atmospheric wind profile of the form $u = u_0(z/z_0)^p$. If an atmospheric profile is prescribed, also prescribe `Z0` for z_0 and `PLE` for p. `VEL` specifies the reference velocity u_0.

Another useful parameter for outdoor simulations is the temperature lapse rate of the atmosphere. Typically, in the first few hundred meters of the atmosphere, the temperature decreases several degrees Celsius per kilometer. These few degrees are important when considering the rise of smoke since the temperature of the smoke decreases rapidly as it rises. `DT0DZ` is the lapse rate of the atmosphere in units of °C/m. This need only be set for outdoor calculations where the height of the domain is tens or hundreds of meters. The default value of `DT0DZ` is -g/c_p ≈-0.0097 °C/m.

Figure 5.3 gives an example of an outdoor fire scenario, with the corresponding input data file given in Figure 5.4. The tanks are built from rectangular obstructions forming a cylindrical shape. The fire is specified on the top of one tank and a wind is prescribed at one of the computational boundaries.

5.14 2D 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 (axisymmetric) calculation can be performed by setting `JBAR=1` on the `&GRID` line, and for axisymmetry replacing the parameter `XBAR` with `RBAR` on the `&PDIM` line. No boundary conditions should be set at the planes $y =$ `YBAR0` or $y =$ `YBAR`, nor at $x =$ `RBAR0` in an axisymmetric calculation in which `RBAR0`=0. For better visualizations, the difference of `YBAR` and `YBAR0` should be small so that the Smokeview rendering appears to be in 2D. An example of an axially-symmetric helium plume is shown in Fig. 5.5 with the corresponding input data file in Fig. 5.6.

5.15 Restoring the Baroclinic Vorticity

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

$$\nabla \cdot \frac{1}{\rho} \nabla \tilde{p} = \frac{1}{\rho_0} \nabla^2 \tilde{p} \tag{5.13}$$

The consequence of this approximation is that the vorticity generated due to the non-alignment of the density and pressure gradients, or the baroclinic torque, is neglected. For most large scale applications, the assumption is justified by the fact that the vorticity generated by buoyancy is the dominant source of vorticity. By neglecting the baroclinic torque the solution of the elliptic partial differential equation obtained by taking the divergence of the momentum equation is greatly simplified. However, an option exists in the code to restore the baroclinic torque by decomposing the relevant term in the pressure equation

$$\nabla \cdot \frac{\nabla \tilde{p}}{\rho} = \nabla \cdot \frac{\nabla \tilde{p}}{\bar{\rho}} + \nabla \cdot \left(\frac{1}{\rho} - \frac{1}{\bar{\rho}} \right) \nabla \tilde{p} \tag{5.14}$$

and evaluating the second term on the right hand side with values of pressure from the previous time step. The expression $\bar{\rho}$ is an average density, equal to $2\rho_{\min}\rho_{\max}/(\rho_{\min} + \rho_{\max})$. To make this correction, simply include the statement

```
BAROCLINIC=.TRUE.
```

on the `MISC` line. In a DNS calculation (`DNS=.TRUE.`), the correction is made by default. However, for an LES calculation (default), the correction must be explicitly invoked. The cost of the correction is not prohibitive – try calculations with and without the correction to determine if its inclusion is warranted.

