

**NIST Technical Note 1889v5**

**CFAST – Consolidated Fire  
and Smoke Transport  
(Version 7)**

**Volume 5: CFAST Fire Data Generator  
(CData)**

Paul A. Renke  
Richard D. Peacock  
Stanley W. Gilbert  
Thomas G. Cleary

This publication is available free of charge from:  
<http://dx.doi.org/10.6028/NIST.TN.1889v5>





# NIST Technical Note 1889v5

## CFAST – Consolidated Fire And Smoke Transport (Version 7)

### Volume 5: CFAST Fire Data Generator (CData)

Paul A. Reneke  
Richard D. Peacock  
Thomas G. Cleary

*Fire Research Division, Engineering Laboratory, Gaithersburg, Maryland*

Stanley W. Gilbert

*Office of Economics, Engeneering Laboratory, Gaithersburg, Maryland*

This publication is available free of charge from:  
<http://dx.doi.org/10.6028/NIST.TN.1889v5>

March 2021  
CFAST Version 7.6.0  
*GIT Revision:* unknown



U.S. Department of Commerce  
*Gina M. Raimondo, Secretary*

National Institute of Standards and Technology

*James K. Olthoff, Performing the Non-Exclusive Functions and Duties of the Under Secretary of Commerce  
for Standards and Technology & Director, National Institute of Standards and Technology*



# Preface

This document provides documentation for creating and running Monte Carlo simulations with the Consolidated Fire And Smoke Transport (CFAST) model using the CFAST Fire Data Generator (CData). The method follows the general framework set forth in the “Standard Guide for Evaluating the Predictive Capability of Deterministic Fire Models,” ASTM E 1355 [1]. Instructions for using CFAST are contained in a separate user’s guide, and model assessment information is contained in a separate verification and validation guide.



# **Disclaimer**

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

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

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

**National Institute of Standards and Technology Technical Note 1889v5  
Natl. Inst. Stand. Technol. Tech. Note 1889v5, 75 pages (March 2021)  
CODEN: NTNOEF**

**This publication is available free of charge from:  
<http://dx.doi.org/10.6028/NIST.TN.1889v5>**



# Acknowledgments

CFAST was originally developed by Walter Jones, formerly of NIST.

Continuing support for CFAST is via internal funding at NIST. In addition, support is provided by other agencies of the U.S. Federal Government, most notably the Nuclear Regulatory Commission (NRC) and the Department of Energy (DOE). The NRC Office of Research has funded key validation experiments, the preparation of the CFAST manuals, and the continuing development of sub-models that are of importance in the area of nuclear power plant safety. Special thanks to Mark Salley and David Stroup for their support. Support to refine the software development and quality assurance process for CFAST has been provided by the DOE. The assistance of Subir Sen and Debra Sparkman is gratefully acknowledged.

We also thank Dr. Wai Cheong Tam, Dr. Justin Williamson, and Nelson Bryner for their reviews and corrections. Any remaining errors are ours.



# Contents

<b>Preface</b>	<b>iii</b>
<b>Disclaimer</b>	<b>v</b>
<b>Acknowledgments</b>	<b>vii</b>
<b>1 Getting Started</b>	<b>1</b>
1.1 Installation . . . . .	2
1.2 Defining the Question and the Analysis . . . . .	2
1.2.1 Building Characteristics . . . . .	3
1.2.2 Fire Scenarios . . . . .	8
<b>2 Defining Data for Analysis, CData Inputs</b>	<b>9</b>
2.1 Namelist MHDR . . . . .	9
2.2 Namelist MRND . . . . .	10
2.3 Namelist MFLD . . . . .	12
2.4 Namelist MFIR . . . . .	15
2.5 Namelist MSTT . . . . .	20
2.6 Namelist DUMP . . . . .	21
<b>3 Creating Multiple CFAST Runs</b>	<b>23</b>
3.1 PreProcessor . . . . .	23
3.2 Running CFAST . . . . .	25
3.3 Generating Statistics . . . . .	26
3.3.1 Accumulator . . . . .	26
3.3.2 Statistics . . . . .	26
<b>4 Examples</b>	<b>33</b>
4.1 Example 1: Flashover in a compartment . . . . .	33
4.1.1 CData Inputs . . . . .	34
4.1.2 Results . . . . .	35
4.2 Example 2: Impact of smoke detector interconnection . . . . .	36
4.2.1 CData Inputs . . . . .	37
4.2.2 Number of Cases . . . . .	40
4.2.3 Analysis . . . . .	41
4.2.4 Number of Cases - Revisited . . . . .	43

4.2.5	Discussion . . . . .	44
4.3	Example 3: Model Sensitivity . . . . .	45
4.3.1	CData Inputs . . . . .	45
4.3.2	Approach . . . . .	48
4.3.3	Number of Cases . . . . .	48
4.3.4	Discussion . . . . .	49
<b>5</b>	<b>Summary</b>	<b>51</b>
<b>References</b>		<b>53</b>
<b>Appendices</b>		<b>55</b>
<b>A</b>	<b>Nomenclature</b>	<b>57</b>
<b>B</b>	<b>CData/CFAST Text-based Input File</b>	<b>59</b>
B.1	Naming the Input File . . . . .	59
B.2	Namelist Formatting . . . . .	59
B.3	Input File Structure . . . . .	60
B.4	Monte Carlo Header, Namelist Group MHDR . . . . .	63
B.5	Monte Carlo Random Input Generators, Namelist Group MRND . . . . .	64
B.6	Monte Carlo Input Field Generators, Namelist Group MFLD . . . . .	65
B.7	Monte Carlo Input Fire Generators, Namelist Group MFIR . . . . .	69
B.8	Monte Carlo Statistical Plots, Namelist Group MSTT . . . . .	73
<b>C</b>	<b>Running CData from a Command Prompt</b>	<b>75</b>
<b>D</b>	<b>R Scripts for Example Cases</b>	<b>77</b>
D.1	R Script for Interconnected Alarms . . . . .	77
D.2	R Script for Sensitivity . . . . .	83

# List of Figures

1.1	Sample CFAST visualization of a single structure subject to a fire. . . . .	3
1.2	Cumulative Probability Distributions for Home Size, Number of Bedrooms and Total Number of Rooms . . . . .	7
3.1	Sample of Histogram generated for the 'Width of Vent' column . . . . .	28
3.2	Sample of Convergence of mean for maximum upper layer temperature . . . . .	29
3.3	Sample of Convergence of mean for maximum upper layer temperature . . . . .	30
3.4	Sample of Decision Tree for maximum upper layer temperature . . . . .	31
4.1	Sample CFAST visualization of a single compartment structure used for example 1.	33
4.2	Comparison of correlations, CFAST predictions, and experimental data for the prediction of flashover in a compartment fire. . . . .	36
4.3	Sample CFAST visualization of a single story residential structure used for example 2. . . . .	37
4.4	Mean percent activation and time to activation for the bedroom 1 alarm as a function of the number of cases. This is a split between flaming (left column) and smoldering (right column) fires showing precent of cases where detector activated (top row) and time to activation (bottom row) . . . . .	40
4.5	Standard deviation of percent activation and time to activation for the bedroom 1 alarm as a function of the number of cases. This is a split between flaming (left column) and smoldering (right column) fires showing precent of cases where detector activated (top row) and time to activation (bottom row) . . . . .	41
4.6	Decision tree separating cases where the bedroom alarm sounded from those where it did not. . . . .	43
4.7	Kernel Density estimate of the time savings for interconnected alarms by fire type.	43
4.8	Quantiles of the time savings as a function of the number of cases. . . . .	44
4.9	Sample CFAST visualization of a single story commercial structure used for example 3. . . . .	46
4.10	Convergence by the number of runs. . . . .	49

# List of Tables

1.1	CFAST Inputs That Can be Varied Based on User-Defined Distributions . . . . .	4
1.2	CFAST Fire Inputs That Can be Varied Based on User-Defined Distributions . . . . .	8
1.3	CFAST Fire Time Histories That Can be Varied Based on One or More User-Defined Distributions . . . . .	8
4.1	Number of cases by alarm activation status. . . . .	42
4.2	Quantiles of delay time for bedroom alarm activation . . . . .	44
4.3	Selected result for sensitivity of maximum temperature FED. . . . .	50
4.4	Selected results for sensitivity of time to non-viability for Foyer Heat FED. . . . .	50
B.1	CFAST Input File Keywords . . . . .	62
B.2	Monte Carlo Header Parameters ( <code>MHDR</code> namelist group) . . . . .	63
B.3	Monte Carlo Random Generators ( <code>MRND</code> namelist group) . . . . .	64
B.4	Monte Carlo Input Field Generators ( <code>MFLD</code> namelist group) . . . . .	65
B.5	<code>&amp;MFLD</code> Inputs That Can be Varied Based on User-Defined Distributions . . . . .	66
B.6	Monte Carlo Fire Generators ( <code>MFIR</code> namelist group) . . . . .	70
B.7	Monte Carlo Statistics Plots ( <code>MSTT</code> namelist group) . . . . .	73

# Chapter 1

## Getting Started

The Monte Carlo method is "a broad class of computational algorithms that use repeated random sampling to obtain numerical results." [2] Stanislaw Ulam is credited with inventing the modern version of the Markov Chain Monte Carlo method while working on nuclear weapons projects in the 1940s [3]. Its application in fire safety dates to at least the 1980s with research ongoing to better understand equivalence between the then standard prescriptive designs and more performance-based designs. Bukowski [4] suggested using the Monte Carlo method as a means of demonstrating when alternative designs were as safe or safer than prescriptive code compliant designs. The basic idea was to take a design made to meet code and compare its fire safety performance to an alternative design using a set of scenarios with a range of fires appropriate for the occupancy. If the relative performance of the alternative design was as good or better than the code compliant design, that would be justification for having the alternative design approved. Clarke et al. [5], as part of demonstrating the viability of using a computer model to predict fire safety, used a Monte Carlo analysis using the Consolidated Fire and Smoke Transport (CFAST) model to predict fire statistics for a residential application. The process continued and now building codes [6] and engineering handbooks [7] provide a legal and technical structure for a fire performance-based analysis using, in part, Monte Carlo methods to characterize the relative fire hazard performance of designs.

Work continued to better characterize the Monte Carlo method in its use in fire safety research and design. Notarianni [8] used Monte Carlo methods to characterize and quantify uncertainty in performance-based designs for fire safety. Bruns [9], in a study using Monte Carlo methods to assess the impact of inner liners in residential upholstered furniture, formalized the mathematics for applying the Monte Carlo method to fire hazard analysis as a means to further incorporate the method in regular performance-based designs.

At one time issues of both insufficient computational power and a lack of tools designed to do the analysis were obstacles to using Monte Carlo methods. However, it is now relatively easy to obtain the computational power and storage to generate and analyze huge amounts of data. What is still largely lacking are the tools to make the process tenable. To that end the Fire Research Division at the National Institute of Standards and Technology (NIST) has been exploring the process [10, 11, 12, 13] in order to develop tools that will make Monte Carlo fire hazard analysis a more widely used form of analysis using the Consolidated Fire And Smoke Transport (CFAST) model. The result of this effort is the CFAST Fire Data Generator (CData), documented in this report.

The fire model being used, CFAST, is documented by four publications, a user's guide [14], a

technical reference guide [15] a verification and validation guide [16], and a configuration management guide [17]. The user's guide describes how to use the model and the input editor CEdit. The technical reference guide describes the underlying physical principles, provides a comparison with other models, and includes an evaluation of the model following the guidelines of ASTM E1355 [1]. The model verification and validation guide documents verification and validation efforts for the model. The configuration management guide documents the processes used during the development and validation of the model. This guide is a companion to the CFAST software and describes the use of the program CData to generate, summarize, and analyze numerous CFAST simulations.

## 1.1 Installation

The CFAST distribution consists of a self-extracting set-up program for Windows-based personal computers. After downloading the set-up program, double-clicking on the file's icon walks you through a series of steps for installation of the program. The most important part of the installation is the creation of a folder (C:\Program Files\CFAST by default) in which the CFAST executable files and supplemental data files are installed. Sample input files are found in the Examples folder. CData is installed as part of the CFAST distribution.

CData also makes use of the statistical software, R, for selected analyses of the data generated for multiple CFAST runs. R can be downloaded from <https://www.r-project.org/>.

## 1.2 Defining the Question and the Analysis

As briefly discussed in the introduction significant research has gone into understanding the basic requirements for a Monte Carlo analysis in a fire safety analysis [4, 5, 6, 7, 9]. Several key areas that need to be addressed in the analysis include definition of:

1. Community / Building / Occupant characteristics
2. Fire scenarios
3. Analysis variables / Criteria for comparisons
4. Statistical analysis of calculation results

Community, building, and fire characteristics define the physical geometries of the model simulations (the range of building geometries, vents between different compartments and compartments to the outside, and the range and position of fires to be studied). Occupant characteristics and criteria for comparisons define additional model inputs that may be necessary for analysis of calculation results (fire detection devices, the choice of additional model outputs to characterize tenability along egress paths, fire severity, or building structural integrity, for example). For the most part, community and occupant characteristics are outside the scope of this report which focuses on developing the range of CFAST model inputs needed to run the desired set of simulations. These may be defined by a single deterministic set of inputs (a single building geometry or desired fire for study), a collection of different, specific inputs (such as a set of specific building designs

of interest), or a statistically-determined range of inputs (for example, defining ranges of compartment sizes or smoke detector activation from experimentally-determined distributions). This section details the process for defining a series of input files for analysis with examples for each major step in the process.

### 1.2.1 Building Characteristics

In CFAST, compartment geometry includes definition of the number of compartments, their size (length, width, and height), and their placement in relation to other compartments. In the study of a single structure, this is simply an enumeration of each compartment. Figure 1.1 shows the compartments in a single structure in a CFAST simulation.

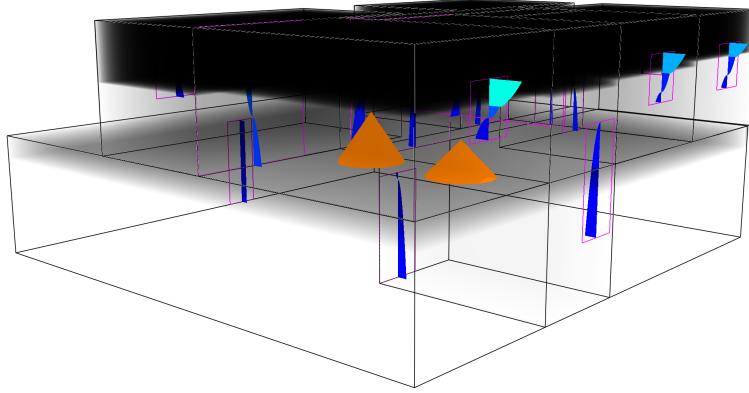


Figure 1.1: Sample CFAST visualization of a single structure subject to a fire.

Of course, if it is desired to study the impact of fires in a set of more than one specific building, the compartment geometry and placement could be defined by multiple individual buildings with the specific building chosen for an individual scenario chosen at random or from a distribution representing the population of each building type in the community under study.

The set of buildings for study can also be chosen from distributions of building and room characteristics. For example Figure 1.2 shows the distribution of the total floor area in a residence, the number of bedrooms in a residence depending on the total floor area, the total number of rooms given a certain number of bedrooms (here shown for residences from  $1000 \text{ ft}^2$  to  $1500 \text{ ft}^2$ ), all taken from the 2015 U.S. Housing Survey [18]. Creating a compartment geometry from these data can be thought of as a six step process:

1. randomly select the total floor area of the structure, Fig. 1.2(a);
2. randomly select the total number of bedrooms for a structure of the size chosen in step 1, Fig. 1.2(b);
3. randomly select the total number of rooms in the structure of the chosen size and number of bedrooms (Fig. 1.2(c) shows a sample distribution for homes ranging from  $1000 \text{ ft}^2$  to  $1500 \text{ ft}^2$ . Distributions for other home sizes are available in ref. [18]);

4. determine room sizes based on a distribution of bedroom sizes, allocating left over space to the other rooms;
5. connect compartments as desired (for example by randomly setting vents as open or closed between compartment pairs); and
6. ensure that the resulting structure is realizable (This random approach to generating connections has a probability of resulting in a floorplan that cannot be instantiated in a single story. More technically, if the floorplan is thought of as a graph with the rooms as vertices and the connections as edges, some of the randomly generated graphs will be nonplanar for cases with more than four rooms. A planar graph is one that can be drawn on a piece of paper and none of the edges cross. The probability of generating a nonplanar floorplan increases as the number of rooms increases. In order to eliminate such nonphysical cases from the analysis, any randomly generated floorplan can be checked for planarity and rejected if necessary and replaced by a new randomly generated floorplan).

Other characteristics of the structure such as materials of construction, vent openings, fire definitions, measurement targets, sprinklers, and detection devices can be varied as desired for the problem being studied. Table B.5 shows variables in the modeling that can be varied based on user-defined distributions <sup>1</sup>.

Table 1.1: CFAST Inputs That Can be Varied Based on User-Defined Distributions

<b>Category</b>	<b>Input</b>	<b>Units</b>
Ambient Conditions	Interior Temperature Exterior Temperature Relative Humidity	°C °C %
Thermal Properties	Thermal Conductivity Specific Heat Density Default Thickness Emissivity	kW/(m °C) kJ/(kg °C) kg/m <sup>3</sup> m
Compartments	Width Depth Height Wall Leak Area Ratio Floor Leak Area Ratio Compartment Materials	m m m m <sup>2</sup> /m <sup>2</sup> m <sup>2</sup> /m <sup>2</sup>
Wall Vents	Sill Soffit Width Open/Close Times Open/Close Fractions Initial Opening Fraction	m m m s 0-1 0-1

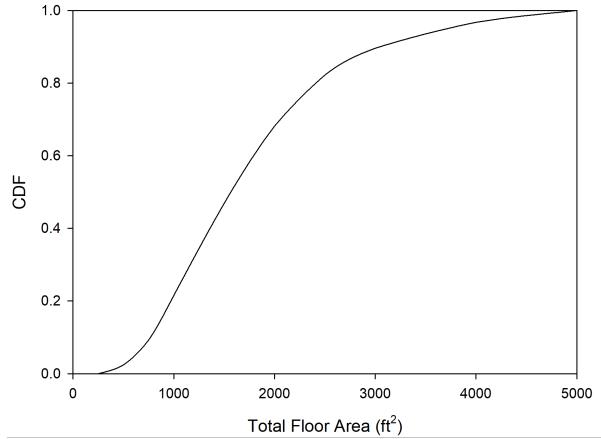
<sup>1</sup>The content of this table follows the form of that used in B-Risk [19], adapted to be applicable to the calculational capabilities of the CFAST model.

Table 1.1: Continued

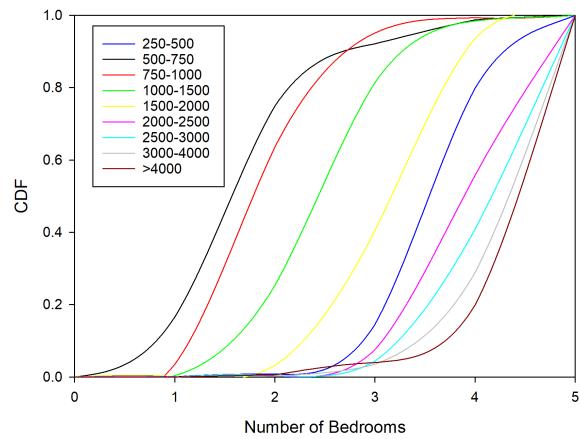
<b>Category</b>	<b>Input</b>	<b>Units</b>
	Open/Close Time Final Opening Fraction Setpoint Pre-Activation Fraction Post-Activation Fraction	s 0-1 s, °C, or kW/m <sup>2</sup> 0-1 0-1
Ceiling / Floor Vents	Cross-Sectional Area Open/Close Times Open/Close Fractions Initial Opening Fraction Open/Close Time Final Opening Fraction Setpoint Pre-Activation Fraction Post-Activation Fraction	m <sup>2</sup> s 0-1 0-1 s 0-1 s, °C, or kW/m <sup>2</sup> 0-1 0-1
Mechanical Vents	From Compartment Area From Compartment Height To Compartment Area To Compartment Height Flow Rate Begin Dropoff End Dropoff Open/Close Times Open/Close Fractions Initial Opening Fraction Open/Close Time Final Opening Fraction Setpoint Pre-Activation Fraction Post-Activation Fraction Filter Efficiency Begin Filtering Time	m <sup>2</sup> m m <sup>2</sup> m m <sup>3</sup> /s Pa Pa s 0-1 0-1 s 0-1 s, °C, or kW/m <sup>2</sup> 0-1 0-1 % s
Fires	See Section 1.2.2	
Targets	Width Target Position Depth Target Position Height Target Position Width Normal Vector Depth Normal Vector Height Normal Vector Target Points To Thickness Internal Temperature Location	m m m 0-1 0-1 0-1 Selection List m m
Detection / Suppression	Width Position	m

Table 1.1: Continued

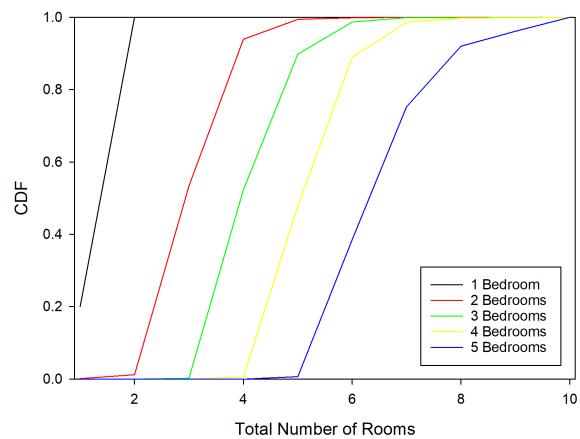
<b>Category</b>	<b>Input</b>	<b>Units</b>
	Depth Position	m
	Height Position	m
	Activation Temperature	°C
	Activation Obscuration	%/m
	RTI	(m s) <sup>1/2</sup>
	Spray Density	m/s



(a) Total Floor Area in a Residence



(b) Number of Bedrooms in a Residence as a Function of Total Floor Area



(c) Total Number of Rooms as a Function of the Number of Bedrooms in a Residence with 1000 ft<sup>2</sup> - 1500 ft<sup>2</sup> of Total Floor Area

Figure 1.2: Example Cumulative Probability Distributions for Home Size, Number of Bedrooms and Total Number of Rooms (excluding Bathrooms) taken from the 2015 U.S. Housing Survey [18]

## 1.2.2 Fire Scenarios

### Individual Variables

The quantitative definition of fires is arguably the most important [20] and complex of all the inputs in any fire modeling scenario. It includes specification of the fire location, fuel composition, and ignition criterion. Heat release rate, burning area, and species yields, which can vary with time over the course of the fire are covered in the next section.

Table 1.2: CFAST Fire Inputs That Can be Varied Based on User-Defined Distributions

Category	Input	Units
Fire Location	Compartment Width Position Depth Position	Selection List m m
Fuel Composition	Carbon Molecules Hydrogen Molecules Oxygen Molecules Nitrogen Molecules Chlorine Molecules Heat of Combustion Radiative Fraction	$\geq 0$ $\geq 0$ $\geq 0$ $\geq 0$ $\geq 0$ kJ/kg 0 - 1
Ignition Criteria	Ignition Criterion Setpoint	Selection List s, °C, or kW/m <sup>2</sup>
Time Histories	See Section 1.2.2	

### Time Histories

Key to the fire definition are the time histories of heat release rate, burning area, and species yields of important combustion products. In some scenarios, these may be constants, but in others, they can vary with time.

Table 1.3: CFAST Fire Time Histories That Can be Varied Based on One or More User-Defined Distributions

Category	Input	Units
Time Histories	Time HRR Fire Height Fire Area CO Yield Soot Yield HCN Yield	s kW m m <sup>2</sup> , >0 kg CO/kg fuel kg Soot/kg fuel kg HCN/kg fuel

# Chapter 2

## Defining Data for Analysis, CData Inputs

Choosing modeling results for analysis depends on the goals of the hazard analysis and the particular technology under study. In the most general sense, this means understanding what results are indicative of improved fire safety. To improve fire safety generally means reducing the deaths and injuries due to a fire. So, the goal is to identify which variables, that can be calculated, would give an indication that a new technology will reduce deaths and injuries.

Each subsection will discuss one set of namelist inputs and what each of the parameters do. All the inputs are combined in Appendix B to serve as a reference for commands.

Section 3.1 has additional details on how to run CData. One example of running CData from a command line is the following

```
cdata Simple.in -P
```

In this case the base file name of the CData/CFAST input file, `Simple`, is referred to as the `<project>` and that name is used as part of the name of a number of different files for the series of calculations.

### 2.1 Namelist MHDR

The `&MHDR` inputs specify general inputs of the scenarios to be generated including the number of cases to be generated, seeds for the random number generator, and locations for input and output files.

**NUMBER\_OF\_CASES** (default value 1): specifies the number of cases that the preprocessor will generate.

**SEEDS** (default value, software chosen random integer pair): defines an integer pair used to determine random number seeds for distributions. Any two integers (excluding -1001 which is used internally to indicate default values) may be specified. Including random seeds here will ensure that the same cases will be generated each time the preprocessor is run for a given input file. Changing the random seeds (or other inputs) will result in a different set of input files. All seeds are written in the `<project>_seeds.csv` file, if specified.

**WRITE\_SEEDS** (default value, .TRUE.): if .TRUE., all random number seeds are saved in the file <project>\_seeds.csv.

**PARAMETER\_FILE** (default value, <project>\_parameters.csv): Summary output file from the preprocessor that list the CFAST file names and parameter values for inputs varied for each CFAST scenario in the set of generated CFAST cases. This file is combined with the summary statistics generated by the accumulator module.

**WORK\_FOLDER** (default value, current folder): folder where the preprocessor creates its set of CFAST inputs file and where the accumulator looks for CFAST output files to be processed.

**OUTPUT\_FOLDER** (default value, current folder): folder where the accumulator or statistics modules put their output files of analysis results.

Examples:

```
&MHDR NUMBER_OF_CASES = 10000 /
&MHDR NUMBER_OF_CASES = 20000 WRITE_SEEDS = .TRUE.
PARAMETER_FILE = 'Outputfile' WORK_FOLDER = 'Z:\' OUTPUT_FOLDER =
'..\project'/
```

## 2.2 Namelist MRND

The &MRND input defines a random number generator that uses a number of inputs to specify distributions for variation. A generator specifies a single random variable for each case so that every input using a particular generator will get the same value for each case. In the simplest cases every field that a user wants to vary will require a separate random generator. However, if the desire is to have all the rooms have the same height ceiling, then all the fields can use the same random number generator.

**ID** Distributions are defined by a unique alphanumeric name. This may be as simple as a single character or number, or a description of the distribution. All IDs must be unique throughout an input file. The ID can be any ASCII string up to 128 characters.

**FYI** A user defined comment that can provide additional information about the input.

**DISTRIBUTION\_TYPE** identifies the type of distribution. Additional required inputs depend of the type of distribution defined. Allowed distributions are CONSTANT, UNIFORM, TRIANGLE, NORMAL, TRUNCATE\_NORMAL, LOG\_NORMAL, TRUNCATED\_LOG\_NORMAL, BETA, LINEAR , and USER\_DEFINED.

**RANDOM\_SEEDS** like RANDOM\_SEEDS in &MHDR, an integer pair specifies initial seeds for the random number generator, but applies only to the specific distribution. Any two integers (excluding -1001 which is used internally to indicate default values) may be specified. Including

random seeds here will ensure that the same cases will be generated each time the preprocessor is run for a given input file. Changing the random seeds (or other inputs) will result in a different set of input files. All seeds are written in the <project>\_seeds.csv file, if specified.

**VALUE\_TYPE** defines the input type in VALUES. Must be INTEGER, REAL, or CHARACTER.

**REAL\_VALUES** a set of real numbers for the USER\_DEFINED distribution type with VALUE\_TYPE REAL. Inputs are input as real constants. Values for an individual scenario are chosen randomly from the list of values provided.

**INTEGER\_VALUES** a set of integer numbers for the USER\_DEFINED distribution type with VALUE\_TYPE INTEGER. Inputs are input as integer constants. Values for an individual scenario are chosen randomly from the list of values provided.

**STRING\_VALUES** a set of character strings for the USER\_DEFINED distribution type with VALUE\_TYPE CHARACTER. Inputs are input as ASCII character string constants. Values for an individual scenario are chosen randomly from the list of values provided.

**LOGICAL\_VALUES** a set of Fortran Boolean values (.TRUE. or .FALSE.) for the USER\_DEFINED distribution type with VALUE\_TYPE LOGICAL. Inputs are input as Boolean constants. Values for an individual scenario are chosen randomly from the list of values provided.

**PROBABILITIES** a set of probabilities values for the USER\_DEFINED distribution type with VALUE\_TYPE PROBABILITIES. These are not cumulative probabilities and the total of all values must add to 1.0

**REAL\_CONSTANT** is a single real value. For the REAL\_CONSTANT distribution with VALUE\_TYPE REAL, it is the value that is returned by the generator.

**INTEGER\_CONSTANT** is a single integer value. For the INTEGER\_CONSTANT distribution with VALUE\_TYPE INTEGER, it is the value that is returned by the generator.

**STRING\_CONSTANT** is a single string value. For the STRING\_CONSTANT distribution with VALUE\_TYPE CHARACTER, it is the value that is returned by the generator.

**MINIMUM** minimum value for the UNIFORM, TRIANGLE, TRUNCATED\_NORMAL, TRUNCATED\_LOG\_NORMAL, and LINEAR distribution types.

**MAXIMUM** maximum value for the UNIFORM, TRIANGLE, TRUNCATED\_NORMAL, TRUNCATED\_LOG\_NORMAL, and LINEAR distribution types.

**PEAK** peak value for the TRIANGLE distribution type.

**ALPHA** alpha value for the BETA distribution type.

**BETA** beta value for the BETA distribution type.

**MEAN** mean value for the NORMAL; geometric mean value for the LOG\_NORMAL distribution type.

**STDEV** standard deviation value for the `NORMAL`; geometric standard deviation value for the `LOG_NORMAL` distribution type.

**MINIMUM\_FIELD** specifies the ID of the specific `&MFLD` within the input whose value is to be taken as the minimum value of the distribution returned. It allows the range to be defined by another input. For example, if the `&MRND` input is used to vary the ceiling height the top of a vent can be used as the lowest value for the ceiling height by including a `MINIMUM_FIELD` input with the `ID` for the `FIELD` that sets the vent height input such as `MINIMUM_FIELD="Vent 1", "TOP"` to limit how low the height of the compartment is set.

**MAXIMUM\_FIELD** specifies the ID of the specific `&MFLD` within the input whose value is to be taken as the maximum value of the distribution returned. It allows the range to be defined by another input. For example, if the `&MRND` input is used to vary the ceiling height the top of a vent can be used as the lowest value for the ceiling height by including a `MAXIMUM_FIELD` input with the `ID` for the `FIELD` that sets the ceiling height input such as `MAXIMUM_FIELD="Comp 1", "HEIGHT"` to limit how high the top of the vent is set.

**ADD\_FIELD** specifies the ID of the specific `&MFLD` within the input whose value is to be added to the value generated by the current random generator. Suppose that a vent is to be closed at a random time, then a second random generator can be set as a constant 1 s and have the first `&MFLD` as the `STACK_FIELD` to have the time the vent finishes closing to be 1 s after the start.

Example:

```
&MRND ID = 'Example rand generator', TYPE = 'UNIFORM', MINIMUM = 10, MAXIMUM  
= 50 /  
&MRND ID = 'Second rand generator', TYPE = 'NORMAL', MEAN = 0, STDEV = 1 /
```

## 2.3 Namelist MFLD

The basic structure in CData to change values of a CFAST input file is the `&MFLD` input. Its basic function is to marry a random generator given in a `&MRND` namelist to a particular value in a CFAST input file. Depending on how many fields are to be changed in a single Monte Carlo analysis there could be a very large number of `&MFLD` inputs and unlike `&MRND` inputs, there is a one to one match between `&MFLD` inputs and CFAST entries to be changed.

**ID** Varied input fields are defined by a unique alphanumeric name. This may be as simple as a single character or number, or a description of the field. All IDs must be unique throughout an input file. The `ID` can be any ASCII string up to 128 characters.

**FYI** A user defined comment that can provide additional information about the input.

**FIELD\_TYPE** specifies the type of field data used to fill in values in the specified field. Additional inputs depend on the type of field specified. Allowed field types are `VALUE`, `SCALING`,

`LABEL`, and `INDEX`. `VALUE` simply takes the value returned from the specified random generator and places it directly in the specified field. `SCALING` takes the returned value and multiplies by the `BASE_SCALING_VALUE` and puts that value in the field. `INDEX` requires the random generator to return integers as an index into the `VALUES` input (starting at 1 to the number of inputs in the `VALUES` input for the current `&MFLD` input). The value in the `INDEX_TYPE`, `REAL`, `INTEGER`, `CHARACTER`, or `LOGICAL`, array at the index is placed in the field.

**FIELD** specifies the name of the specific input (i.e., the name of a compartment, vent, etc.) and the specific field within that input that is to be varied. For example, `FIELD="Comp 1", "HEIGHT"` would vary the height of the compartment named `Comp 1`.

**RAND\_ID** specifies the ID of associated `&MRND` input used to provide random inputs for the field. These may be unique to each field input or more than one field input may use the same `&MRND` input to coordinate values for multiple input fields. The number of inputs in the `&MRND` input must match those for the field type.

**PARAMETER\_COLUMN\_LABEL** specifies the column title for this input field in the parameters file, `<project>_parameters.csv`. If it is not included CData will use a combination of the `FIELD` inputs to create a column label.

**ADD\_TO\_PARAMETERS** set to `.TRUE.` to include the value of the field in the parameters file, `<project>_parameters.csv`. Default value is `.TRUE.`

**VALUE\_TYPE** defines the input type in the `VALUES` included in the input. Must be `INTEGER`, `REAL`, `CHARACTER`, or `LOGICIAL`.