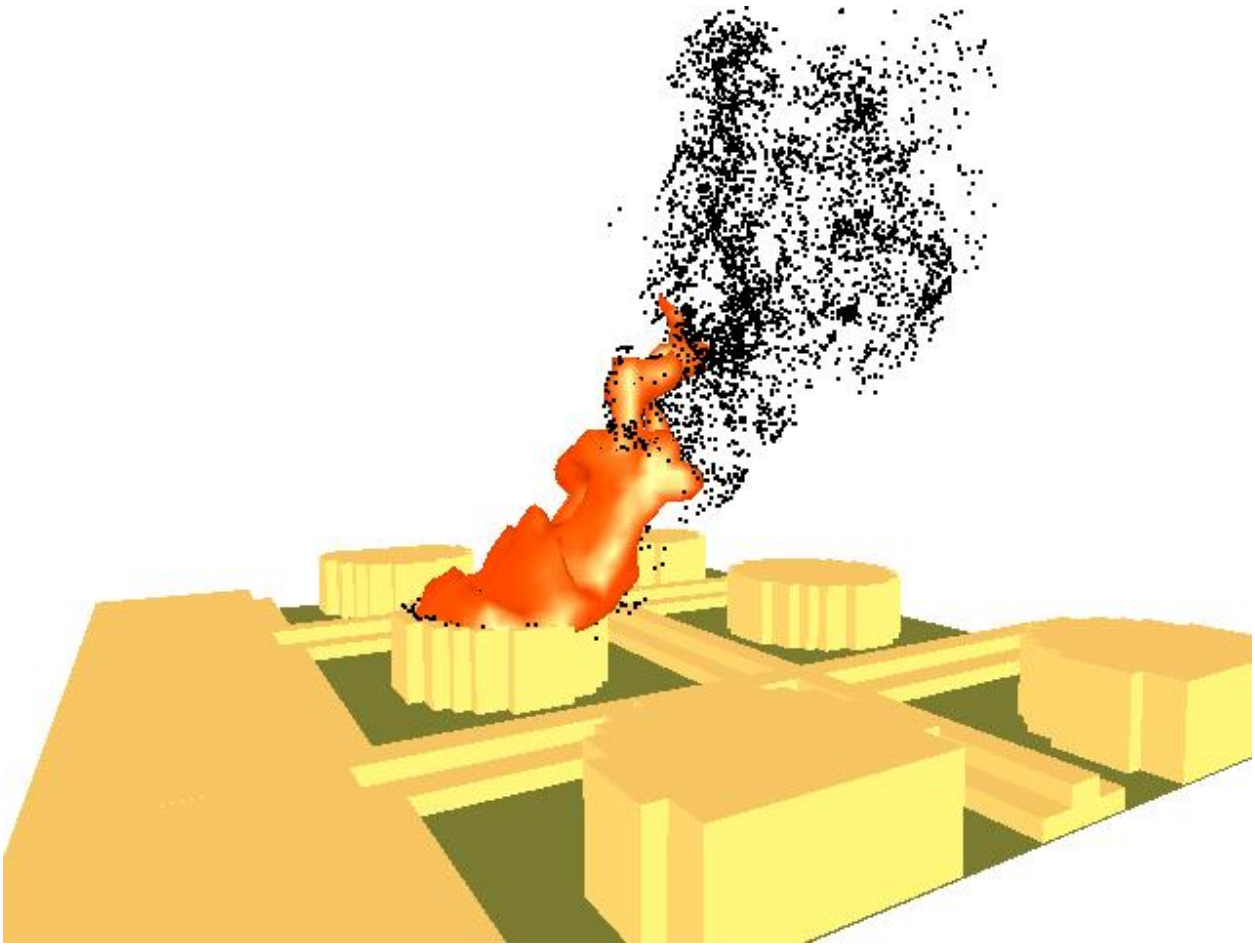


Figure 5.3: **Instantaneous snapshot of a large oil fire simulation. The domain is 384 m by 384 m by 288 m. Each tank is 84 m in diameter and 27 m tall. The grid is uniform with 6 m grid cells.**

```
&HEAD CHID='tankfarm2',TITLE='Japanese Oil Storage Tank Farm' /
&GRID IBAR=64,JBAR=64,KBAR=48 /
&PDIM XBAR0=64.,XBAR=448.,YBAR0=192.,YBAR=576.,ZBAR=288. /
&TIME TWFIN=60. /

&MISC DT0DZ=0. /

&SURF ID='WIND',VEL=-5.,PROFILE='ATMOSPHERIC',Z0=27.,PLE=0.15  /
&SURF ID='FIRE',HRRPUA=1500.,PART_ID='tracers' /
&PART ID='tracers',MASSLESS=.TRUE. /

&OBST XB=  0.,138.,  0.,768.,  0.,  9. /
&OBST XB=138.,144.,  0.,768.,  0.,  6. /
&OBST XB=144.,150.,  0.,768.,  0.,  3. /
                    .
                    .
                    .
&OBST XB=174.,180.,372.,396.,  0., 27.,SURF_IDS='FIRE','INERT','INERT' /
&OBST XB=180.,186.,360.,408.,  0., 27.,SURF_IDS='FIRE','INERT','INERT' /
                    .
                    .
&VENT CB='XBAR0',SURF_ID='WIND' /
&VENT CB='YBAR' ,SURF_ID='OPEN' /
&VENT CB='YBAR0',SURF_ID='OPEN' /
&VENT CB='XBAR' ,SURF_ID='OPEN' /
&VENT CB='ZBAR' ,SURF_ID='OPEN' /

&SLCF PBY=384.,QUANTITY='TEMPERATURE',VECTOR=.TRUE. /
&SLCF PBY=384.,QUANTITY='HRRPUV' /
&SLCF PBY=384.,QUANTITY='MIXTURE_FRACTION' /

&BNDF QUANTITY='HEAT_FLUX' /

&PL3D DTSAM=60. /
```

Figure 5.4: **Input data file for tankfarm scenario.**

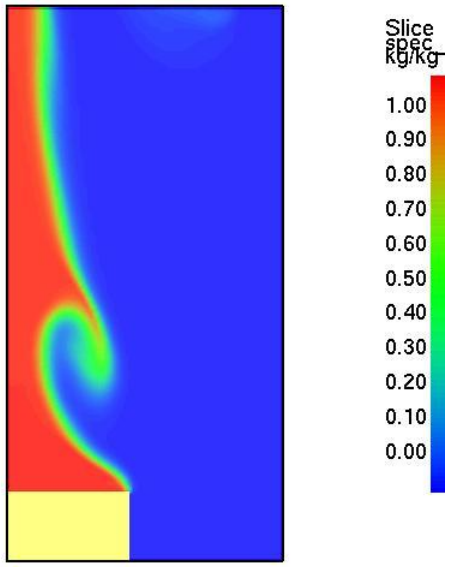


Figure 5.5: **Instantaneous snapshot of an axially-symmetric helium plume.**

```
&HEAD CHID='helium',TITLE='Axisymmetric Helium Plume' /
&GRID IBAR=72,JBAR=1,KBAR=144 /
&PDIM RBAR=0.08,YBAR=0.001,ZBAR=0.16 /
&TIME TWFIN=10.0 /
&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 CB='XBAR' ,SURF_ID='OPEN' /
&VENT CB='ZBAR' ,SURF_ID='OPEN' /
&OBST XB= 0.0,0.036, 0.00, 0.01,0.00,0.02,SURF_IDS='HELIUM','INERT','INERT' /
&PL3D DTSAM=2.0,QUANTITIES(1)='DENSITY',QUANTITIES(5)='HELIUM' /
&SLCF PBY=0.000,QUANTITY='DENSITY',VECTOR=.TRUE. /
&SLCF PBY=0.000,QUANTITY='HELIUM' /
```

Figure 5.6: **Input data file for helium plume calculation.**

5.16 Fine-Tuning the Radiation Transport Model

There are several ways to improve the performance of the Finite Volume Method in solving the radiation transport equation (RTE), most of which increase the computation time. The solver has two modes of operation – a gray gas model (default) and a wide band model [1]. Modifications to these models can be made via a namelist group called `RADI`. If running in gray gas mode (default), increase the number of angles from the default 100 with the integer parameter `NUMBER_RADIATION_ANGLES`. The frequency of calls to the radiation solver can be reduced from every 3 time steps with integer `TIME_STEP_INCREMENT`. The increment over which the angles are updated can be reduced from 5 with the integer `ANGLE_INCREMENT`. Briefly, if `TIME_STEP_INCREMENT` and `ANGLE_INCREMENT` are both set to 1, the radiation field is completely updated in a single time step, but the cost of the calculation increases significantly.

A few parameters affecting the absorption of radiation by water droplets are as follows: `RADTMP` is the assumed radiative source temperature. It is used in the computation of the mean scattering and absorption cross sections of water droplets. The default is 900 °C. `NMIEANG` is the number of discretization points of the polar angle in the integration of the single droplet 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 that the radiation solver has been designed to work with the mixture fraction combustion model. If the mixture fraction model is not being used, the radiation solver can still be used if `CARBON DIOXIDE` is prescribed as a product of combustion, or if a constant absorption coefficient `KAPPA0` is specified on the `RADI` line. Note also that it is possible to turn off the radiation transport solver (saving roughly 20 % in CPU time) by adding the statement `RADIATION=.FALSE.` to the `MISC` line. For isothermal calculations, the radiation is turned off automatically. If burning is taking place and radiation is turned off, then `RADIATIVE_FRACTION` from the `REAC` line is subtracted from the total heat release rate. This radiated energy completely disappears from the calculation.

5.17 Defying Gravity

Most users of FDS assume that the acceleration of gravity points toward the negative end of the z axis, or more simply, downward. However, to change the direction of gravity to model a sloping roof or tunnel, for example, specify the gravity vector on the `MISC` line with a triplet of numbers of the form `GVEC=0.0,0.0,-9.81` (units are m/s^2). This is the default, but it can be changed to be any direction.