**REAL\_VALUES** a set of values for the `INDEX` field type. Values for an individual scenario are chosen randomly from the list of values provided. The type of the inputs is determined by the type of value needed from the `FIELD` input but all inputs in the `VALUES` inputs are specified in a character array input.

**INTEGER\_VALUES** a set of values for the `INDEX` field type. Values for an individual scenario are chosen randomly from the list of values provided. The type of the inputs is determined by the type of value needed from the `FIELD` input but all inputs in the `VALUES` inputs are specified in a character array input.

**STRING\_VALUES** a set of values for the `INDEX` field type. Values for an individual scenario are chosen randomly from the list of values provided. The type of the inputs is determined by the type of value needed from the `FIELD` input but all inputs in the `VALUES` inputs are specified in a character array input.

**LOGICAL\_VALUES** a set of values for the `INDEX` field type. Values for an individual scenario are chosen randomly from the list of values provided. The type of the inputs is determined by the type of value needed from the `FIELD` input but all inputs in the `VALUES` inputs are specified in a character array input.

**SCENARIO\_TITLES** an optional set of character strings that describe each of the different fields in an `INDEX` set of values.

**BASE\_SCALING\_VALUE** Initial value of the field for the SCALING type field. Defaults to the value in the base input file if not included.

Example:

The following example shows the height of the bedroom varying uniformly from 2.44 m to 3.66 m.

```
&MRND ID = 'Room height generator', TYPE = 'UNIFORM', MINIMUM = 2.44,  
MAXIMUM = 3.66 /  
&MFLD ID = 'Height bedroom' FIELD = 'Bedroom', 'HEIGHT' RAND_ID = 'Room  
height generator' PARAMETER_COLUMN_LABEL = 'Bedroom height' /
```

The following example shows the height of a door to the bedroom in the above example varying from a minimum of 2 m to a maximum 0.15 m below the ceiling of the varying height of the bedroom ceiling.

```
&MRND ID = 'Door Top Generator' DISTRIBUTION_TYPE = 'UNIFORM' VALUE_TYPE =  
'REAL'  
MINIMUM = 2.0 MAXIMUM_FIELD = 'Height Bedroom' MAXIMUM_OFFSET = -0.15 /  
&MFLD ID = 'Door Top' FIELD_TYPE = 'VALUE' RAND_ID = 'Door Top Generator'  
FIELD = 'Wall Vent' 'TOP' /
```

## 2.4 Namelist MFIR

Automatically-generated fires are created for an individual test case by modifying an existing fire input in the input file. The `&MFIR` input allows the user to either scale the HRR curve (multiplying the time and/or HRR values by a constant defined by a user-specified distribution), to define a power law fire based on several inputs, or to modify individual time points from a constant defined by a user-specified distribution.

**ID** All inputs are defined by a unique alphanumeric name. This may be as simple as a single character or number, or a description of the field. All IDs must be unique throughout an input file. The `ID` can be any ASCII string up to 128 characters.

**FYI** A user defined comment that can provide additional information about the input.

**FIRE\_ID** Specifies the associated fire in the data file that is used as a template to be modified by this `&MFIR` input. All inputs in the current `&MFIR` namelist will modify this fire.

**MODIFY\_FIRE\_AREA\_TO\_MATCH\_HRR** IF set to `.TRUE.`, the fire area is calculated from the heat release rate values for the time curve from the formula. Values are calculated from heat release rate such that the fire Froude number is unity<sup>1</sup>.

**FIRE\_COMPARTMENT\_RANDOM\_GENERATOR\_ID** A random generator that returns an index to pick the compartment the fire is located. Usually uses `USER_DEFINED_DISCRETE_DISTRIBUTION`.

**FIRE\_COMPARTMENT\_IDS** A list of compartment names that have a chance of having the fire in them. This matches the index chosen by `FIRE_COMPARTMENT_RANDOM_GENERATOR_ID` to actual compartment names in the input file.

**ADD\_FIRE\_COMPARTMENT\_ID\_TO\_PARAMETERS** set to `.TRUE.` to include the value of the field in the parameters file, `<project>_parameters.csv`. Default value is `.TRUE.`

**FIRE\_COMPARTMENT\_ID\_COLUMN\_LABEL** specifies the column title for this input field in the parameters file, `<project>_parameters.csv`. If it is not included, it will be filled in by the code.

Fires can be defined by either scaling time and/or HRR values from a base fire or by defining a power law fire growth / decay with a peak plateau period. Either fire can be proceeded by an optional period of incipient fire growth period. Firstly, the inputs for the incipient fire are described.

**FLAMING\_SMOLDERING\_IGNITION\_DELAY\_RANDOM\_GENERATOR\_ID** This and the two entries `INCIPIENT_FIRE_TYPES` and `TYPE_OF_INCIPIENT_GROWTH` work together. This one determines if you have a smoldering or flaming incipient fire for a specific generated scenario. If not defined, then only one of the other two sets of two can be defined and the other two must not be left blank. This is an `INDEX` field type.

---

<sup>1</sup>The Fire Froude Number,  $\dot{Q}^*$ , is defined as  $\dot{Q}^* = \frac{\dot{Q}}{\rho_{\infty} c_p T_{\infty} \sqrt{g D D^2}}$ . It is essentially the ratio of the fuel gas exit velocity and the buoyancy-induced plume velocity. Jet fires are characterized by large Froude numbers. Typical accidental fires have a Froude number near unity.

**INCIPIENT\_FIRE\_TYPES** A list made up of 'FLAMING' and 'SMOLDERING'. This allows users to coordinate the incipient beginning of the fire with other inputs in the scenario.

**TYPE\_OF\_INCIPIENT\_GROWTH** Defaults to NONE but can be FLAMING, SMOLDERING, and RANDOM

**FLAMING\_INCIPIENT\_DELAY\_RANDOM\_GENERATOR\_ID** This is the first of the flaming set of generators and defines an associated &MRND ID that defines the length of the incipient flaming fire burning period. If you do not want the value to change you can set it up to a CONSTANT random generator. If included, both FLAMING\_IGNITION\_DELAY\_RANDOM\_GENERATOR\_ID and PEAK\_FLAMING\_IGNITION\_RANDOM\_GENERATOR\_ID must be defined.

**FLAMING\_INCIPIENT\_PEAK\_RANDOM\_GENERATOR\_ID** This is the second of the flaming set of generators and defines an associated &MRND ID that defines the final HRR for the incipient flaming fire ramp. If you do not want the value to change you can set it up to a CONSTANT random generator. If included, both FLAMING\_IGNITION\_DELAY\_RANDOM\_GENERATOR\_ID and PEAK\_FLAMING\_IGNITION\_RANDOM\_GENERATOR\_ID must be defined. This one determines the final HRR for the flaming fire ramp. It is important to understand that using this means the first two HRR points will be overwritten and all times for all entries in the fire will be set to the same.

**SMOLDERING\_INCIPIENT\_DELAY\_RANDOM\_GENERATOR\_ID** This is the smoldering pair that match the above.

**SMOLDERING\_INCIPIENT\_PEAK\_RANDOM\_GENERATOR\_ID** Same as above for smoldering fire.

**ADD\_IGNITION\_TYPE\_TO\_PARAMETERS** set to .TRUE. to include the value of the field in the parameters file, <project>\_parameters.csv. Default value is .TRUE. Column displays 'FLAMING' or 'SMOLDERING' depending on which type of ignition happens.

**ADD\_SMOLDERING\_IGNITION\_TIME\_TO\_PARAMETERS** set to .TRUE. to include the value of the field in the parameters file, <project>\_parameters.csv. Default value is .TRUE.

**ADD\_SMOLDERING\_IGNITION\_PEAK\_TO\_PARAMETERS** Name shortened because adding the 'ING' made the variable name too long and it got truncated. Defaults to .TRUE.

**ADD\_FLAMING\_IGNITION\_TIME\_TO\_PARAMETERS** set to .TRUE. to include the value of the field in the parameters file, <project>\_parameters.csv. Default value is .TRUE.

**ADD\_FLAMING\_IGNITION\_PEAK\_HRR\_TO\_PARAMETERS** set to .TRUE. to include the value of the field in the parameters file, <project>\_parameters.csv. Default value is .TRUE.

**IGNITION\_TYPE\_COLUMN\_LABEL** specifies the column title for this input field in the parameters file, <project>\_parameters.csv. If it is not included, it will be filled in by the code.

**FLAMING\_HRR\_PEAK\_COLUMN\_LABEL** specifies the column title for this input field in the parameters file, `<project>_parameters.csv`. If it is not included, it will be filled in by the code.

**FLAMING\_TIME\_COLUMN\_LABEL** specifies the column title for this input field in the parameters file, `<project>_parameters.csv`. If it is not included, it will be filled in by the code.

**SMOLDERING\_HRR\_PEAK\_COLUMN\_LABEL** specifies the column title for this input field in the parameters file, `<project>_parameters.csv`. If it is not included, it will be filled in by the code.

**SMOLDERING\_TIME\_COLUMN\_LABEL** specifies the column title for this input field in the parameters file, `<project>_parameters.csv`. If it is not included, it will be filled in by the code.

Secondly, inputs for the scaling fire are described.

**BASE\_FIRE\_ID** For scaling fires only. This provides internal temporary storage for all the base values for a fire to be scaled. Typically, it will just be a copy of the corresponding `FIRE_ID` input.

**SCALING\_FIRE\_HRR\_RANDOM\_GENERATOR\_ID** an associated &MRND ID that defines the scaling factor that is applied to all the HRR values in a fire. This is done before the incipient fire model is calculated so it won't impact the delay or HRR value for the incipient fire portion of the fire curve.

**SCALING\_FIRE\_TIME\_RANDOM\_GENERATOR\_ID** an associated &MRND ID that defines the scaling value that is applied to all the TIME values in a fire. This is done before the Incipient fire model is calculated so it won't impact the delay or HRR value for the incipient portion of the fire curve.

**ADD\_HRR\_SCALE\_TO\_PARAMETERS** set to .TRUE. to include the value of the field in the parameters file, `<project>_parameters.csv`. Default value is .TRUE.

**HRR\_SCALE\_COLUMN\_LABEL** specifies the column title for this input field in the parameters file, `<project>_parameters.csv`. If it is not included, it will be filled in by the code.

**ADD\_TIME\_SCALE\_TO\_PARAMETERS** set to .TRUE. to include the value of the field in the parameters file, `<project>_parameters.csv`. Default value is .TRUE.

**TIME\_SCALE\_COLUMN\_LABEL** specifies the column title for this input field in the parameters file, `<project>_parameters.csv`. If it is not included, it will be filled in by the code.

Finally, the inputs for a general power law fire are described. The calculation of the growth and decay points is straightforward enough. The inputs are  $t_0$ ,  $t_1$ ,  $\dot{Q}_0$ ,  $\dot{Q}_1$ , and the exponential growth  $r$ . The variables  $t_0$  and  $t_1$  are the beginning and ending time points and  $\dot{Q}_0$  and  $\dot{Q}_1$  are the HRR at the beginning and end and  $r$  is already defined. For growth  $\hat{t} = t + t_0$  where  $0 < t < t_1 - t_0$  and for

decay  $\hat{t} = t_1 - t$  where  $0 < t < t_1 - t_0$ . The equation for the HRR at time  $t$  in the growth phase is  $\dot{Q}(\hat{t}) = ((\dot{Q}_1 - \dot{Q}_0)/(t_1 - t_0)^r)\hat{t} + \dot{Q}_0$  and for decay is  $\dot{Q}(\hat{t}) = ((\dot{Q}_0 - \dot{Q}_1)/(t_1 - t_0)^r)\hat{t} + \dot{Q}_1$ . It can be done based on which,  $\dot{Q}_0$  or  $\dot{Q}_1$ , are larger.

**FIRE\_TIME\_GENERATOR\_IDS** a set of up to 100 associated &MRND inputs that define the time intervals for fire growth If you do not want the values to change you can set any of the &MRND inputs to a CONSTANT random generator. The first point is always at  $t = 0$  and  $\dot{Q} = 0$ . If there is an incipient fire defined, that is point two. If there is a power law growth defined (by GROWTH\_EXPONENT and GROWTH\_EXPONENT, these follow the first or second time point (depending on whether there is an incipient fire defined). If a power law decay is defined (by DECAY\_EXPONENT and DECAY\_EXPONENT, these are defined by the next to last and last generators defined.

**FIRE\_HRR\_GENERATOR\_IDS** a set of up to 100 associated &MRND inputs that define the HRR values corresponding with each defined time generator. If you do not want the values to change you can set any of the &MRND inputs to a CONSTANT random generator.

**NUMBER\_OF\_GROWTH\_POINTS** specifies the number of data points included in the growth phase of the fire.

**NUMBER\_OF\_DECAY\_POINTS** specifies the number of data points included in the decay phase of the fire.

**GROWTH\_EXPONENT** specifies the exponent of the power law for the growth phase of the fire. Must be a positive number. Only one of GROWTH\_EXPONENT or TIME\_TO\_1054\_KW can be included in an &MFIR input.

**TIME\_TO\_1054\_KW** specifies the time for the growth phase of a power law fire to grow from 0 kW to 1054 kW. Must be a positive number. Only one of GROWTH\_EXPONENT or TIME\_TO\_1054\_KW can be included in an &MFIR input.

**DECAY\_EXPONENT** specifies the exponent of the power law for the decay phase of the fire. Must be a positive number. Only one of DECAY\_EXPONENT or TIME\_TO\_0\_KW can be included in an &MFIR input.

**TIME\_TO\_0\_KW** specifies the time for the growth phase of a power law fire to decay from its peak value to 0 kW. Must be a positive number. Only one of DECAY\_EXPONENT or TIME\_TO\_0\_KW can be included in an &MFIR input.

**ADD\_HRR\_TO\_PARAMETERS** set to .TRUE. to include the value of the field in the parameters file, <project>.parameters.csv. Default value is .TRUE.

**ADD\_TIME\_TO\_PARAMETERS** set to .TRUE. to include the value of the field in the parameters file, <project>.parameters.csv. Default value is .TRUE.

**TIME\_COLUMN\_LABELS** specifies the column titles for this input field in the parameters file, <project>.parameters.csv. The labels are input as an array of names that can be less than or equal to the total number of FIRE\_TIME\_GENERATORS or FIRE\_HRR\_GENERATORS. If it is not included or set to NULL, it will be filled in by the code.

**HRR\_COLUMN\_LABELS** specifies the column titles for this input field in the parameters file, <project>.parameters.csv. The labels are input as an array of names that can be less than or equal to the total number of FIRE\_TIME\_GENERATORS or FIRE\_HRR\_GENERATORS. If it is not included or set to NULL, it will be filled in by the code.

Example:

```

&MRND ID = 'Generator for fire rooms' DISTRIBUTION_TYPE =
  'USER_DEFINED_DISCRETE' VALUE_TYPE = 'INTEGER'
  INTEGER_VALUES = 1, 2, 3 PROBABILITIES = 0.3333 0.3333 0.3334 /
&MRND ID = 'Flaming Smoldering Generator' DISTRIBUTION_TYPE =
  'USER_DEFINED_DISCRETE' VALUE_TYPE = 'INTEGER'
  INTEGER_VALUES = 1 2 PROBABILITIES = 0.5 0.5 /
&MRND ID = 'Flaming Time Generator' DISTRIBUTION_TYPE = 'TRUNCATED_NORMAL'
  VALUE_TYPE = 'REAL' MEAN = 207 STDEV = 46
  MINIMUM = 10 MAXIMUM = 1000000/
&MRND ID = 'Flaming Peak Generator' DISTRIBUTION_TYPE = 'TRUNCATED_NORMAL'
  VALUE_TYPE = 'REAL' MEAN = 23000 STDEV = 7000
  MINIMUM = 100 MAXIMUM = 1000000/
&MRND ID = 'Smoldering Time Generator' DISTRIBUTION_TYPE = 'TRUNCATED_NORMAL'
  VALUE_TYPE = 'REAL' MEAN = 6863 STDEV = 1812
  MINIMUM = 60 MAXIMUM = 14490/
&MRND ID = 'Smoldering Peak Generator' DISTRIBUTION_TYPE = 'TRUNCATED_NORMAL'
  VALUE_TYPE = 'REAL' MEAN = 11000 STDEV = 3000
  MINIMUM = 100 MAXIMUM = 1000000/
&MRND ID = 'Growth Time Generator', DISTRIBUTION_TYPE = 'TRUNCATED_NORMAL'
  VALUE_TYPE = 'REAL' MEAN = 205, STDEV = 65
  MINIMUM = 75 MAXIMUM = 500 /
&MRND ID = 'Peak HRR Generator', DISTRIBUTION_TYPE = 'CONSTANT' VALUE_TYPE =
  'REAL' REAL_CONSTANT_VALUE = 1054000 /
&MRND ID = 'Peak Plateau End Time' DISTRIBUTION_TYPE = 'CONSTANT' VALUE_TYPE
  = 'REAL' REAL_CONSTANT_VALUE = 10 /
&MRND ID = 'Fire End Time' DISTRIBUTION_TYPE = 'CONSTANT' VALUE_TYPE = 'REAL'
  REAL_CONSTANT_VALUE = 10 /
&MRND ID = 'Fire End HRR' DISTRIBUTION_TYPE = 'CONSTANT' VALUE_TYPE = 'REAL'
  REAL_CONSTANT_VALUE = 0 /

&MFIR ID = 'Fire Generator' FIRE_ID = 'Random'
  FIRE_COMPARTMENT_RANDOM_GENERATOR_ID = 'Generator for fire rooms'
  FIRE_COMPARTMENT_IDS = 'Dining Room', 'Kitchen', 'Living Room'
  FLAMING_SMOLDERING_IGNITION_RANDOM_GENERATOR_ID = 'Flaming Smoldering
    Generator' TYPE_OF_INCIPIENT_GROWTH = 'RANDOM'
  NUMBER_OF_INCIPIENT_FIRE_TYPES = 2 INCIPIENT_FIRE_TYPES = 'FLAMING'
    'SMOLDERING'
  PEAK_FLAMING_IGNITION_RANDOM_GENERATOR_ID = 'Flaming Time Generator'
  FLAMING_IGNITION_DELAY_RANDOM_GENERATOR_ID = 'Flaming Peak Generator'
  PEAK_SMOLDERING_IGNITION_RANDOM_GENERATOR_ID = 'Smoldering Time Generator'
  SMOLDERING_IGNITION_DELAY_RANDOM_GENERATOR_ID = 'Smoldering Peak
    Generator'
  FIRE_TIME_GENERATORS = 'Growth Time Generator' 'Peak Plateau End Time'
    'Fire End Time'

```

```

FIRE_HRR_GENERATORS = 'Peak HRR Generator'      'Peak HRR Generator'
                      'Fire End HRR'
NUMBER_OF_GROWTH_POINTS = 10 GROWTH_EXPONENT = 2 /

```

## 2.5 Namelist MSTT

CData has a limited number of analysis tools built into it. These tools are activated when CData is called as

```
CData <project>.in -s
```

Control of what tools are used and what data it is run on is determined with the `&MSTT` namelist inputs. The `&MSTT` namelist is the only namelist that can be added or changed and used on the Monte Carlo data without having to rerun the entire analysis.

**ID** All inputs are defined by a unique alphanumeric name. This may be as simple as a single character or number, or a description of the field. All IDs must be unique throughout an input file. The `ID` can be any ASCII string up to 128 characters.

**FYI** A user defined comment that can provide additional information about the input.

**ANALYSIS\_TYPE** Describes the type of analysis to be done. The allowed types are `HISTOGRAM`, `EMPERICAL_PDF`, `CONVERGENCE_OF_MEAN`, and `DECISION_TREES`.

**INPUT\_FILENAME** Comma delimited file that contains the data. The default is the `i<project>_accumulate`

**OUTPUT\_FILENAME** The filename for the finished graphic is a required input. The extension is required as that is the method of determine the format of the file. The accepted formats are `*.jpg`, `*.svg`, `*.tif`, `*.pdf`, `*.png`.

**ERROR\_FILENAME** Error filename defaults to `OUTPUT_FILENAME` with `err` extension by default

**LOG\_FILENAME** Log filename defaults to `OUTPUT_FILENAME` with `log` extension by default

**COLUMN\_LABEL** The name in the first row for the column to use for the analysis.

Example:

```

&MSTT ID = 'Width of Vent' ANALYSIS_TYPE = 'HISTOGRAM' OUTPUT_FILENAME =
  'simple_width.jpg'
  COLUMN_TITLE = 'Wall Vent_WIDTH' /
&MSTT ID = 'Top of Vent' ANALYSIS_TYPE = 'HISTOGRAM' OUTPUT_FILENAME =
  'simple_top.jpg'
  COLUMN_TITLE = 'Wall Vent_TOP' /

```

## 2.6 Namelist DUMP

When cases are run in CFAST, a number of single values for the case such as maximum temperature in a compartment or the time to temperature reaching 600 °C. These values are defined with the &DUMP namelist input.

**ID** Summary outputs are defined by a unique alphanumeric name. The ID is not only used to identify the output but is also the column label for the output. This may be as simple as a single character or number, or a description of the output. All IDs must be unique throughout an input file. The ID can be any ASCII string up to 64 characters.

**FYI** A user defined comment that can provide additional information about the input.

**FILE** specifies which of the CFAST output files are used for the summary output. Allowable inputs are COMPARTMENTS, DEVICES, FIRES, MASSES, or WALLS.

**TYPE** specifies the type of summary data to calculate. Allowable input are MIN, MAX, TRIGGER\_LESSER, TRIGGER\_GREATER, INTEGRATE, TOTAL\_HRR. For MIN or MAX, only the FIRST\_FIELD input is required. For TRIGGER\_LESSER or TRIGGER\_GREATER, the value the of first device when the second device passes the value in CRITERION. For INTEGRATE, FIRST\_FIELD must be "Time", "Simulation Time". INTEGRATE integrates the values of SECOND\_FIELD over the entire simulation time.

**FIRST\_FIELD** specifies the name of the specific input (i.e., the name of a compartment, vent, etc.) and the specific field within that input that is to be used for the first input in a calculation. For example, FIELD="Time", "Simulation Time" would specify the simulation time.

**SECOND\_FIELD** specifies the name of the specific input (i.e., the name of a compartment, vent, etc.) and the specific field within that input that is to be used for the second input in a calculation, if required. For example, FIELD="Comp 1", "Upper Layer Temperature" would specify the upper layer temperature in compartment named Comp 1.

**CRITERION** Specifies the value to be evaluated in a TRIGGER\_LESSER or TRIGGER\_GREATER calculation.

Example:

```
&DUMP ID = 'Total Time Completed'  
    FILE_TYPE = 'DEVICES'  TYPE = 'MAXIMUM'  
    FIRST_FIELD = 'Time', 'Simulation Time' /  
&DUMP ID = 'Fire Room Ion Detector'  
    FILE = 'DEVICES'  TYPE = 'TRIGGER_GREATER'  CRITERIA = 1  
    FIRST_FIELD = 'Time', 'Simulation Time'  
    SECOND_FIELD = 'Ionization Detector Room 1', 'Sensor Activation' /
```



# Chapter 3

## Creating Multiple CFAST Runs

The CData program has several functions, 1) a preprocessor function that generates individual CFAST input files from a user-specified distribution and range for one or more inputs, 2) an accumulator function that collects data of user-specified variables and creates a spreadsheet of summary data for all the individual runs, and 3) a statistical function that creates several different statistical outputs of the summary data to facilitate further analysis. Section 2 discusses all the namelist commands for CData. The final section discusses the significant issue of storage, which can limit on the size of the analysis that can be done.

All inputs for generating a set of multiple CFAST runs are contained within a single CFAST input file with inputs that define the base case for analysis, and Monte Carlo-related inputs that define how those inputs will be varied to create multiple individual CFAST input files. In addition, the file may contain a series of specific inputs to generate summary outputs for statistical analysis of the results and a series of inputs that define simple statistical analyses on the summary data. This chapter presents a simple example from start to finish.

The base case, `Simple.in`, defines a single compartment 3 m x 3 m x 3 m with a single door to the outside. Compartment surface are constructed of 0.15 m thick concrete. For the example, the width and height of the door will be varied along with the peak heat release rate of the fire, assumed to be a simple fire that grows to its maximum in 10 s, burns at a specified constant heat release rate for 900 s and decays back to a zero heat release rate in 10 s. For the simple analysis, we will look at the maximum upper layer temperature and time for the upper layer to descend to a height of 1.5 m, both indicators of increasing hazard within the compartment.

### 3.1 PreProcessor

To create the varying door size inputs, the vent is defined in the input file and specifications for varying the width and height are included. Varying specific CFAST input requires two inputs, one that defines the distribution of values for the input (the `&MRND` inputs below), and one or more that define the variables in the CFAST input file that depend on the specified distribution (`&MFLD` inputs below).

```
!! Wall Vents
&MRND ID = 'Vent Width Generator' DISTRIBUTION_TYPE = 'UNIFORM' VALUE_TYPE = 'REAL'
  MINIMUM = 0.25 MAXIMUM = 2.0/
&MFLD ID = 'Wall Vent Width' FIELD_TYPE = 'VALUE' RAND_ID = 'Vent Width Generator'
```

```

FIELD = 'Wall Vent' 'WIDTH' /
&MRND ID = 'Vent Height Generator' DISTRIBUTION_TYPE = 'UNIFORM' VALUE_TYPE = 'REAL'
    MINIMUM = 1.5 MAXIMUM = 2.5/
&MFLD ID = 'Wall Vent Height' FIELD_TYPE = 'VALUE' RAND_ID = 'Vent Height Generator'
    FIELD = 'Wall Vent' 'TOP' /
&VENT TYPE = 'WALL' ID = 'Wall Vent' COMP_IDS = 'Comp 1' 'OUTSIDE' , BOTTOM = 0 TOP = 2,
    WIDTH = 1 FACE = 'FRONT' OFFSET = 1 /

```

The first two inputs define the distribution for the door width, a uniform distribution from 0.25 m to 2.0 m. The second two input define the distribution for the height of the door, a uniform distribution from 1.5 m to 2.5 m. The last input defines the normal CFAST input defining the base values for the door. The `WIDTH` and `HEIGHT` inputs are replaced with random values for each individual CFAST input file generated. All other values in the `&VENT` input remain at the base values.

To define the fire, the `&MRND` input defines the peak heat release rate and the time intervals for the fire curve. The `&MFIR` input combines these with the rest of the fire definition included in the input file to create individual fire inputs for each generated CFAST input file.

Here, we define the peak heat release rate as a uniform distribution from 500 kW to 3000 kW, with a 10 s ramp from zero HRR (by default, the first time point is defined at zero time and zero HRR), a 900 s constant fire at the peak heat release rate, and a 10 s decay back to zero.

As part of the base input file used to generate the set of individual CFAST inputs files to be run, one or more `&DUMP` inputs can be included to specify summary outputs for each CFAST simulation. This can include maximum/minimum values, time to chosen trigger values (for example, time to peak heat release rate or time to a chosen upper layer temperature), or trigger values based on other inputs (for example, heat release rate when upper layer temperature reaches a chosen value).

For the example in this chapter, inputs are included to determine the peak heat release rate, maximum upper layer temperature, minimum height of the layer interface, the time for the layer interface to descend to 1.5 m from the floor, the time for the upper layer temperature to reach 600 °C. and the heat release rate when the upper layer temperature reaches 600 °C.

```

&DUMP ID = 'Maximum Actual HRR' FILE = 'COMPARTMENTS' TYPE = 'MAXIMUM'
    FIRST_FIELD = 'Fire' 'HRR Actual' /
&DUMP ID = 'Maximum Upper Layer Temp' FILE = 'COMPARTMENTS' TYPE = 'MAXIMUM'
    FIRST_FIELD = 'Comp 1' 'Upper Layer Temperature' /
&DUMP ID = 'Minimum Layer Height' FILE = 'COMPARTMENTS' TYPE = 'MINIMUM'
    FIRST_FIELD = 'Comp 1' 'Layer Height' /

&DUMP ID = 'Time to Layer Height 1.5 m' FILE = 'COMPARTMENTS' TYPE = 'TRIGGER_LESSER'
    FIRST_FIELD = 'Time' 'Simulation Time' SECOND_FIELD = 'Comp 1' 'Layer Height' CRITERION =
        1.5 /
&DUMP ID = 'Time to Upper Layer 600 C' FILE = 'COMPARTMENTS' TYPE = 'TRIGGER_GREATER'
    FIRST_FIELD = 'Time' 'Simulation Time' SECOND_FIELD = 'Comp 1' 'Upper Layer Temperature'
        CRITERION = 600 /

&DUMP ID = 'Actual HRR at Upper Layer 600 C' FILE = 'COMPARTMENTS' TYPE = 'TRIGGER_GREATER'
    FIRST_FIELD = 'Fire' 'HRR Actual' SECOND_FIELD = 'Comp 1' 'Upper Layer Temperature'
        CRITERION = 600 /

```

With the rest of the input file defining the base input file, creating a set of CFAST input files requires running CData from a command prompt with the `-P` option as

```
cdata Simple.in -P
```

This creates all the individual input files, batch scripts to run the file on either Windows or Linux (with minor editing to define the locations of the required executables), and a summary spreadsheet of all in varied input values.

## 3.2 Running CFAST

As part of the process of creating the individual CFAST inputs files, CData creates batch scripts for both windows and Linux operating systems. Each batch script depends on external software to support running multiple CFAST jobs in parallel<sup>1</sup>. Both of these scripts include information on the locations of these external files (plus the location of the CFAST executable which may need to be modified to suit a particular computer hardware. Default examples are shown below. Once configured, running the set of CFAST simulations is accomplished by running the appropriate script for Windows or Linux. By default, a maximum of 100 000 iterations are set for each run to ensure that jobs which take an extremely long time to run do not prevent the rest from running. This value can be changed in both of the batch scripts.

### Default Windows Batch Script for Simple.in:

```

echo off
rem change the path to background.exe and cfast.exe as appropriate.
rem Here we just assume it is in the path
set bgexe=background.exe
set CFAST_EXE=cfast.exe
set MAX_ITER=100000

rem you should not need to change anything from here on
set bg=%bgexe% -u 6
set CFAST=%bg% %CFAST_EXE%
echo %MAX_ITER% > Simple-1.stop
%CFAST% Simple-1.in -v
echo %MAX_ITER% > Simple-2.stop

```

---

<sup>1</sup>For Windows, a program, `background.exe` is used and is included in the CFAST software distribution. For Linux, the batch script depends on a script developed for running multiple FDS runs, `qfds.sh`, is used.

```
%CFAST% Simple-2.in -v
echo %MAX_ITER% > Simple-3.stop
%CFAST% Simple-3.in -v

:loop1
tasklist | find /i /c "CFAST" > temp.out
set /p numexe=<temp.out
echo waiting for %numexe% jobs to finish
if %numexe% == 0 goto finished
Timeout /t 30 >nul
goto loop1
:finished
```

### **Default Linux Batch Script for Simple.in:**

```
#!/bin/bash
CFAST=~/firemodels/cfast/Build/CFAST/intel_linux_64/cfast7_linux_64
MAX_PROCESSORS=30
BATCH=batch

export STOPFDSSMAXITER=100000

qfds.sh -U $MAX_PROCESSORS -e $CFAST -q $BATCH Simple-1.in
qfds.sh -U $MAX_PROCESSORS -e $CFAST -q $BATCH Simple-2.in
qfds.sh -U $MAX_PROCESSORS -e $CFAST -q $BATCH Simple-3.in
```

## **3.3 Generating Statistics**

From these summary values defined in the last section with the `&DUMP` inputs, selected statistical analyses are available in CData. This section describes how to create a spreadsheet of summary values (the `Accumulate` function in CData), and how to generate several statistical analyses from those data (the `Statistics` function in CData).

### **3.3.1 Accumulator**

In order to allow CFAST cases to be run efficiently in parallel on systems with that capability, each simulation independently creates a file, the `<project>_nn_calculations.csv` file where `nn` is the number of the file. This file has the summary values specified in the `&DUMP` inputs for that case. To create the spreadsheet file of summary values after all of the CFAST cases have been run, CData is run with the `-A` option,

```
cdata Simple.in -A
```

### **3.3.2 Statistics**

The base input file used to generate the set of individual CFAST input files may also contain specifications to generate summary statistics of the collected data. Unlike the `&DUMP` inputs, this can be added after all the runs are completed to do new statistics. However, it is important to keep in mind that analysis can only be done on data that was generated during the CFAST runs. So it is important to include `&DUMP` inputs for everything that might be of interest to avoid having to rerun

all the cases. The summary statistics can include histograms of input or output data (often used to verify that the range of input values match expectations), convergence of mean value (used to determine if sufficient runs have been made, below), probability density plots, and correlation trees of the relative importance of selected inputs to the calculated outputs. For the simple example in this chapter, several of these are included.

```
&MSTT ID = 'Width of Vent' ANALYSIS_TYPE = 'HISTOGRAM' OUTPUT_FILENAME = 'simple_width.jpg'
COLUMN_TITLE = 'Wall Vent_WIDTH' /
&MSTT ID = 'Top of Vent' ANALYSIS_TYPE = 'HISTOGRAM' OUTPUT_FILENAME = 'simple_top.jpg'
COLUMN_TITLE = 'Wall Vent_TOP' /
&MSTT ID = 'Peak HRR' ANALYSIS_TYPE = 'HISTOGRAM' OUTPUT_FILENAME = 'simple_peak_hrr.jpg'
COLUMN_TITLE = 'Fire_HRR_PT_2' /
&MSTT ID = 'Time to FO' ANALYSIS_TYPE = 'HISTOGRAM' OUTPUT_FILENAME = 'Simple_Time_to_FO.jpg'
COLUMN_TITLE = 'Time to Upper Layer 600 C' /
&MSTT ID = 'Max Upper Temp' ANALYSIS_TYPE = 'HISTOGRAM'
OUTPUT_FILENAME = 'Simple_MaxUpperTemp.jpg' COLUMN_TITLE = 'Maximum Upper Layer Temp' /

&MSTT ID = 'Convergence of Layer Height Reaching 1.5' OUTPUT_FILENAME = 'Simple_time_to_1p5.jpg'
ANALYSIS_TYPE = 'CONVERGENCE_OF_MEAN' COLUMN_TITLE = 'Time to Layer Height 1.5 m' /
&MSTT ID = 'Convergence of Max Temp' OUTPUT_FILENAME = 'Simple_max_temp.jpg'
ANALYSIS_TYPE = 'CONVERGENCE_OF_MEAN' COLUMN_TITLE = 'Maximum Upper Layer Temp' /

&MSTT ID = 'Decision Tree on Temp' OUTPUT_FILENAME = 'Simple_tree_temp.jpg'
ANALYSIS_TYPE = 'DECISION TREES' COLUMN_TITLE = 'Maximum Upper Layer Temp' /
```

To create the spreadsheet file of summary values after all of the CFAST cases have been run, CData is run with the `-S` option,

```
cdata Simple.in -S
```

When a `&MSTT` namelist is processed, it generates a number of files. These include an error file (\*.err) if there is an error in the analysis, a log file (\*.log) that documents the steps that are taken in the analysis and the graphic file in the format requested. For '`HISTOGRAM`', '`EMPERICAL_PDF`', '`CONVERGENCE_OF_MEAN`' there is also a final \*.csv file. This allows users to use their preferred graphics packages to create the graphics. There is no extra output for the '`DECISION TREES`' because there is not a standard simple method of documenting a tree graph.