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

5.18 Isothermal and Salt Water Simulations

Although FDS was designed specifically for fire simulations, it can be used for other fluid flow simulations that do not include fire or heat addition of any kind. First, set `ISOTHERMAL=.TRUE.` on the `MISC` line. This logical parameter indicates that the calculation does not involve any change in temperature and no radiation heat transfer, thus reducing the number of equations that must be solved, simplifying those that are, and reducing the computation time.

A valuable model validation exercise is to simulate the mixing of fresh and salt water. These types of experiments have traditionally been performed to mimic the movement of smoke in a building. Although FDS is rarely used for this purpose now, the capability still exists in the code to handle fresh/salt water interactions. To invoke this feature, include the statements

```
&MISC ISOTHERMAL=.TRUE.,BACKGROUND_SPECIES='FRESH WATER',DENSITY=1000.,
      VISCOSITY=1.0E-3,SC=1. /
&SPEC ID='SALT WATER',DENSITY=1052.,VISCOSITY=1.0E-3 /
```

in the input file, along with appropriate boundary conditions for the salt water. Note that the names `'FRESH WATER'` and `'SALT WATER'` have no particular meaning as far as the code is concerned. What matters is the designation of `DENSITY` and `VISCOSITY` for both, plus a global Schmidt number `SC`. Without the designation `DNS=.TRUE.`, the calculation is performed as a Large Eddy Simulation (LES), in which case the `VISCOSITY` serves as a lower bound for the Smagorinsky viscosity, and the Schmidt number is used to relate the viscosity to the material diffusivity.

5.19 Non-rectangular Geometry

The efficiency of FDS is due to the simplicity of its numerical grid. However, there are situations in which certain geometric features do not conform to the rectangular grid. 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.

5.20 Texture Mapping

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

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

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

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

applies paneling to an obstruction whose dimensions are 1 m by 1 m by 2 m, such that the image of the paneling is positioned at the point (1.0,3.0,5.0). The default value of `TEXTURE_ORIGIN` is (0,0,0), and the global default can be changed by added a `TEXTURE_ORIGIN` statement to the `MISC` line.

Chapter 6

Conclusion

The equations and numerical algorithm described in this document form the core of an evolving fire model. As research into specific fire-related phenomena continues, the relevant parts of the model can be improved. Because the model was originally designed to analyze industrial scale fires, it can be used reliably when the fire size is specified and the building is relatively large in relation to the fire. In these cases, the model predicts flow velocities and temperatures to within 20 % of the experimental measurements. In cases where the fire is large relative to the enclosure, the uncertainty of the model is greater due both to the lack of input data for material properties and combustion chemistry and to greater numerical error in combustion and radiation transport.

Any user of the numerical model must be aware of the assumptions and approximations being employed. There are two issues to consider before embarking on calculations. 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, the physical phenomena of interest may not be simulated due to limitations in the model algorithms or numerical grid. Except for those few materials that have been studied to date at NIST, the user must supply the thermal properties of the materials, and then validate the performance of the model with experiments to ensure that the model has the necessary physics included. Only then can the model be expected to predict the outcome of fire scenarios that are similar to those that have actually been tested.

Bibliography

[1] K.B. McGrattan (editor). Fire Dynamics Simulator (Version 4), Technical Reference Guide. NIST Special Publication 1018, National Institute of Standards and Technology, Gaithersburg, Maryland, July 2004.

[2] G.P. Forney and K.B. McGrattan. User's Guide for Smokeview Version 4. NIST Special Publication 1017, National Institute of Standards and Technology, Gaithersburg, Maryland, July 2004.

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

[4] C.M. Fleischmann and F.F. Chen. Radiant Ignition of Upholstered Furniture. In *Proceedings of the International Conference on Engineered Fire Protection Design*, 2001. Society of Fire Protection Engineers, Bethesda, Maryland.

[5] U.O. Koylu and G.M. Faeth. Carbon Monoxide and Soot Emissions from Liquid-Fueled Buoyant Turbulent Diffusion Flames. *Combustion and Flame*, 87:61–76, 1991.

[6] T. Ma. Numerical Simulation of an Axi-symmetric Fire Plume: Accuracy and Limitations. Master's thesis, University of Maryland, 2001.

[7] S. Welsh and P. Rubini. Three-dimensional Simulation of a Fire-Resistance Furnace. In *Fire Safety Science – Proceedings of the Fifth International Symposium*. International Association for Fire Safety Science, 1997.

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

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

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

[11] Y.P. He, A. Fernando, and M.C. Luo. Determination of interface height from measured parameter profile in enclosure fire experiment. *Fire Safety Journal*, 31:19–38, 1998.

Appendix A

Compiling the Source Code for FDS

This section describes how one should compile the FDS source code to run on a variety of different computer platforms. 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 **fds4.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, the user must compile the code. Fortran 90/95 and C compilers are needed for compilation.

A.1 Serial Compilation

Table A.1 lists the files that make up the source code. The files with suffix ".f" contain fixed form Fortran 90 instructions conforming to the ANSI and ISO standards, with the exception of those instructions that pass information between the C and Fortran routines. The source files should be compiled in the order in which they are listed in Table A.1 because some routines are dependent on others. For Unix/Linux users, **Makefiles** for various platforms have been provided that assist in the compilation. Compiler options differ from platform to platform. Note the following:

- The source code consists mainly of *fixed-format* Fortran 90 statements organized into 13 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**. All the other files have the suffix .f, indicating fixed-format Fortran.

- 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, check for compiler options that prevent the compiler from adding the underscore to the subroutine names.

- The only non-standard Fortran used is a call to a routine called `FLUSH`. This routine forces the contents of a given output file buffer to be emptied, making it easier to check on the progress of a given calculation. Most Fortran compilers accept this routine, but if a compiler complains, comment out the line in **mods.f** containing the string `CALL FLUSH(IUNIT)`.

- The only compiler option necessary, in addition to any needed to address the above issues, is for full optimization (usually -O). Some compilers have a standard optimization level, plus various degrees of "aggressive" optimization. Be cautious in using the highest levels of optimization.

- Table A.2 lists some command and options for various different compilers and/or operating systems. Consult the compiler manual, help/man pages for more details and options. Report any problems if these options do not work.
- For the single processor version of FDS, use **main.f**
- The parallel version of FDS uses **main_mpi.f** instead of **main.f**, plus additional MPI libraries need to be installed. See next section for more details about parallel compilation.