## Histograms

The result of the first `&MSTT` is shown in Fig. 3.1.

There is clearly some noise in the data and all the columns are not the same height as would be expected in theoretical uniform distribution. However, there really is only one column, the first one, that is significantly out of line with the others. One way of trying to understand why that column is so much smaller is to go into the `simple_width.csv` file. In that file the first bar is for the range 0.2 m to 0.3 m but the random generator had a lower bounds of 0.25 m. Doubling the 134 count puts it right in the middle of the values for the other bars. With that question answered the chart seems to be a reasonable representation of a uniform distribution.

In this automatic analysis the number of bins used here is automatically selected using Sturges rule [21] which chooses the number of bins according to the rule:

$$k = \lceil \log_2 n \rceil + 1, \quad (3.1)$$

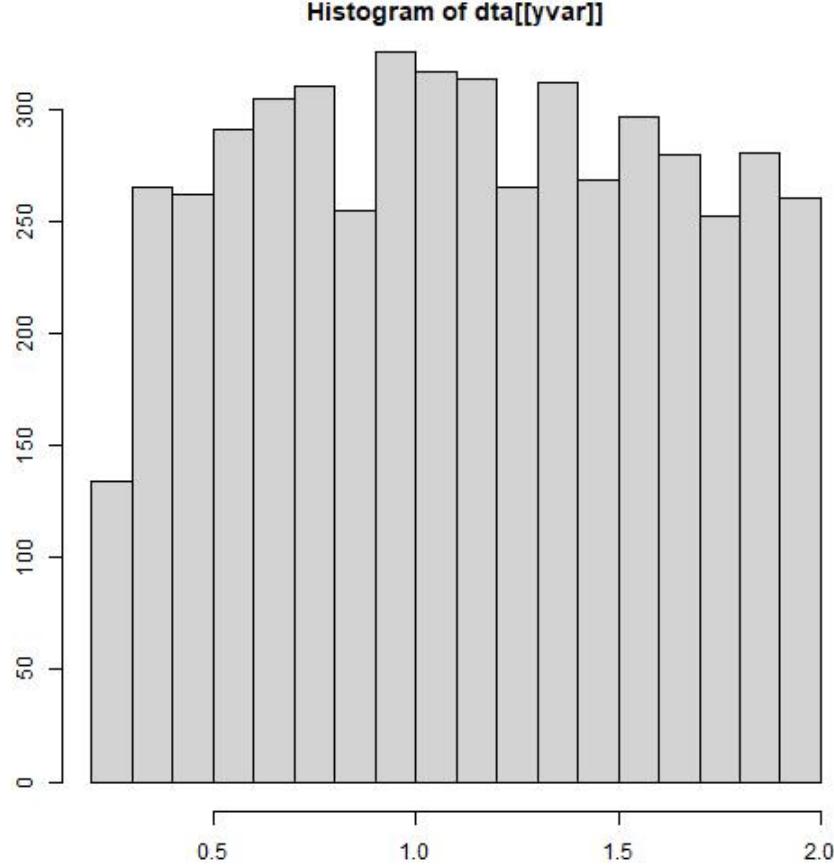


Figure 3.1: Sample of Histogram generated for the 'Width of Vent' column

where  $\lceil \log_2 n \rceil$  represents the ceiling function for  $\log_2 n$ .

### Empirical Probability Density Function

The post-processor is also capable of generating an empirical probability density function. It does so using the techniques of Kernel Density Estimation [22]. To determine the probability density at a point  $x$  it associates a weight with each point in the data set. The weight is a function of the distance between the point in the data set and  $x$ . The probability density is then the sum of those weights.

Typically the Gaussian function is used as the Kernel (i.e., the weighting function), although other Kernels are also commonly used. For the Gaussian Kernel (which is used here) the estimated probability density at a point  $x$  is:

$$(x) = \frac{1}{Nb} \sum_{j=1}^N \phi \left( \frac{x - x_j}{b} \right) \quad (3.2)$$

where  $N$  is the number of data points,  $x_j$  is an individual data point,  $\phi$  is the standard normal density function, and  $b$  is the "bandwidth." The bandwidth determines the width of the window within which the data points contribute significantly to the density estimate. In this implementation, the bandwidth is determined automatically.

## Determining If Enough Runs Have Been Made

Monte Carlo analysis is fundamentally grounded in The Law of Large Numbers, which states that for a large-enough number of independent samples, the average converges to the expected value. For example, if we want to know what the probability of flashover is in the kitchen for a certain class of fires, the Law of Large Numbers assures us that the percentage of Monte Carlo runs where flashover occurs converges to the probability as the number of runs becomes large.

The problem here is determining how many runs is enough. Put differently, how many runs does it take to guarantee a result that is "close enough" to be useful? Two strategies are used here; the first is graphing the evolution of the mean versus number of runs, the second is to graph the standard deviation of the mean versus the number of runs. The first strategy should show decreasing variation as the number of runs increases and should converge on a point. The second strategy should show the variance decreasing with the number of runs and approaching zero. The determination of whether the estimates are "close enough" is a judgment of the analyst and will depend on what level of accuracy is needed.

In this simple example the mean of two outputs 'Time to Layer Height 1.5 m' and 'Maximum Upper Layer Temp' are generated. Figure 3.2 shows the convergence of the mean. From this graph it seems clear that the mean has converged and that fewer cases could have been used.

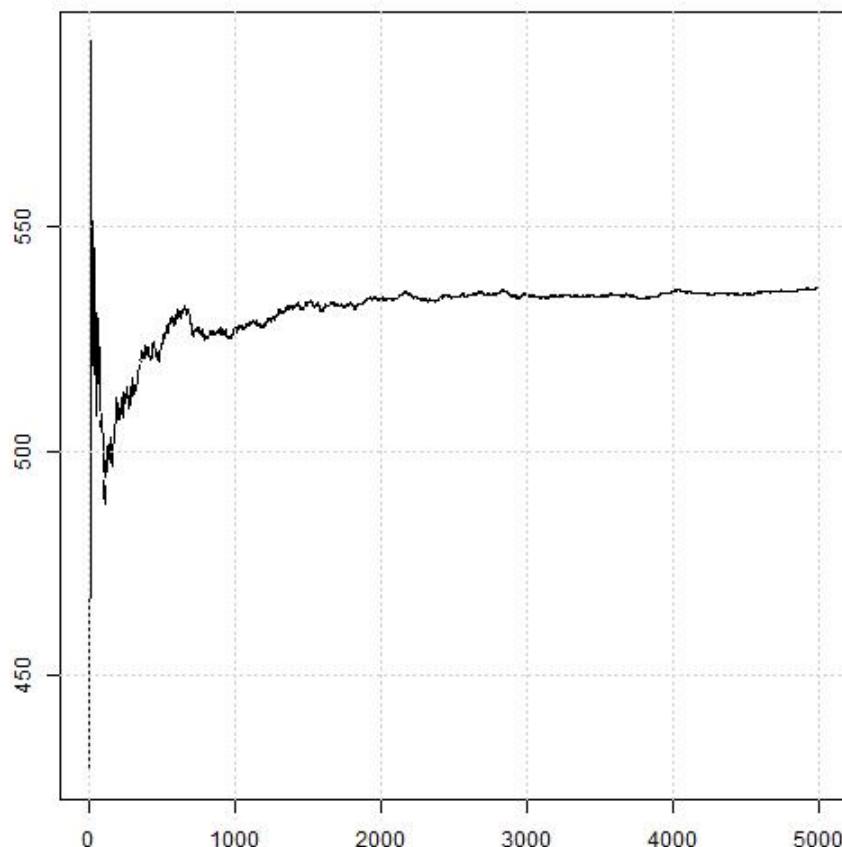


Figure 3.2: Sample of Convergence of mean for maximum upper layer temperature

Figure 3.3 shows the standard deviation, which clearly is tending toward zero.

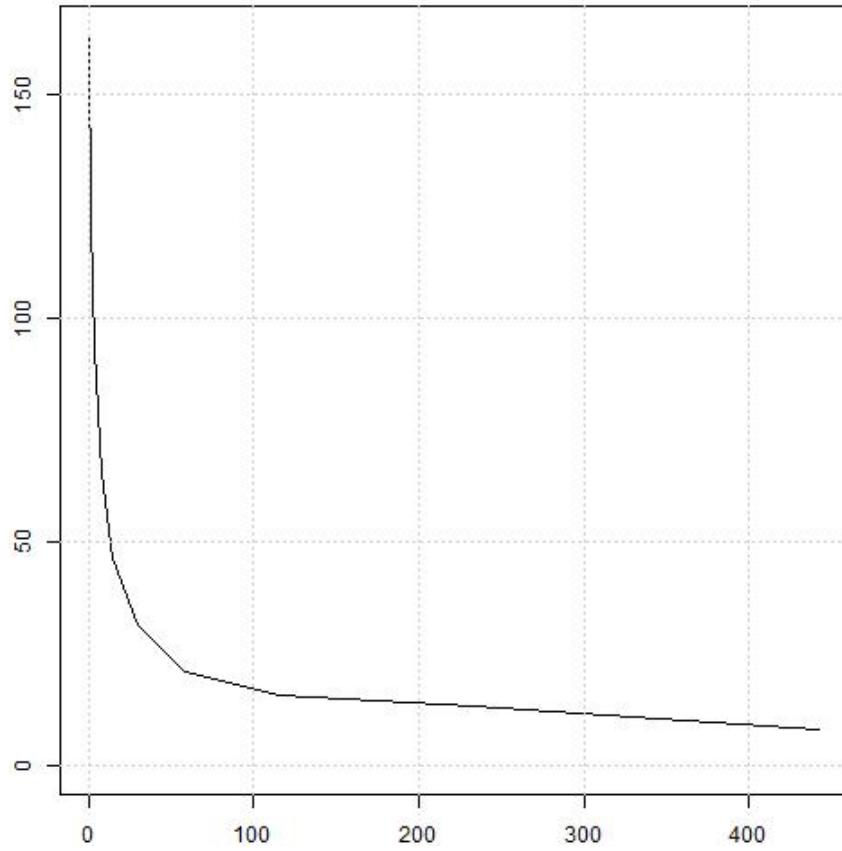


Figure 3.3: Sample of Convergence of mean for maximum upper layer temperature

## Decision Trees

Sometimes what is of interest is the relationship one variable has to other variables in the output. There are many ways of estimating relationships, but one of the most intuitive is that of Decision Trees. The approach implemented here is closely related to that of Classification and Regression Trees [22]. The output of the algorithm is a binary tree. Each node splits the data into two, until the tree arrives at the leaf nodes. For variables that are discrete, for example if a detector has activated or not, each leaf node is associated with the best-fit category or most likely. For continuous variables, each leaf node is associated with the average of the data that arrives at that node.

Decision trees have the advantage of being very easy to interpret. Their interpretability is one reason for their popularity. Trees have a couple of drawbacks as well. The most noticeable is that their predictions are not smooth. In addition, trees can be highly unstable. That is, small changes in the data can produce large changes in the estimated tree. In addition, there are cases where the same data can be represented by very different trees.

One of the &MSTT namelists for Simple.in creates a decision tree for maximum layer temperature. Figure 3.4 shows the tree. The top node or root node says that if the second HRR point is less than 158.5 kW, which is the left branch, then the average maximum upper temperature is less than flashover. However if the HRR at the second point is greater than 230.5 kW, taking the right branch from the root and then the right branch again, the average maximum upper layer temperature is above flashover, although with a big enough vent the average is close to 600 °C, meaning that likely

some cases do not flashover.

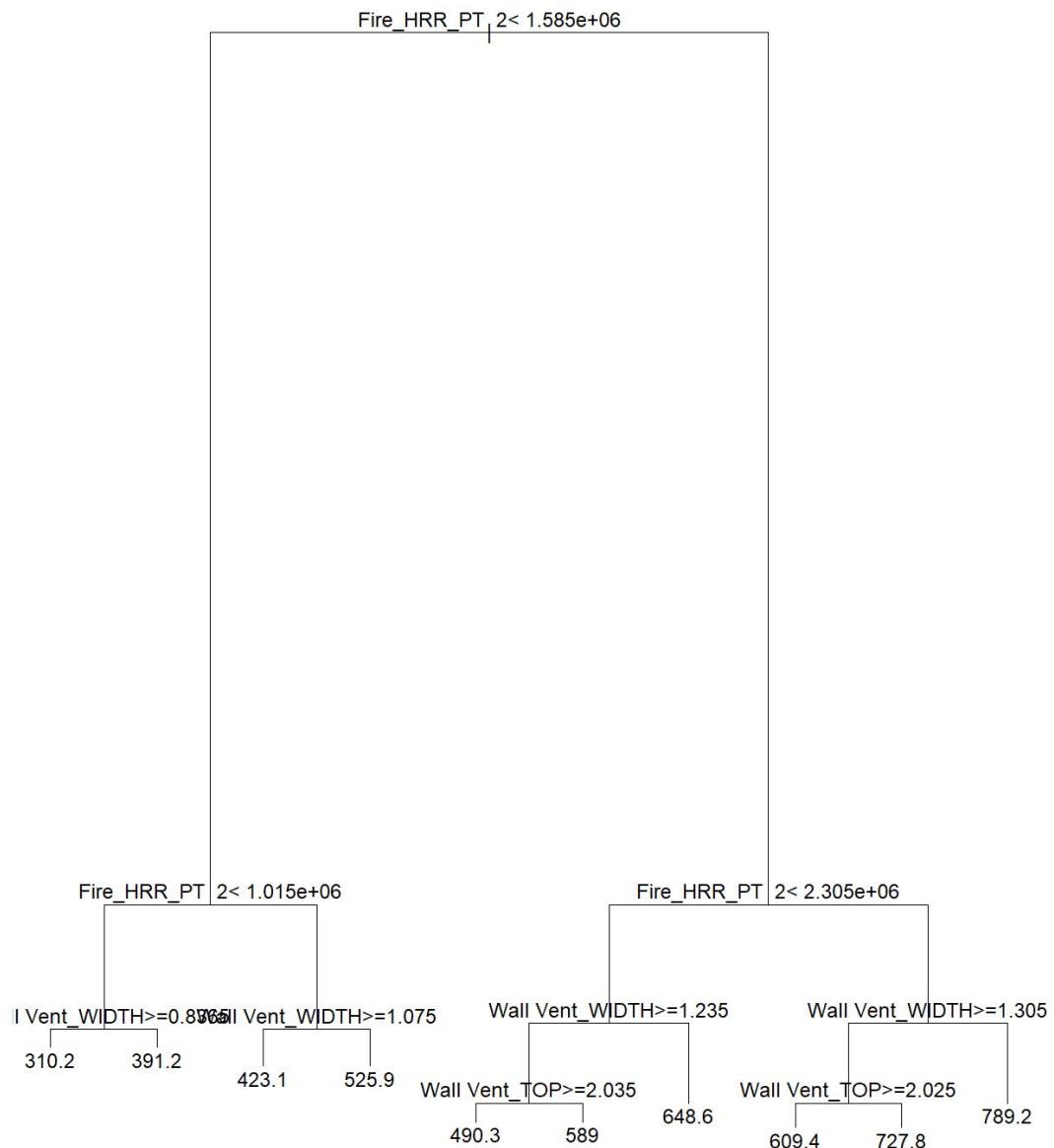


Figure 3.4: Sample of Decision Tree for maximum upper layer temperature



# Chapter 4

## Examples

### 4.1 Example 1: Flashover in a compartment

The occurrence of flashover within a room is of considerable interest since it is perhaps the ultimate signal of untenable conditions within the room of fire origin and a sign of greatly increased risk to other rooms within the building. Many experimental studies of full-scale fires have been performed that quantify the onset of flashover in terms of measurable physical properties. Several approaches have been taken to estimate the onset of flashover within a room. These methods are typically based on simplified mass and energy balances on a single-compartment fire along with correlations to fire experiments. Walton, Thomas, and Ohmiya [23] provide a review of available methods for calculating temperatures in fires in a single compartment with an open door. Three methods are identified from the works of Babrauskas [24], McCaffrey et al. [25] and Thomas [26]. Additional correlations by Babrauskas [24] and Hüggelund [27] are available. This example uses the Monte Carlo capabilities in CData to generate a set of data to compare these correlations to a range of CFAST simulations. Figure 4.1 shows the compartment geometry for the simulations.

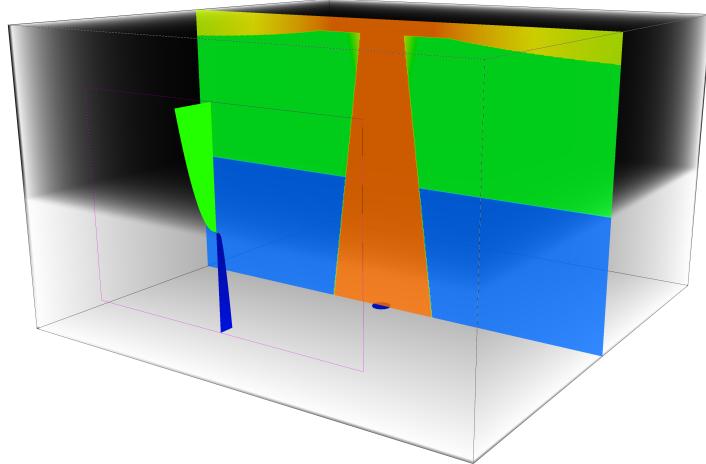


Figure 4.1: Sample CFAST visualization of a single compartment structure used for example 1.

### 4.1.1 CData Inputs

A single compartment with gypsum ceiling and walls and a concrete floor includes a single vent. For this example, the room dimensions, vent dimensions, and fire growth rate are varied as follows:

**Compartment dimensions:** Compartment sizes varied with a uniform distribution scaling all compartment dimensions from 2 m x 2 m x 2 m to 20 m x 20 m x 20 m.

**Vent dimensions:** Vent sizes varied with a uniform distribution scaling vent bottom from 0 m to 1.5 m, a uniform distribution scaling vent top from 1.65 m to 19.85 m limited to 0.15 m below the compartment height, and a uniform distribution scaling vent width from 0.25 m to 19 m limited to 1.0 m less than the compartment width.

**Fire growth rate:** Fire growth is varied with a uniform distribution ranging from 75 s to 1000 s to the peak heat release rate. Peak heat release rate set to 160 MW to ensure flashover in the largest compartments when there is sufficient oxygen.

For the analysis, we will look at the minimum heat release rate necessary to achieve flashover in the compartment defined by a typically-used metrics of 600 °C upper layer temperature [28] [29].

The example below shows how to vary the width uniformly over the range of 2.0 m to 20.0 m. Other dimensions are handled in a similar fashion.

```
&MRND ID = 'Comp 1 Width Generator' DISTRIBUTION_TYPE = 'UNIFORM' VALUE_TYPE  
= 'REAL'  
    MINIMUM = 2.0 MAXIMUM = 20.0/  
&MFLD ID = 'Comp 1 Width' FIELD_TYPE = 'VALUE' RAND_ID = 'Comp 1 Width  
Generator' FIELD = 'Comp 1' 'WIDTH'  
ADD_TO_PARAMETERS = .TRUE. /
```

The vent height and width also vary randomly, but must be limited to the compartment height and width. Here, the maximum value for the top of the vent is defined as 0.15 m below the compartment ceiling (specified by the MAXIMUM\_OFFSET of -0.15 m, with the negative number indicating an offset relative to the top of the compartment) and the vent width maximum is defined as 1 m less than the vent width (specified by the MAXIMUM\_OFFSET of -1.0 m).

```
&MRND ID = 'Wall Vent Bottom Generator' DISTRIBUTION_TYPE = 'UNIFORM'  
VALUE_TYPE = 'REAL'  
    MINIMUM = 0.0 MAXIMUM = 1.5/  
&MFLD ID = 'Wall Vent Bottom' FIELD_TYPE = 'VALUE' RAND_ID = 'Wall Vent  
Bottom Generator' FIELD = 'Wall Vent' 'BOTTOM'  
ADD_TO_PARAMETERS = .TRUE. /  
  
&MRND ID = 'Wall Vent Top Generator' DISTRIBUTION_TYPE = 'UNIFORM' VALUE_TYPE  
= 'REAL'  
    MINIMUM = 1.65 MAXIMUM_FIELD = 'Comp 1' 'Height' MAXIMUM_OFFSET = -0.15/  
&MFLD ID = 'Wall Vent Top' FIELD_TYPE = 'VALUE' RAND_ID = 'Wall Vent Top  
Generator' FIELD = 'Wall Vent' 'TOP'
```

```

ADD_TO_PARAMETERS = .TRUE. /

&MRND ID = 'Wall Vent Width Generator' DISTRIBUTION_TYPE = 'UNIFORM'
  VALUE_TYPE = 'REAL'
    MINIMUM = 0.25 MAXIMUM_FIELD = 'Comp 1' 'Width' MAXIMUM_OFFSET = -1.0/
&MFLD ID = 'Wall Vent Width' FIELD_TYPE = 'VALUE' RAND_ID = 'Wall Vent Width
Generator' FIELD = 'Wall Vent' 'WIDTH'
  ADD_TO_PARAMETERS = .TRUE. /

```

The fire specification is naturally the most complex. For this example, we use a t-squared growth rate fire that grows to a peak value in a time period from 75 s to 1000 s, followed by a 10 s plateau and 10 s decay back to 0 kW. Since both the plateau and decay are set to 10 s, the same generator input can be used for both time points.

```

&MRND ID = 'End of Time Growth Generator', DISTRIBUTION_TYPE = 'UNIFORM'
  VALUE_TYPE = 'REAL'
    MINIMUM = 75 MAXIMUM = 1000 /
&MRND ID = 'Plateau End Time' DISTRIBUTION_TYPE = 'CONSTANT' VALUE_TYPE =
  'REAL' REAL_CONSTANT_VALUE = 10 /
&MRND ID = 'Fire End Time' DISTRIBUTION_TYPE = 'CONSTANT' VALUE_TYPE = 'REAL'
  REAL_CONSTANT_VALUE = 10 /

&MRND ID = 'Peak HRR Generator', DISTRIBUTION_TYPE = 'UNIFORM' VALUE_TYPE =
  'REAL' MINIMUM = 1500000 MAXIMUM = 160000000/
&MRND ID = 'End of Fire HRR' DISTRIBUTION_TYPE = 'CONSTANT' VALUE_TYPE =
  'REAL' REAL_CONSTANT_VALUE = 0 /

&MFIR ID = 'Fire_generator' FIRE_ID = 'New Fire 1' FIRE_TIME_GENERATORS =
  'End of Time Growth Generator'
  'Plateau End Time' 'Fire End Time' FIRE_HRR_GENERATORS = 'Peak HRR
  Generator' 'Peak HRR Generator'
  'End of Fire HRR' NUMBER_OF_GROWTH_POINTS = 20 GROWTH_EXPONENT = 2 /

```

## 4.1.2 Results

Flashover is a term describing the transition of a relatively localized interior fire to one engulfing the entire compartment. It is of interest to the fire service because of the danger to fire fighters and to building designers because of life safety and the attendant impact on occupants. Several papers have looked at the capability of CFAST to predict the conditions under which flashover can occur [30, 31, 32, 33]. In addition, a comparison of CFAST with a number of simple correlations was used by Peacock and Babrauskas [28, 29] to simulate a range of geometries and fire conditions to predict the development of the fire up to the point of flashover. The important test of these prediction methods is in the comparison of the predictions with actual fire observations. Figure 4.2 (reference [29]) presents estimates of the energy required to achieve flashover for a range of room and vent sizes from the CFAST runs above. This figure is an extension of the earlier work of Babrauskas [34] and includes additional experimental measurements from a variety of sources, most notably the work of Deal and Beyler [35]. For a number of the experimental observations,

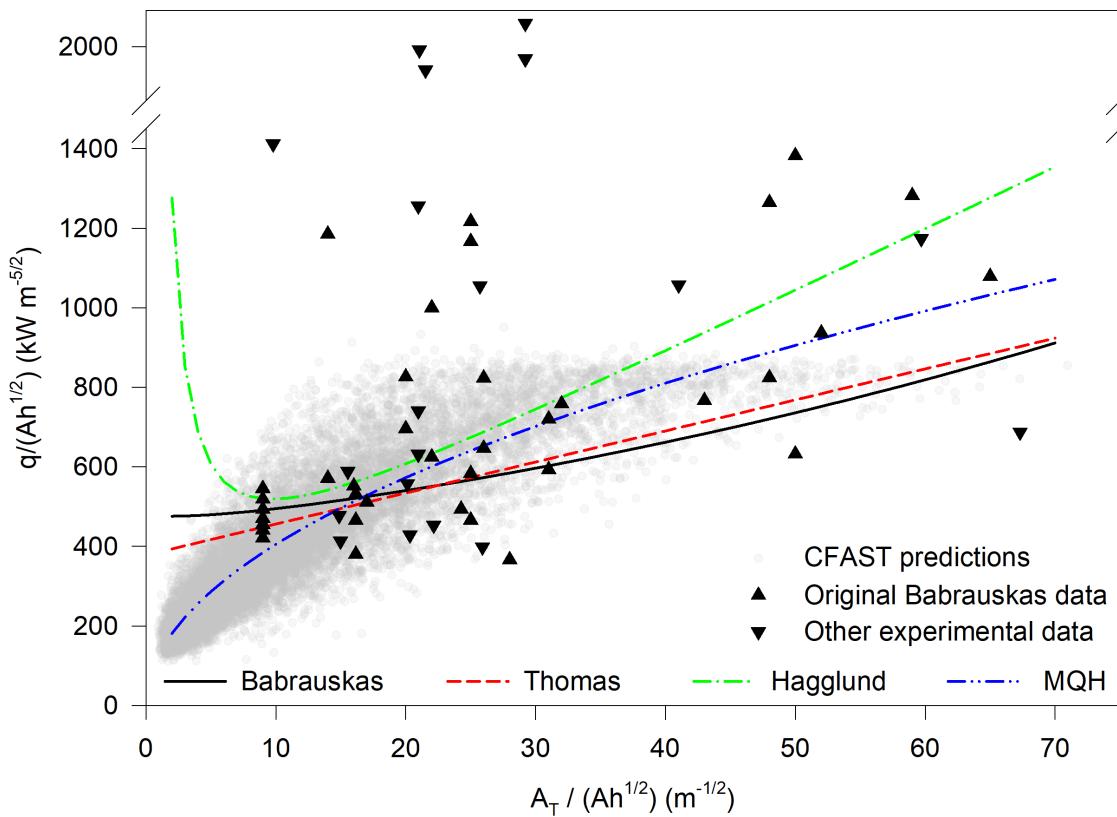


Figure 4.2: Comparison of correlations, CFAST predictions, and experimental data for the prediction of flashover in a compartment fire.

values are included that were not explicitly identified as being a minimum value at flashover.

## 4.2 Example 2: Impact of smoke detector interconnection

Battery-powered smoke alarms with wireless interconnect signal transmission allow for retrofitting of homes with interconnected smoke alarms up to the current code-required installation locations without hardwiring. However, wireless communication draws additional current beyond basic smoke alarm operation and will affect battery life. To receive an interconnect signal, wireless smoke alarms must periodically power up the transceiver and listen for a transmitted signal from another alarm that initially responds to smoke. Naturally a question arises: what is an acceptable maximum delay time for wireless interconnected smoke alarms given the desire to extend battery life?

The script that was written to do the extra analysis in the following subsections for this example is listed in Appendix D.1.

### 4.2.1 CData Inputs

In this simplified example based on earlier work [13], we look at the impact of a periodic activation of wireless communications in detectors. We will use a fixed single-story residential structure for the simulations [36], varying the fire placement and size, the doors being fully open, partitionally open and closed within the structure, and the properties of the smoke detectors placed in each compartment. Figure 4.3 shows the compartment geometry for the simulations.

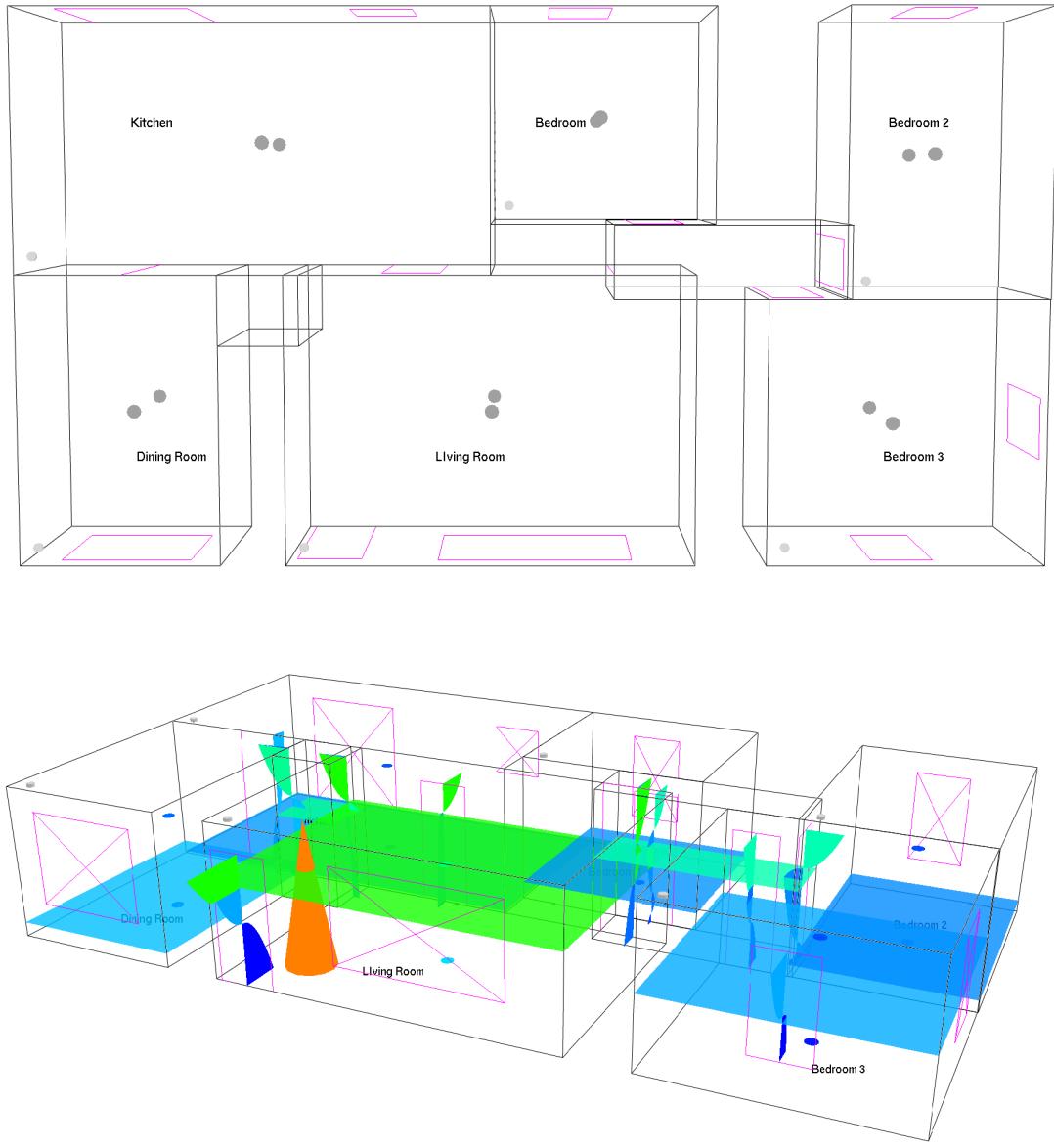


Figure 4.3: Sample CFAST visualization of a single story residential structure used for example 2.

For this example, the fire location, fire growth, fire peak heat release rate, vent opening/closing for all interior vents, and smoke detector sensitivity will be varied as follows:

**Fire location:** Fire is placed in a fixed location in a randomly chosen compartment in the structure and is equally split between flaming and smoldering fires.

**Initial fire growth rate:** Flaming fires grow from zero to  $23 \text{ kW} \pm 7 \text{ kW}$  in  $207 \text{ s} \pm 46 \text{ s}$  using normal distributions, followed by a  $t^2$  growth to  $1054 \text{ kW}$  in  $222 \text{ s} \pm 47 \text{ s}$  again using a normal distribution. Smoldering fires grow from zero to  $11 \text{ kW} \pm 3 \text{ kW}$  in  $6863 \text{ s} \pm 1812 \text{ s}$  using normal distributions, followed by a  $t^2$  growth to  $1054 \text{ kW}$  in  $189 \text{ s} \pm 48 \text{ s}$  again using a normal distribution.