Table A.1: **Source Code Files**

File Name	Description
mods.f	Global arrays and constants
misc.f	Miscellaneous routines
pois.f	Poisson (pressure) solver
radi.f	Radiation solver
sprk.f	Lagrangian particle transport and sprinkler activation
read.f	Read input parameters
init.f	Initialize variables and Poisson solver
divg.f	Compute the flow divergence
pres.f	Spatial discretization of pressure (Poisson) equation
mass.f	Mass equation(s) and thermal boundary conditions
velo.f	Momentum equations
dump.f	Dumps output data into files
isob.c	C Routine for computing isosurface triangles
main.f, main_mpi.f	Main programs, serial and parallel versions

Table A.2: **Commands and options for various Fortran and C compilers**

Compiler	Fortran Command and Options	C Command and Options
Lahey/Fujitsu	lf95 -O –staticlink	gcc -O
SGI/IRIX	f90 -O3 -OPT:Olimit=0	cc -O2 -n32 -mips4
IBM/AIX	xlf95 -O5 -qfixed=72	xlc -O
Intel Fortran	ifc -O3 -xiW -ip -Vaxlib	gcc -O -Dpp_noappend
HPUX	f90 -O3	cc -O -DHPUX_NOAPPEND
Sun	f90/f95 -O2	cc -O
DEC	f90 -O5 -fast	cc -O -Dpp_noappend
Digital Visual Fortran	See Section A.2	See Section A.2

A.2 Parallel Compilation using Windows and MPICH

Following is a set of very specific instructions for compiling and running the parallel version of FDS under Microsoft Windows using a version of MPI known as MPICH, a product of Argonne National Laboratory in the USA, and the Digital Visual Fortran (DVF) compiler. For those who do not have a Fortran compiler of any kind, a pre-compiled executable file **fds4_mpi.exe** is available from the FDS/Smokeview web site. In either case, all must follow the directions of installing MPICH and running the code. Those without a compiler can skip steps 3 and 4 below.

1. Download the Windows version of MPICH from http://www-unix.mcs.anl.gov/mpi/mpich/download.html

2. Install MPICH for each machine. The SDK needs to be installed only on the machine used to compile FDS.

3. (Only for those with DVF.) Create a new project for a Fortran Console application.

4. (Only for those with DVF.) Apply the following settings:

   - Project ⇒Settings ⇒C/C++, Pre-processor Definitions, Add the phrase **pp_noappend**
   - Project ⇒Settings ⇒Link, Output, Stack Allocations 50000000
   - Project ⇒Settings ⇒Link, General, add **dfor.lib** to the beginning of the list of objects
   - (older versions of compiler) Project ⇒Settings ⇒Fortran; Libraries; Check "Use Debug C Libraries"
   - Project ⇒Settings ⇒Link Input, add C:\Program Files\MPICH\SDK\LIB to 'Additional Library Path'
   - Project ⇒Settings ⇒Fortran Preprocessor, add C:\Program Files\MPICH\SDK\INCLUDE to 'INCLUDE AND USE PATHS'
   - Project ⇒Settings ⇒Link Input, add **ws2_32.lib** and **mpich.lib** to list of objects/libraries.
   - Under Project ⇒Settings ⇒Fortran External Procedures, set Argument Passing Conventions to "C. By Reference", and String Length Argument Passing to "After All Args". (equivalent to /iface:cref /iface:nomixed_str_len_arg)
   - Import source files, inluding **main_mpi.f**, but not **main.f** (this is for serial version)
   - Build

5. Make sure that each machine can find the TCP/IP address of other machines by name. This can be ensured by adding the IP address-name combinations to the C:\Windows\system32\drivers\etc\hosts file.

6. Parallel jobs are launched either using the graphical MPIRun interface, or by executing mpirun from the command line. For easier use, add directory C:\Program Files\MPICH\mpd\bin to PATH. To do this, under My Computer ⇒Properties ⇒Advanced ⇒Environment Variables, add the above directory name to the Path list.

   For command line use, the easiest way to define things for mpirun is to use a configuration file. Two examples of the config.txt for a job containing three meshes are shown below.

   A. **fds4_mpi.exe** is located in the same place on each computer, and both computers machine1 and machine2 can see the current directory. Here we are using 2 processors on machine1 (or two threads if machine1 only has one processor), and 1 processor on machine2.

```
exe C:\nist\fds\fds4_mpi.exe
hosts
machine1 2
machine2 1
```

B. **fds4_mpi.exe** can be found on the servermachine, and the working directory \fds\test is on a shared drive of machine1.

```
exe \\servermachine\nist\fds\fds4_mpi.exe
dir \\machine1\fds\Test
hosts
machine1 2
machine2 1
```

To start the simulation, type 'mpirun config.txt' on the command line. Remember that the input file name should be written into a file called **fds.data** See MPICH documentation for additional information.

Note: Make sure the working directory is fully "shared" since all computers have to write files into it. This is often the biggest impediment to parallel processing – one must make the connection between machines as transparent as possible. Have the same username and password on all machines. Check that all machines can "see" each other. And be patient. Even experienced computer users are going to have problems at first ensuring that everything has been set up properly.

Appendix B

Alphabetical List of Input Parameters

This section lists the input parameters for FDS as a function of Namelist group, in alphabetical order. It should only be used as a quick reference. Do not use this list before reading the detailed description of the parameters. The reason 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 that are necessary to describe the desired scenario.

`BNDF` (Boundary File Parameters)				
`DTSAM`	Real	Time increment	s	`TWFIN/NFRAMES`
`QUANTITY`	Character	Quantity to visualize		

`GRID` (Grid Parameters)				
`IBAR`	Integer	No. cells in *x* direction		10
`JBAR`	Integer	No. cells in *y* direction		10
`KBAR`	Integer	No. cells in *z* direction		10

`HEAD` (Header Material)				
`CHID`	Character	Job Identification String		`'output'`
`TITLE`	Character	Title for job		

HEAT (Heat Detectors)				
ACTIVATION_TEMPERATURE	Real	Link temperature	°C	74
LABEL	Character	Descriptor		
RTI	Real	Response Time Index	$\sqrt{\text{m s}}$	100
XYZ	Real Triplet	Coordinates	m	

HOLE (Obstruction Cutout Parameters)				
HEAT_CREATE	Character	Name of HEAT detector to create		
HEAT_REMOVE	Character	Name of HEAT detector to remove		
RGB	Real Triplet	Color indices for obstruction(s) to be cut		
T_CREATE	Real	Time to Create Hole	s	0
T_REMOVE	Real	Time to Remove Hole	s	1000000
XB	Real Sextuplet	Physical coordinates	m	

INIT (Initial Conditions)				
QUANTITY	Character	Parameter to initialize		
VALUE	Real	Initial value		
XB	Real Sextuplet	Coordinates	m	

ISOF (Isosurface Parameters)				
DTSAM	Real	Time increment	s	TWFIN/NFRAMES
QUANTITY	Character	Quantity to visualize		
VALUE	Real	Contour value		

MISC (Miscellaneous Parameters)				
BACKGROUND_SPECIES	Character	Background species		
BAROCLINIC	Logical	Baroclinic torque correction		.FALSE.
CSMAG	Real	Smagorinsky constant		0.20
DATABASE	Character	Name of Database file		
DATABASE_DIRECTORY	Character	Name of Database directory		
DENSITY	Real	Density of background liquid	kg/m^3	
DNS	Logical	Direct Numerical Simulation		.FALSE.
DT0DZ	Real	Atmospheric temperature lapse rate	°C/m	-0.0097
DTCORE	Real	Time between core dumps	s	
DTPAR	Real	Time between particle inserts	s	0.05
DTSPAR	Real	Time between sprinkler droplets	s	0.05
GVEC	Real triplet	Gravity vector	m/s^2	0,0,-9.81
ISOTHERMAL	Logical	Isothermal calculation		.FALSE.
INCOMPRESSIBLE	Logical	Incompressible calculation		.FALSE.
NFRAMES	Integer	Number of Frames of output data		1000
PR	Real	Prandtl number (LES only)		0.5
PINF	Real	Ambient pressure	Pa	101325
POROUS_FLOOR	Logical	Droplets disappear at floor		.TRUE.
RADIATION	Logical	Radiation calculation flag		.TRUE.
REACTION	Character	Combustion reaction identifier		
RESTART	Logical	Restart previous calculation		.FALSE.
SC	Real	Schmidt number (LES only)		0.5
SMOKE3D	Logical	Flag for 3D Smoke Visualization		.TRUE.
SUPPRESSION	Logical	Suppression calculation flag		.TRUE.
SURF_DEFAULT	Character	Default SURFace type		'INERT'
TEXTURE_ORIGIN	Char. Triplet	Origin of Texture Map	m	
TMPA	Real	Ambient Temperature	°C	20.
TMPO	Real	Outside Temperature	°C	20.
U0,V0,W0	Reals	Prevailing velocity field	m/s	0.

OBST (Obstruction Parameters)				
BNDF_BLOCK	Logical	Draw Boundary Info		.TRUE.
HEAT_CREATE	Character	Name of Controlling Heat Detector		
HEAT_REMOVE	Character	Name of Controlling Heat Detector		
COLOR	Character	Color		
OUTLINE	Logical	Draw as Outline		.FALSE.
PERMIT_HOLE	Logical	Allow a Hole		.TRUE.
RGB	Real Triplet	Color indices		
SAWTOOTH	Logical	Smooth Obstruction		.FALSE.
SURF_ID	Character	Associated Surface		
SURF_IDS	Character Triplet	Associated Surfaces (top-side-bottom)		
SURF_ID6	Character Sextuplet	Associated Surfaces (like XB		
T_CREATE	Real	Time to Create Obstruction	s	
T_REMOVE	Real	Time to Remove Obstruction	s	
TEXTURE_ORIGIN	Real Triplet	Coordinates of Texture Map		
THICKEN	Logical	No Thin Obstructions		.FALSE.
XB	Real Sextuplet	Physical coordinates	m	

PART (Lagrangian Particles/Droplets)				
AGE	Real	Droplet lifetime	s	100000
DENSITY	Real	Droplet density	kg/m^3	1000
DIAMETER	Real	Median Volumetric Diameter	μm	100
DROPLETS_PER_SECOND	Integer	Drops per second per head		1000
FUEL	Logical	Liquid Fuel		.FALSE.
GAMMA_D	Real	Parameter for size distribution		2.4
HEAT_OF_COMBUSTION	Real	Heat of Combustion	kJ/kg	
HEAT_OF_VAPORIZATION	Real	Latent Heat of Vap	kJ/kg	2259
ID	Character	Identifier		
INITIAL_TEMPERATURE	Real	Initial Temperature	°C	TMPA
MASSLESS	Logical	Massless tracers		.FALSE.
MASS_PER_VOLUME	Real	Droplet mass per unit volume	kg/m^3	1
MELTING_TEMPERATURE	Real	Melting Temperature	°C	0
NUMBER_INITIAL_DROPLETS	Integer	Number of droplets at start		0
QUANTITY	Character	Quantity for coloring		
SAMPLING_FACTOR	Integer	Filter for output file		1
SPECIFIC_HEAT	Real	Droplet specific heat	kJ/kg/K	4.184
STATIC	Logical	Stationary Particles		.FALSE.
VAPORIZATION_TEMPERATURE	Real	Liquid Droplet Boiling Temp	°C	100
WATER	Logical	Water Droplet		.FALSE.

PDIM (Physical Dimensions)				
XBAR0, XBAR	Real	Limits of *x* coordinate	m	
YBAR0, YBAR	Real	Limits of *y* coordinate	m	
ZBAR0, ZBAR	Real	Limits of *z* coordinate	m	
RBAR0, RBAR	Real	Limits of *r* (cylindrical coordinates)	m	

PIPE (Sprinkler Pipe Parameters)				
DELAY	Real	Time delay after first activation	s	0
PRESSURE	Real	Pressure in sprinkler pipe	bar	

PL3D (Plot3D Parameters)				
DTSAM	Real	Time increment	s	TWFIN/5
QUANTITIES	Character Quintuplet	Name of Quantities to save		
WRITE_XYZ	Logical	Write Coordinate File		.FALSE.

RADI (Radiation Parameters)				
ANGLE_INCREMENT	Integer	Number of angles skipped per update		5
CH4_BANDS	Logical	Include extra fuel bands		.FALSE.
KAPPA0	Real	Constant absorption coefficient	1/m	0
NMIEANG	Integer	Number of polar angles		15
NUMBER_RADIATION_ANGLES	Integer	Number of solid angles		100
PATH	Real	Path length for radiation calc.	m	
RADTMP	Real	Assumed radiative source temp.	°C	900
TIME_STEP_INCREMENT	Integer	Number time steps skipped		3
WIDE_BAND_MODEL	Logical	Non-gray gas assumption		.FALSE.