**Interior doors:** Doors between all interior compartments are each randomly set to be initially fully open or with a 2.5 cm undercut the full width of the door. These openings remain unchanged throughout a simulation.

**Smoke detectors:** Smoke detectors follow a statistical smoke alarm activation model developed for upholstered furniture containing polyurethane foam [37]. For this simple example, we use only one detector who activation varies with a log-normal distribution with a geometric mean of  $9.5 \text{ %/m} \pm 4.2 \text{ %/m}$  ( $3.0 \text{ %/ft} \pm 1.3 \text{ %/ft}$ ) for flaming fires and  $15.5 \text{ %/m} \pm 4.2 \text{ %/m}$  ( $5.0 \text{ %/ft} \pm 1.3 \text{ %/ft}$ ) for smoldering fires.

The full file for this case is in Detectors.in. Here are some examples of the inputs. The following gives the full specification of the fire.

```
&MRND ID = 'Generator for fire rooms' DISTRIBUTION_TYPE =
  'USER_DEFINED_DISCRETE' VALUE_TYPE = 'INTEGER'
  INTEGER_VALUES = 1, 2, 3 PROBABILITIES = 0.3333 0.3333 0.3334 /
&MRND ID = 'Flaming Smoldering Generator' DISTRIBUTION_TYPE =
  'USER_DEFINED_DISCRETE' VALUE_TYPE = 'INTEGER'
  INTEGER_VALUES = 1 2 PROBABILITIES = 0.5 0.5 /
&MRND ID = 'Flaming Peak Generator' DISTRIBUTION_TYPE = 'TRUNCATED_NORMAL'
  VALUE_TYPE = 'REAL' MEAN = 23 STDEV = 7
  MINIMUM = 10.0 MAXIMUM = 30.0/
&MRND ID = 'Flaming Time Generator' DISTRIBUTION_TYPE = 'TRUNCATED_NORMAL'
  VALUE_TYPE = 'REAL' MEAN = 207 STDEV = 46
  MINIMUM = 150 MAXIMUM = 1200/
&MRND ID = 'Smoldering Peak Generator' DISTRIBUTION_TYPE = 'TRUNCATED_NORMAL'
  VALUE_TYPE = 'REAL' MEAN = 11 STDEV = 3
  MINIMUM = 10.0 MAXIMUM = 20.0/
&MRND ID = 'Smoldering Time Generator' DISTRIBUTION_TYPE = 'TRUNCATED_NORMAL'
  VALUE_TYPE = 'REAL' MEAN = 6863 STDEV = 1812
  MINIMUM = 4840 MAXIMUM = 10910/
&MRND ID = 'End of Time Growth Generator', DISTRIBUTION_TYPE =
  'TRUNCATED_NORMAL' VALUE_TYPE = 'REAL' MEAN = 205, STDEV = 65
  MINIMUM = 75 MAXIMUM = 500 /
&MRND ID = 'Peak HRR Generator', DISTRIBUTION_TYPE = 'CONSTANT' VALUE_TYPE =
  'REAL' REAL_CONSTANT_VALUE = 1054 /
&MRND ID = 'Plateau End Time' DISTRIBUTION_TYPE = 'CONSTANT' VALUE_TYPE =
  'REAL' REAL_CONSTANT_VALUE = 10 /
&MRND ID = 'Fire End Time' DISTRIBUTION_TYPE = 'CONSTANT' VALUE_TYPE = 'REAL'
  REAL_CONSTANT_VALUE = 10 /
```

```

&MRND ID = 'End of Fire HRR' DISTRIBUTION_TYPE = 'CONSTANT' VALUE_TYPE =
  'REAL' REAL_CONSTANT_VALUE = 0 /

&MFIR ID = 'Compartment_generator'
  FIRE_ID = 'Random'
    FIRE_COMPARTMENT_RANDOM_GENERATOR_ID = 'Generator for fire rooms'
    FIRE_COMPARTMENT_IDS = 'Dining Room', 'Kitchen', 'Living Room'
    FLAMING_SMOLDERING_INCIPIENT_RANDOM_GENERATOR_ID = 'Flaming Smoldering
      Generator'
    NUMBER_OF_INCIPIENT_FIRE_TYPES = 2
    INCIPIENT_FIRE_TYPES = 'FLAMING' 'SMOLDERING'
    TYPE_OF_INCIPIENT_GROWTH = 'RANDOM'
    FLAMING_INCIPIENT_PEAK_RANDOM_GENERATOR_ID = 'Flaming Peak Generator'
    FLAMING_INCIPIENT_DELAY_RANDOM_GENERATOR_ID = 'Flaming Time Generator'
    SMOLDERING_INCIPIENT_DELAY_RANDOM_GENERATOR_ID = 'Smoldering Peak
      Generator'
    SMOLDERING_INCIPIENT_PEAK_RANDOM_GENERATOR_ID = 'Smoldering Time
      Generator'
    FIRE_TIME_GENERATORS = 'End of Time Growth Generator' 'Plateau End Time'
      'Fire End Time'
    FIRE_HRR_GENERATORS = 'Peak HRR Generator' 'Peak HRR Generator' 'End of
      Fire HRR'
    NUMBER_OF_GROWTH_POINTS = 10 GROWTH_EXPONENT = 2 /

```

The doors all follow the same pattern. As an example here is the specification for the DR LR 1 door.

```

&MRND ID = 'Generator for DR LR 1' DISTRIBUTION_TYPE =
  'USER_DEFINED_DISCRETE' VALUE_TYPE = 'INTEGER'
  INTEGER_VALUES = 1, 2, 3, 4, 5 PROBABILITIES = 0.2, 0.2, 0.2, 0.2 0.2 /

&MFLD ID = 'Height DR LR 1' FIELD_TYPE = 'INDEX' RAND_ID = 'Generator for DR
  LR 1' VALUE_TYPE = 'REAL' NUMBER_IN_INDEX = 5
  REAL_VALUES = 2.4, 2.4, 2.4, 2.4 0.025 FIELD = 'DR LR 1', 'TOP'
  ADD_TO_PARAMETERS = .TRUE. /
&MFLD ID = 'Width DR LR 1' FIELD_TYPE = 'INDEX' RAND_ID = 'Generator for DR
  LR 1' VALUE_TYPE = 'REAL' NUMBER_IN_INDEX = 5
  REAL_VALUES = 0.96 0.72 0.48 0.24 0.96 FIELD = 'DR LR 1', 'WIDTH'
  ADD_TO_PARAMETERS = .TRUE. /
&MFLD ID = 'Label DR LR 1' FIELD_TYPE = 'LABEL' RAND_ID = 'Generator for DR
  LR 1' VALUE_TYPE = 'CHARACTER' NUMBER_IN_INDEX = 5
  STRING_VALUES = 'open' 'three-fourth' 'one-half' 'one-fourth' 'closed'
  ADD_TO_PARAMETERS = .TRUE.
  PARAMETER_COLUMN_LABEL = 'DR LR 1 Opening Status'/

```

Notice that the first two &MFLD entries of FIELD\_TYPE = ' INDEX'. They are for the height and width of the door the first 4 of 5 indexes are for the door being 'open', 'three-fourth', 'one-half', and 'one-fourth' open. For these indexes only the width of the door is changed and the height is the full 2.4 m. The fifth index is for the 'closed' door and has the width the full 0.96 m but the height is only 0.025 m to represent the crack under the door. To simplify modeling the data it would be helpful to have a single column that represents the state of the door. This is why the

third &MFLD entry of FIELD\_TYPE = 'LABEL' is included.

Like the doors all the detectors follow the same pattern. All the detectors could have been set up with the same sensitivity for each case but here it was decided that the sensitivity of the detectors would be independent. An example of the CData specification for one detector follows

```
&MRND ID = 'UL Alarm Living Room Trigger Generator' DISTRIBUTION_TYPE =
    'TRUNCATED_LOG_NORMAL' VALUE_TYPE = 'REAL'
    MEAN = 9.5 STDEV = 1.3 MINIMUM = 1.6 MAXIMUM = 41.3/
&MFLD ID = 'UL Alarm Living Room Trigger' FIELD_TYPE = 'VALUE' RAND_ID = 'UL
    Alarm Living Room Trigger Generator'
    FIELD = 'UL Alarm Living Room', 'TRIGGER' /
```

In the analysis, we will look at the impact of delays in triggering of secondary, interconnected smoke detectors on the overall tenability for a range of fires.

#### 4.2.2 Number of Cases

As a first step, we do a preliminary check to see if we have enough cases, Figs. 4.4 and 4.5.

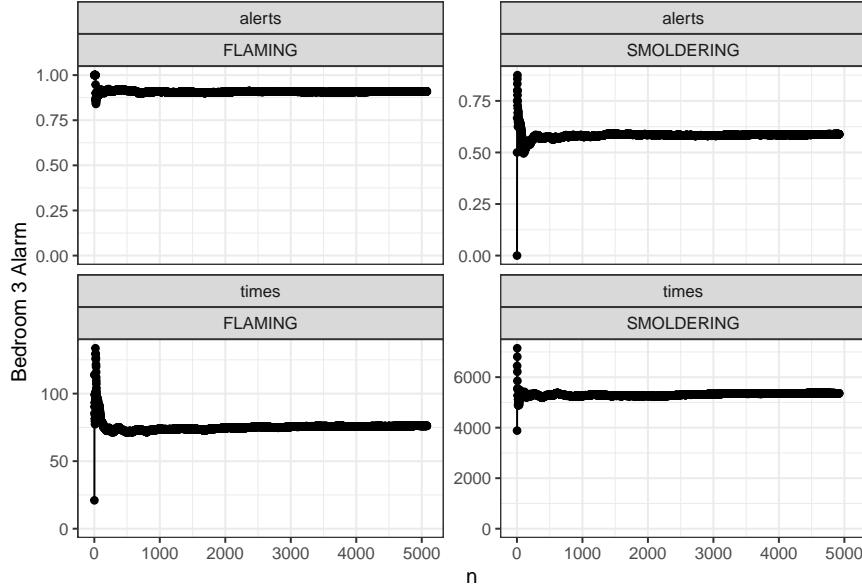


Figure 4.4: Mean percent activation and time to activation for the bedroom 1 alarm as a function of the number of cases. This is a split between flaming (left column) and smoldering (right column) fires showing percent of cases where detector activated (top row) and time to activation (bottom row)

To be meaningful we have to look at one of the output columns; here we look at the Bedroom 1 Alarm data. Looking at the data it becomes very quickly apparent that the results for Flaming type fires and Smoldering type fires are very different. Also, nearly half the cases have no activation within the simulation time. So the evaluation of convergence is evaluated separately for flaming type fires and smoldering type fires. Then the time to activation for cases where activation occurs

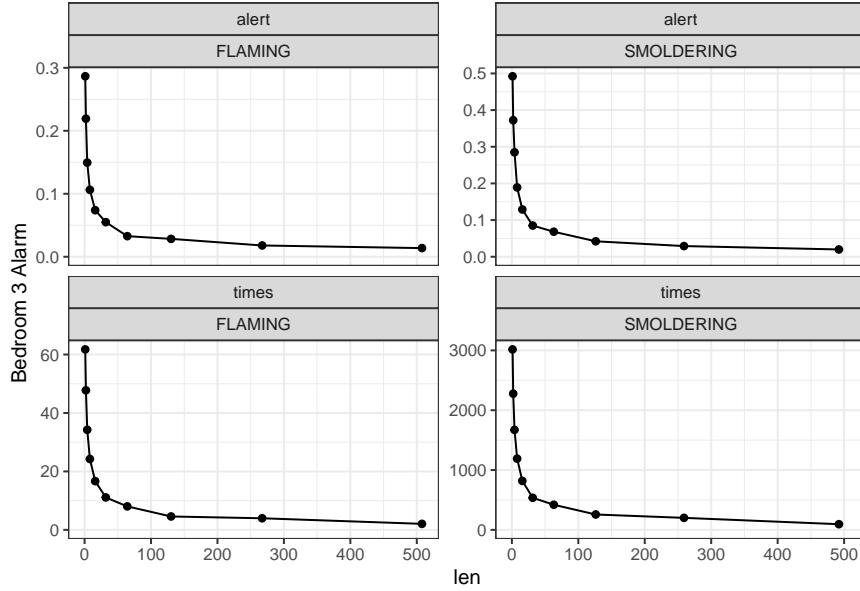


Figure 4.5: Standard deviation of percent activation and time to activation for the bedroom 1 alarm as a function of the number of cases. This is a split between flaming (left column) and smoldering (right column) fires showing percent of cases where detector activated (top row) and time to activation (bottom row)

is evaluated separately from the percent of cases where activation occurs. If any of these do not converge, then we do not have enough cases.

The preliminary results suggest that we have enough cases. The averages all appear to have converged, and the standard errors are small.

### 4.2.3 Analysis

The question explored here is the benefit gained by connecting alarms. The effect of interconnecting alarms is to reduce the amount of time before people in the house are notified of a fire even though they may be far away from the location of the fire. In this example, there are fires in the dining room, kitchen, or living room, and we will assume that people are in the bedrooms. Further assume the people in the bedrooms will not hear an alarm unless it is the one in the bedroom. So the effect of interconnecting alarms is to notify people to a fire when the alarm in one of the front rooms goes off rather than waiting until one of the bedroom alarms sounds.

Assume for this example that there is one alarm in the Living Room and one alarm in Bedroom 1 (which will proxy for all the bedroom alarms). This example, then, examines the difference in activation times between the Living Room Alarm and the Bedroom 1 Alarm.

Exploratory analysis of the data is an important part of any analysis. Here, initial exploration indicated that in a substantial number of cases one or more of the alarms never activated.

Table 4.1 displays the number of cases where each of the alarms activated. Cases marked NA, are cases where the model itself did not converge. That was a very small number of cases for this example.

For more than half the cases, both alarms activated. For most of the remaining cases the Living Room alarm activated but the Bedroom alarm did not. For less than 10 % of the cases neither alarm

Table 4.1: Number of cases by alarm activation status.

	<b>Bedroom 1</b>	<b>No</b>	<b>Yes</b>	<b>NA</b>
Living Room				
No		410	0	0
Yes		2021	7566	0
NA		0	0	3

activated. There were no cases where the bedroom alarm activated but the Living Room alarm did not.

To better understand the cases where the bedroom alarm fails to sound, a decision tree was generated for the bedroom alarm sounding shown in Fig 4.6. The paths to the three leaf nodes with the highest percentages of occurrence are given as follows. Starting at the top of the tree note that if the LR to BR Hall door is closed, the left branch, than the Bedroom 1 alarm doesn't sound and this occurs roughly 20 % of the time. This means that the LR to BR Hall door being only partially open doesn't stop the alarm from sounding but it may delay activation. If the LR to BR Hall door is open any amount but the Bedroom 1 door is closed the alarm will not sound and this happens approximately 16 % of the time. Starting at the top and going down the right branch every time, meaning that the four doors referenced are all open, than Bedroom 1 alarm is activated and this occurs about 41 % of the time. Also note that because of rounding errors adding up the percentages in all the leaf nodes equals 101 %.

It is unusual for a tree to produce results as stark as this one. What becomes clear on looking at the tree is that the state of the intervening doors and the type of fire determines whether the bedroom alarm sounds. If the fire is smoldering and is isolated from the bedroom by closed doors then the alarm will not sound. If the fire is flaming, it must either be in the dining room or kitchen with at least one door closed between the fire and the alarm or if it is in the living room, there must be two closed doors between it and the alarm. Otherwise the alarm will go off.

Interconnected alarms do not monitor continuously. Rather they check periodically to see if some other alarm has sounded. For this example, we will assume they check every 60 s. The effect is that there is a random delay, with the delay drawn from a uniform random distribution of between 0 s and 60 s. Adding the random delay is not done in CFAST but in R during the analysis. A Kernel Density estimate of the distribution of time savings is shown in Fig 4.7, with fire type on separate charts. Any negative “time delay” means that the bedroom alarm sounds on its own before it gets notification from the Living Room alarm.

With 10 000 data points we can empirically estimate the average time savings that interconnected alarms would provide, as well as median time savings and various quantiles. We are interested in the quantiles because adverse outcomes, like death or injury, are tail events. So we also look at the tails of the distribution. Here the “1 % Quantile” means that 1 % of alarms saved more time than this (see Table 4.2).