RAMP (Ramping Parameters)				
F	Real	Function value		
ID	Character	Identifier		
T	Real	Time (or Temperature)	s (or °C)	

REAC (Reaction Parameters)				
BOF	Real	Pre-exponential Factor	cm^3/mol/s	
DELTAH	Real	Heat of Combustion	kJ/kg	
E	Real	Activation Energy	kJ/kmol	
EPUMO2	Real	Energy per Unit Mass O_2	kJ/kg	
FUEL	Character	Name of Fuel		
FUEL_N2	Real	Number of N_2 molecules in fuel		0
ID	Character	Identifier		
MASS_EXTINCTION_COEFFICIENT	Real	Visibility Parameter	m^2/kg	7600
MAXIMUM_VISIBILITY	Real	Visibility Parameter	m	30
MW_FUEL	Real	Molecular Weight of Fuel	g/mol	44
NU_CO2	Real	Stoich. Coefficient for CO_2		3
NU_FUEL	Real	Stoich. Coefficient for Fuel		1
NU_H2O	Real	Stoich. Coefficient for Water		4
NU_O2	Real	Stoich. Coefficient for Oxygen		5
RADIATIVE_FRACTION	Real	Radiative Loss Fraction		0.35
CO_YIELD	Real	Fraction of CO from fuel		
SOOT_YIELD	Real	Fraction of soot from fuel		0.01
VISIBILITY_FACTOR	Real	Visibility Parameter		3
XNO, XNF	Real	Arrhenius Exponent		
Y_F_INLET	Real	Mass Fraction of Fuel in Burner	kg/kg	1.0
Y_O2_INFTY	Real	Ambient Oxygen Mass Fraction	kg/kg	0.23

SLCF (Slice File Parameters)				
CB	Character	Plane to save slice file		
DTSAM	Real	Time increment	s	TWFIN/5
K_HIGH	Integer	Upper K value for LAYER HEIGHT		KBAR
K_LOW	Integer	Lower K value for LAYER HEIGHT		1
MESH_NUMBER	Integer	Save only slices in this mesh		
PBX	Real	*x*-plane to save slice file		
PBY	Real	*y*-plane to save slice file		
PBZ	Real	*z*-plane to save slice file		
QUANTITY	Character	Name of Quantity to save		
VECTOR	Logical	Include flow vectors		.FALSE.
WRITE_XYZ	Logical	Write Coordinate File		.FALSE.
XB	Real Sextuplet	Coordinates of region to save	m	

SPEC (Species Parameters)				
DENSITY	Real	Density of Liquid Species	kg/m^3	
DIFFUSION_COEFFICIENT	Real	Diffusivity *D*	m^2/s	
EPSILONKLJ	Real	Leonard-Jones Parameter		0
ID	Character	Name of species		
MASS_FRACTION_0	Real	Initial mass fraction		0
MW	Real	Molecular Weight	g/mol	29
NU	Real	Stoich. coefficient		
SIGMALJ	Real	Leonard-Jones Parameter		0
THERMAL_CONDUCTIVITY	Real	Conductivity *k*	W/m/K	
VISCOSITY	Real	Dynamic Viscosity *mu*	kg/m/s	

SPRK (Sprinkler Parameters)				
LABEL	Character	Identifying Label for output		
MAKE	Character	Name of sprinkler (.spk) file		
ORIENTATION	Real Triplet	Direction vector		0,0,-1
PART_ID	Character	Name of associated PART line		
ROTATION	Real Triplet	Rotation Angle	deg	0
T_ACTIVATE	Real	User-prescribed Activation	s	
T_DEACTIVATE	Real	User-prescribed De-Activation	s	
XYZ	Real Triplet	Physical coordinates	m	

SURF (Surface Properties)				
ADIABATIC	Logical	Adiabatic thermal BC		.FALSE.
A	Real	Pre-exponential factor	m/s	6.5E5
BACKING	Character	Back face boundary condition		'VOID'
BURN_AWAY	Logical	Remove burnt objects		.FALSE.
BURNING_RATE_MAX	Real	Maximum Burning Rate	$kg/m^2/s$	0.1
C_DELTA_RHO	Real	Specific heat x thickness x density	m	
C_P	Real	Specific heat	kJ/kg/K	
C_P_CHAR	Real	Specific heat of char	kJ/kg/K	0.7
DELTA	Real	Wall thickness	m	0.1
DENSITY	Real	Solid mass per unit volume	kg/m^3	450.
E	Real	Activation energy	kJ/kmol	
E_COEFFICIENT	Real	Extinguishing coefficient	1/s	0.
EMISSIVITY	Real	Emissivity		1.
FUEL_FRACTION	Real	Mass concentration of fuel	kg/kg	1.
HEAT_FLUX	Real	Heat flux at surface	kW/m^2	0.
HEAT_OF_COMBUSTION	Real	Heat of combustion	kJ/kg	
HEAT_OF_VAPORIZATION	Real	Heat of vaporization	kJ/kg	
HRRPUA	Real	Heat Release Rate Per Unit Area	kW/m^2	0.
HRRPUA_MAX	Real	Maximum HRRPUA	kW/m^2	
ID	Character	Surface IDentifier		
KS	Real	Thermal conductivity	W/m/K	
KS_CHAR	Real	Thermal conductivity of char	W/m/K	0.1
LEAKING	Logical	Leakage calculation		.FALSE.
MASS_FLUX(I)	Real Array	Mass flux of species I		
MASS_FLUX_CRITICAL	Real	Mass flux at TMPIGN		0.02
MASS_FRACTION(I)	Real Array	Mass fraction of species I		
MOISTURE_FRACTION	Real	Water content by mass	kg/kg	0.
NPPC	Integer	Number of particles per cell		1
PART_ID	Character	Lagrangian Particle ID		
PHASE	Character	Phase of boundary		'SOLID'
POROSITY	Real	Porosity fraction		0.
PLE	Real	Atmospheric profile exponent		0.3
PROFILE	Character	Name of velocity profile		
RAMP_C_P	Character	Ramp ID for specific heat		
RAMP_C_P_CHAR	Character	Ramp ID for char specific heat		
RAMP_KS	Character	Ramp ID for conductivity		
RAMP_KS_CHAR	Character	Ramp ID for char conductivity		
RAMP_MF(I)	Character	Ramp ID for species I		
RAMP_Q	Character	Ramp ID for HRR		
RAMP_V	Character	Ramp ID for velocity		

SURF (Surface Properties, continued)				
RGB	Real Triplet	Color indices for surface		1.0,0.8,0.4
SURFACE_DENSITY	Real	Density × Thickness	kg/m^2	
TAU_MF(I)	Real Array	Ramp-up time for species I	s	1.
TAU_Q	Real	Ramp-up time for HRR	s	1.
TAU_V	Real	Ramp-up time for velocity	s	1.
TEXTURE_HEIGHT	Real	Height of texture image	m	1.
TEXTURE_MAP	Character	Name of texture map file		
TEXTURE_WIDTH	Real	Width of texture image	m	1.
TMPIGN	Real	Ignition temperature	°C	5000.
TMPWAL	Real	Surface temperature	°C	20.
TMPWAL0	Real	Initial solid temperature	°C	20.
VBC	Real	Velocity Slip Index		0.5
VEL	Real	Normal velocity	m/s	0.
VEL_T	Real Pair	Tangential velocity components	m/s	0.
VOLUME_FLUX	Real	Normal velocity x vent area	m^3/s	0.
WALL_POINTS	Integer	Internal wall points		20
Z0	Real	Atmospheric profile origin	m	10.

THCP (Thermocouples and Point Measurements)				
DEPTH	Real	Depth into surface for internal wall temp.	m	0
DIAMETER	Real	Diameter of Thermocouple Bead	m	0.001
DTSAM	Real	Time increment	s	TWFIN/NFRAMES
EMISSIVITY	Real	Emissivity of Thermocouple Bead		0.85
IOR	Integer	Orientation Parameter		
K_HIGH	Integer	Upper K value for LAYER HEIGHT		KBAR
K_LOW	Integer	Lower K value for LAYER HEIGHT		1
LABEL	Character	Identifier for output		
QUANTITY	Character	Name of Quantity to save		
XB	Real Sextuplet	Coordinates of non-point measurement	m	
XYZ	Real Triplet	Coordinates of point measurement	m	

TIME (Time Parameters)				
DT	Real	Initial time step	s	
TWFIN	Real	Time When FINished	s	1.

TRNX, TRNY, TRNZ (Transformation Parameters)				
CC	Real	Computational coordinate	m	
IDERIV	Integer	Order of polynomial transformation		
MESH_NUMBER	Integer	Number of mesh to transform		
PC	Real	Physical coordinate or derivative		

VENT (Vent Parameters)				
CB	Character	Coordinate Plane		
HEAT_ACTIVATE	Character	Name of Controlling Heat Detector		
HEAT_CLOSE	Character	Name of Controlling Heat Detector		
HEAT_DEACTIVATE	Character	Name of Controlling Heat Detector		
HEAT_OPEN	Character	Name of Controlling Heat Detector		
OUTLINE	Logical	Draw vent as outline		.FALSE.
PBX, PBY, PBZ	Real	Coordinate Plane		
RGB	Real Triplet	Color indices		
SURF_ID	Character	Associated Surface		
T_ACTIVATE	Real	Same as T_OPEN	s	
T_CLOSE	Real	Time to Close or De-activate Vent	s	
T_DEACTIVATE	Real	Same as T_CLOSE	s	
T_OPEN	Real	Time to Open or Activate Vent	s	
TEXTURE_ORIGIN	Real Triplet	Coordinates of Texture Map		
COLOR	Character	Color		
XB	Real Sextuplet	Physical coordinates	m	

Appendix C

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.

C.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`. Often, a quantity called the `Fire Resolution Index` appears. This is an indicator of how well resolved the calculation is – it is the fraction of the ideal stoichiometric value of the mixture fraction that is being used in the calculation. 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.

C.2 Plot3D Data

Presently, field data is output every `DTSAM` seconds 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 `PL3D` line, then the grid data is written out to a file called **CHID.xyz**

```
      WRITE(LU13) IBAR+1,JBAR+1,KBAR+1
      WRITE(LU13) (((X(I),I=0,IBAR),J=0,JBAR),K=0,KBAR),
     .            (((Y(J),I=0,IBAR),J=0,JBAR),K=0,KBAR),
     .            (((Z(K),I=0,IBAR),J=0,JBAR),K=0,KBAR),
     .     (((IBLK(I,J,K),I=0,IBAR),J=0,JBAR),K=0,KBAR)
```

where `X, Y` and `Z` are the coordinates of the cell corners, and `IBLK` is an indicator of whether or not the cell is blocked. If the point (`X,Y,Z`) is completely embedded within a solid region, then `IBLK` is 0. Otherwise, `IBLK` is 1. Normally, the grid file is not dumped.

The flow variables are written to a file called **CHID_****_**.q**, where the stars indicate a time at which the data is output. The file is written with the lines

```
      WRITE(LU14) IBAR+1,JBAR+1,KBAR+1
      WRITE(LU14) ZERO,ZERO,ZERO,ZERO
      WRITE(LU14) ((((QQ(I,J,K,N),I=0,IBAR),J=0,JBAR),K=0,KBAR),N=1,5)
```

The five channels `N=1,5` are by default the temperature (°C), the *u*, *v* and *w* components of the velocity (m/s), and the heat release rate per unit volume (kW/m^3). Alternate variables can be specified with the input parameter `QUANTITIES` under the `PL3D` namelist group. 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 grid.

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].

[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.

C.3 Thermocouple Data

Temperature (or other scalar quantity) data at discrete points specified in the input file under the namelist group `THCP` is output in comma delimited format in a file called **CHID_tc.csv**. The format of the file is as follows

```
NTC
TIME , LABEL(1)    , LABEL(2)    , ... , LABEL(NTC)
TIME , QUANTITY(1) , QUANTITY(2) , ... , QUANTITY(NTC)
s    , UNITS(1)    , UNITS(2)    , ... , UNITS(NTC)
T(1) , TC(1,1)     , TC(2,1)     , ... , TC(NTC,1)
T(2) , TC(1,2)     , TC(2,2)     , ... , TC(NTC,2)
                       .
                       .
                       .
```

where `NTC` is the number of thermocouples, `LABEL(I)` is the user-defined label of the `I`th thermocouple, `QUANTITY(I)` is the physical quantity represented, `UNITS(I)` the units, `T(J)` the time of the `J`th dump, and `TC(I,J)` the value at the `I`th thermocouple at the `J`th time. The files can be imported into Microsoft Excel or almost any other spread sheet program.

C.4 Sprinkler Data

If sprinklers are prescribed in the calculation, then a file called **CHID_spk.csv** is generated. The file lists the temperature of each sprinkler link. The format of the file is

```
NSPR
TIME , TEMP    , TEMP    , ... , TEMP
s    , C       , C       , ... , C
T(1) , TL(1,1) , TL(2,1) , ... , TL(NSPR,1)
T(2) , TL(1,2) , TL(2,2) , ... , TL(NSPR,2)
                 .
                 .
                 .
```

where `NSPR` is the number of sprinklers, `T(J)` is the time of the `J`th dump, `TL(I,J)` is the link temperature of the `I`th sprinkler at the `J`th time. The data is ordered the same way as the sprinklers are listed in the input file.

C.5 Heat Release Rate

Quantities associated with the overall energy budget are reported in the comma delimited file **CHID_hrr.csv**. 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 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.

The radiative loss to the boundaries can be interpreted as either a volume or boundary integral

$$\dot{Q}_r = \int \nabla \cdot \mathbf{q}_r \, dV = \int \mathbf{q}_r \cdot d\mathbf{S} = \int \dot{q}_r'' \, dA \tag{C.1}$$

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

$$\dot{Q}_{conv} = \int c_p \, \rho \, (T - T_\infty) \, \mathbf{u} \cdot d\mathbf{S} \tag{C.2}$$

where the integral is positive if the flow is out of the domain. The conductive loss to the solid surface is given by

$$\dot{Q}_{cond} = \int \dot{q}_r'' + \dot{q}_c'' \, dA \tag{C.3}$$

where the integral is positive if heat is being lost into a wall colder than the gas. See the Technical Reference Guide for more details.

C.6 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.

C.7 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 rest the mass fractions of the various gases.

C.8 Slice Files

The slice files defined under the namelist group `SLCF` are named **CHID_*n*.sf** (*n*=01,02...), and are written out unformatted, unless otherwise directed. These files are written out from **dump.f** with the following lines:

```
WRITE(LUSF) QUANTITY
WRITE(LUSF) SHORT_NAME
WRITE(LUSF) UNITS
WRITE(LUSF) I1,I2,J1,J2,K1,K2
WRITE(LUSF) TIME
WRITE(LUSF) (((QQ(I,J,K),I=11,I2),J=J1,J2),K=K1,K2)
        .
        .
        .
WRITE(LUSF) TIME
WRITE(LUSF) (((QQ(I,J,K),I=11,I2),J=J1,J2),K=K1,K2)
```

QUANTITY, SHORT_NAME and UNITS are character strings of length 30. The sextuplet (I1,I2,J1,J2,K1,K2) denotes the bounding grid cell nodes. The sextuplet indices correspond to grid cell nodes, or corners, thus the entire grid would be represented by the sextuplet (0,IBAR,0,JBAR,0,KBAR).

There is a short Fortran 90 program provided, called **fds2ascii.f**, that can convert slice files into text files that can be read into a variety of graphics packages. The program combines multiple slice files corresponding to the same "slice" of the computational domain, time-averages the data, and writes the values into one file, consisting of a line of numbers for each node. Each line contains the physical coordinates of the node, and the time-averaged quantities corresponding to that node. In particular, the graphics package Tecplot reads this file and produces contour, streamline and/or vector plots. See Section 4.7.6 for more details about the program **fds2ascii**.

C.9 Boundary Files

The boundary files defined under the namelist group BNDF are named **CHID_*n*.bf** (*n*=01,02...), and are written out unformatted. These files are written out from **dump.f** with the following lines:

```
WRITE(LUBF) QUANTITY
WRITE(LUBF) SHORT_NAME
WRITE(LUBF) UNITS
WRITE(LUBF) NPATCH
WRITE(LUBF) I1,I2,J1,J2,K1,K2,IOR
WRITE(LUBF) I1,I2,J1,J2,K1,K2,IOR
        .
        .
        .
WRITE(LUBF) TIME
WRITE(LUBF) (((QQ(I,J,K),I=11,I2),J=J1,J2),K=K1,K2)
WRITE(LUBF) (((QQ(I,J,K),I=11,I2),J=J1,J2),K=K1,K2)
        .
        .
        .
WRITE(LUBF) TIME
WRITE(LUBF) (((QQ(I,J,K),I=11,I2),J=J1,J2),K=K1,K2)
WRITE(LUBF) (((QQ(I,J,K),I=11,I2),J=J1,J2),K=K1,K2)
        .
        .
        .
```

QUANTITY, SHORT_NAME and UNITS are character strings of length 30. NPATCH is the number of planes (or "patches") that make up the solid boundaries plus the external walls. The sextuplet (I1,I2,J1,J2,K1,K2) defines the cell nodes of each patch. IOR is an integer indicating the orientation of the patch ($\pm 1, \pm 2, \pm 3$). The user does not prescribe these. Note that the data is planar, thus one pair of cell nodes is the same.

Presently, Smokeview is the only program available to view the boundary files.

C.10 Particle Data

The tracer particles and sprinkler droplets are animated using a file called **CHID.part**. It contains the particle positions. There is a one line header followed by particle locations output every DTSAM seconds.

The structure of the data file is given in the file **dump.f** by a subroutine called PDMP3D.

```
WRITE(LU10) ARX,ARY,DTSAM,IPART,NPPS
WRITE(LU10) IBAR,JBAR,KBAR
WRITE(LU10) (X(I),I=0,IBAR),(Y(J),J=0,JBAR),(Z(K),K=0,KBAR)
WRITE(LU10) NB
DO N=1,NB
WRITE(LU10) IB1(N),IB2(N),JB1(N),JB2(N),KB1(N),KB2(N),1
END DO
WRITE(LU10) NV
DO N=1,NV
WRITE(LU10) IV1(N),IV2(N),JV1(N),JV2(N),KV1(N),KV2(N),2
END DO
WRITE(LU10) NSPR
DO N=1,NSPR
WRITE(LU10) XSP0(N),YSP0(N),ZSP0(N)
END DO
```

where ARX=ZBAR/XBAR and ARY=ZBAR/YBAR are the aspect ratios of the overall domain, IPART is the index of the scalar quantity associated with the particles, NPPS is the maximum number of particles per frame, IB1, IB2, *etc.* are the indices of blocked grid cells, IV1, IV2, *etc.* indicate vent cell nodes, and XSP0, YSP0, ZSP0 are the coordinates of the sprinklers. The arrays X(I), Y(J), Z(K) contain the coordinates of the grid. Every DTSAM seconds the coordinates of the tracer particles and sprinkler droplets are output

```
 WRITE(LU10) TIME,NP,NN,(ISPR(N),N=1,NSPR)
 WRITE(LU10) (XP(I),I=1,NP),
.            (YP(I),I=1,NP),
.            (ZP(I),I=1,NP),
.            (BP(I),I=1,NP)
 IF (NASPR.GT.0) THEN
 WRITE(LU10) NSP
 WRITE(LU10) (XSP(I),I=1,NSP),(YSP(I),I=1,NSP),(ZSP(I),I=1,NSP)
 ENDIF
```

where NP is the number of tracer particles, NN is a dummy integer kept for compatibility reasons, ISPR denotes whether the sprinkler has activated, NSPR is the number of sprinklers, XP, YP, ZP are the coordinates of the element, BP is a the quantity referenced by IPART, NASPR is the number of active sprinklers, NSP is the number of sprinkler droplets, and XSP, YSP, ZSP are the droplet coordinates.