This looks only at cases where the Bedroom 1 alarm sounded. If this were a serious attempt to identify the effect of interconnected alarms then the percent of non-activations would need to be

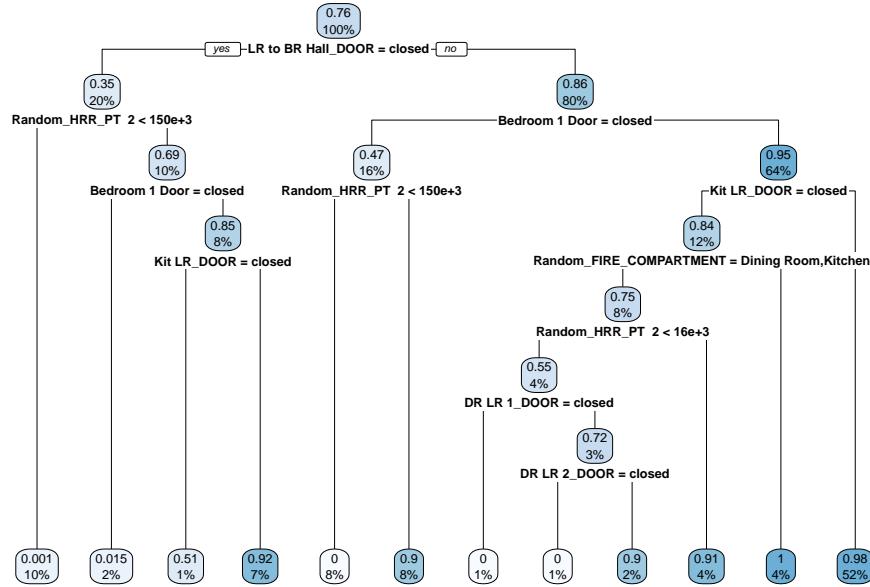


Table 4.2: Quantiles of delay time for bedroom alarm activation

Result	Flaming	Smoldering
Mean	101.7	1821.8
Median	96.0	1597.7
25 % Quantile	136.3	2936.7
10 % Quantile	176.4	4139.8
5 % Quantile	199.5	4792.5
1 % Quantile	238.3	6109.8

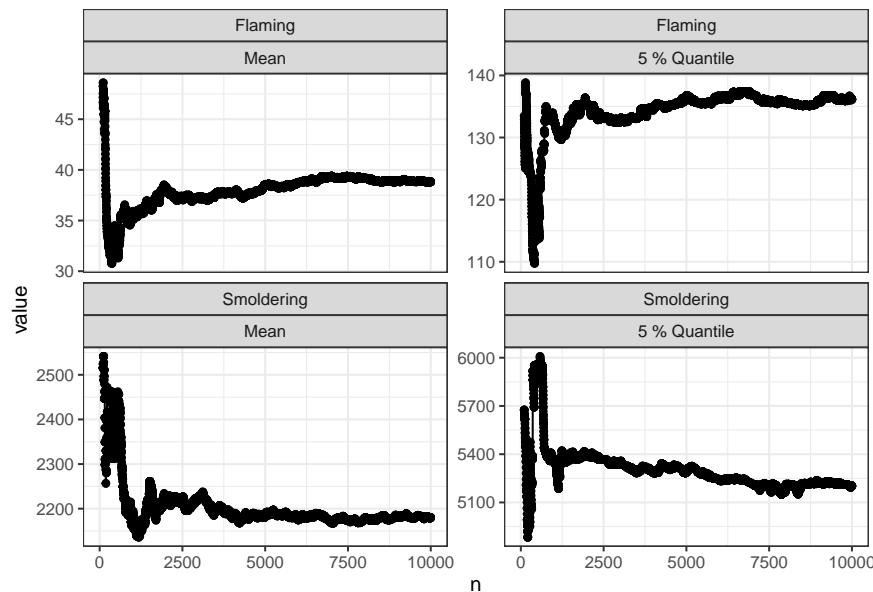


Figure 4.8: Quantiles of the time savings as a function of the number of cases.

While the preliminary results look settled down within a couple of thousand cases, these results take longer. In particular the extreme quantiles take somewhat longer to converge. In particular, the 5 % quantile value doesn't really settle down until we have 7500 cases or so. However, looking at the results it looks like our results have settled down sufficiently by the 10 000 cases we actually use to have reasonable confidence in our results.

#### 4.2.5 Discussion

For a study like this it is important to get the distribution of fire parameters correct. For example, if the proportion of fires starting in the Kitchen were low compared to the real world then all of the time-delay values would be underestimated. If the likelihood of closed doors were overestimated, then there would be far fewer cases of alarm failure for the bedroom alarms, and the benefit of interconnection would be lower.

## 4.3 Example 3: Model Sensitivity

In this example, we look at a simple fire model sensitivity analysis, which ASTM E 1355 defines as a study of how changes in model parameters affect the results [1]. In other words, sensitivity refers to the rate of change of the model output with respect to input variations. The standard also indicates that model predictions may be sensitive to (1) uncertainties in input data, (2) the level of rigor employed in modeling the relevant physics and chemistry, and (3) the accuracy of numerical treatments. Thus, the purpose of a sensitivity analysis is to assess the extent to which uncertainty in the model inputs is manifested as uncertainty in the model results of interest. For this simple example, we will use a single office building and look at the impact of changing important model inputs by  $\pm 10\%$ . Figure 4.9 shows the compartment geometry for the example.

A sensitivity analysis involves defining a base case scenario, and varying selected input parameters. The resultant variations in the model output are then measured with respect to the base case scenario in order to consider the extent to which uncertainty in model inputs influences model output. Therefore, a sensitivity analysis of CFAST should account for variations in the extensive number of input parameters that describe the building geometry, compartment connections, construction materials, and description of one or more fires.

The script that was written to do the extra analysis in the following subsections for this example is listed in Appendix D.2.

### 4.3.1 CData Inputs

For this example, we will independently vary the following inputs by  $\pm 10\%$ , recognizing that these are not an exhaustive list of input variables, but which should represent the major model inputs to a simulation:

**Compartment geometry:** Compartment size will be scaled to  $\pm 10\%$  of base size in all dimensions.

**Compartment materials:** Thermal properties for all materials will be varied by  $\pm 10\%$  of base values for thermal conductivity, density, and specific heat. Thickness will be varied by  $\pm 10\%$  of base values.

**Compartment doors:** The size of compartment doors, both interior and exterior, will be varied by  $\pm 10\%$  of base values for height and width. All interior vents will be open for this example.

**Mechanical ventilation:** For mechanical ventilation, the height of each vent, area of each vent, and fan flowrate will each be varied by  $\pm 10\%$  of base values.

**Fire:** Fire is in one of the rooms. Peak heat release rate of the fire and time of the peak will be varied by  $\pm 10\%$  of base value. Yields of carbon monoxide and soot will each be varied by  $\pm 10\%$  of base value. Position of the fire is varied by  $\pm 10\%$  of base values in width and depth, and height.

**Detectors:** Sensitivity of heat detectors will be varied by  $\pm 10\%$  of base values for activation temperature and RTI. Smoke detector activation obscuration will be varied by  $\pm 10\%$  of base values.

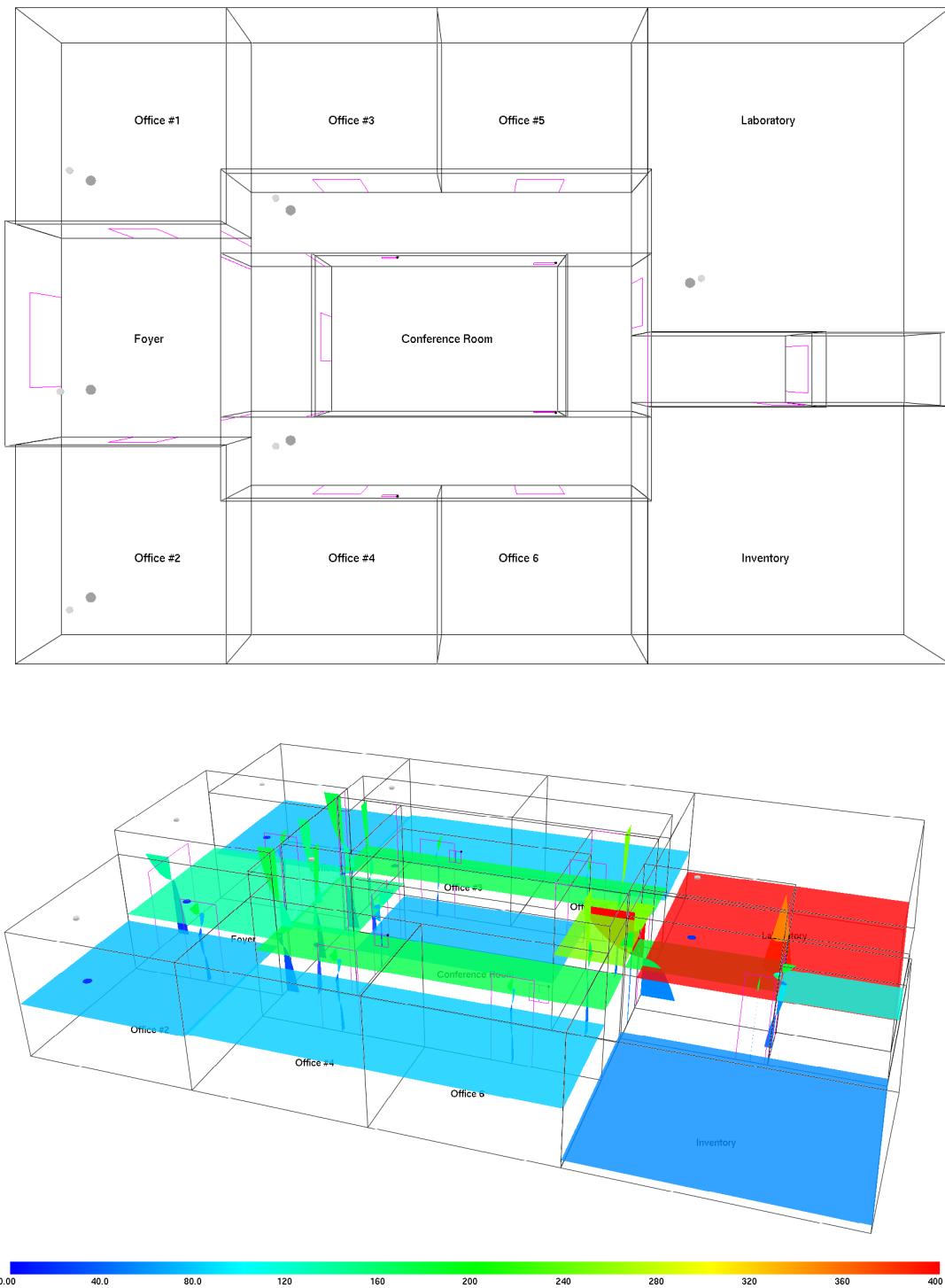


Figure 4.9: Sample CFAST visualization of a single story commercial structure used for example 3.

In the analysis, we will look at typically-used outputs to determine those which are more sensitive (a 10 % change in an input value leads to more than a 10 % change in an output) and less sensitive (a 10 % change in an input value leads to less than a 10 % change in an output) to a

change in selected model inputs to identify the relative importance of selected model inputs to the results of a simulation.

Since this case is looking at varying almost everything that can be varied, it is the largest file of the three examples. It is possible to actually vary the generators over the range that the variable can take but that means a lot of data entry that could easily lead to errors that would be difficult to find. One FIELD\_TYPE is the SCALING type. Using this greatly increases the speed of creating the input file.

As part of the inputs one of the things to keep in mind is that the foyer and halls should be the same height because they are all essentially the same space even though in CFAST they are modeled as a number of compartments and connections. The result is that there is a single random generator that is used by all the rooms and connections making up the halls of the structure as can be seen in the following input snippet.

```
&MRND ID = 'Scaling height', FYI = 'Scaling for everything that has the same
max height', DISTRIBUTION_TYPE = 'UNIFORM',
    VALUE_TYPE = 'REAL', MINIMUM = 0.9 MAXIMUM = 1.1 /
&MFLD ID = 'Foyer height' FIELD_TYPE = 'SCALING' FIELD = 'Foyer'  'HEIGHT'
    RAND_ID = 'Scaling height' ADD_TO_PARAMETERS = .TRUE. PARAMETER_HEADER =
        'Foyer_and_Halls_height' /
&MFLD ID = 'Even Hallway height' FIELD_TYPE = 'SCALING' FIELD = 'Even
    Hallway'  'HEIGHT' RAND_ID = 'Scaling height' /
&MFLD ID = 'Odd Hallway height' FIELD_TYPE = 'SCALING' FIELD = 'Odd Hallway'
    'HEIGHT' RAND_ID = 'Scaling height' /
&MFLD ID = 'Front Hall height' FIELD_TYPE = 'SCALING' FIELD = 'Front Hall'
    'HEIGHT' RAND_ID = 'Scaling height' /
&MFLD ID = 'Back Hall height' FIELD_TYPE = 'SCALING' FIELD = 'Back Hall'
    'HEIGHT' RAND_ID = 'Scaling height' /
&MFLD ID = 'Bathroom Hallway height' FIELD_TYPE = 'SCALING' FIELD = 'Bathroom
    Hall'  'HEIGHT' RAND_ID = 'Scaling height' /
&MFLD ID = 'Foyer2FrontHall height' FIELD_TYPE = 'SCALING' FIELD =
    'Foyer2FrontHall'  'TOP' RAND_ID = 'Scaling height' /
&MFLD ID = 'Foyer2EvenHall height' FIELD_TYPE = 'SCALING' FIELD =
    'Foyer2EvenHall'  'TOP' RAND_ID = 'Scaling height' /
&MFLD ID = 'Foyer2OddHall height' FIELD_TYPE = 'SCALING' FIELD =
    'Foyer2OddHall'  'TOP' RAND_ID = 'Scaling height' /
&MFLD ID = 'FrontHall2EvenHall height' FIELD_TYPE = 'SCALING' FIELD =
    'FrontHall2EvenHall'  'TOP' RAND_ID = 'Scaling height' /
&MFLD ID = 'FrontHall2OddHall height' FIELD_TYPE = 'SCALING' FIELD =
    'FrontHall2OddHall'  'TOP' RAND_ID = 'Scaling height' /
&MFLD ID = 'BackHall2EvenHall height' FIELD_TYPE = 'SCALING' FIELD =
    'BackHall2EvenHall'  'TOP' RAND_ID = 'Scaling height' /
&MFLD ID = 'BackHall2OddHall height' FIELD_TYPE = 'SCALING' FIELD =
    'BackHall2OddHall'  'TOP' RAND_ID = 'Scaling height' /
```

After all the fields that are to be varied are set up there is also the fire to vary. Instead of varying the fire we will scale the HRR and time with single independent scaling factors. This requires only two random generators and is implemented as follows.

```

&MRND ID = 'Scaling HRR', FYI = 'Just making sure it works',
DISTRIBUTION_TYPE = 'UNIFORM',
VALUE_TYPE = 'REAL', MINIMUM = 0.9 MAXIMUM = 1.1 /
&MRND ID = 'Scaling Time', FYI = 'Just making sure it works',
DISTRIBUTION_TYPE = 'UNIFORM',
VALUE_TYPE = 'REAL', MINIMUM = 0.9 MAXIMUM = 1.1 /
&MFIR ID = 'Scale fire' FIRE_ID = 'Fire' BASE_FIRE_ID = 'Back Fire'
SCALING_FIRE_HRR_RANDOM_GENERATOR_ID = 'Scaling HRR'
SCALING_FIRE_TIME_RANDOM_GENERATOR_ID = 'Scaling Time'
ADD_HRR_SCALE_TO_PARAMETERS = .TRUE. ADD_TIME_SCALE_TO_PARAMETERS =
.TRUE. /

```

The other advantage, besides ease of implementation, is that it eliminates the possibility of variation of individual time points reducing the impact of the variability. For example if one point has the HRR increasing by some amount but the next HHR value decreases it won't completely average out but it could reduce the impact on the overall scenario.

### 4.3.2 Approach

For this example it is assumed that the output variables of interest are 'Foyer Max Heat FED' and 'Foyer Heat FED'. Notionally, we might say that we are interested in the amount of time people have to escape (represented by the time during which they foyer-through which they must evacuate—remains viable) and the worst temperature they might have to face in the foyer during the evacuation.

Sensitivity analysis is interested in the sensitivity the output has to each input. This is typically expressed in terms of the percent variation in output produced by a one percent variation in each input. As it turns out, simple linear regression of the *log* of the inputs against the *log* of the output will give exactly that value.

The second target variable—Foyer Heat FED—was selected to illustrate an additional complication in the analysis. This variable expresses the time till the FED reaches the value of 1, the point where only 50 % of people survive. The complication is that in a number of cases non-viability does not occur within the time of the simulation; here some 89.7 % of cases do not result in non-viability. That presents a problem because analyzing the data without taking that into account will result in incorrect estimates of the sensitivity.

In this case what is run is a 'Tobit' analysis[38] which properly accounts for the effect of the cases where non-viability did not occur. The 'Tobit' analysis is also performed on a log-log transformation of the data.

### 4.3.3 Number of Cases

An initial run of 1000 cases was evaluated. Figure 4.10 shows the coefficient for Lab\_depth for the first model as a function of the number of cases. Examination of the chart shows that the value for the coefficient is still changing substantially at the end of 1000 cases. When the number of cases were increased to 2500 the results still did not seem very stable. When the cases increased again to 5000 there was a noticeable trend over the last thousand or so cases. When the cases were increased to 10 000, the estimated value over the last couple of thousand cases seems to have settled down. So these results are based on 10 000 cases.

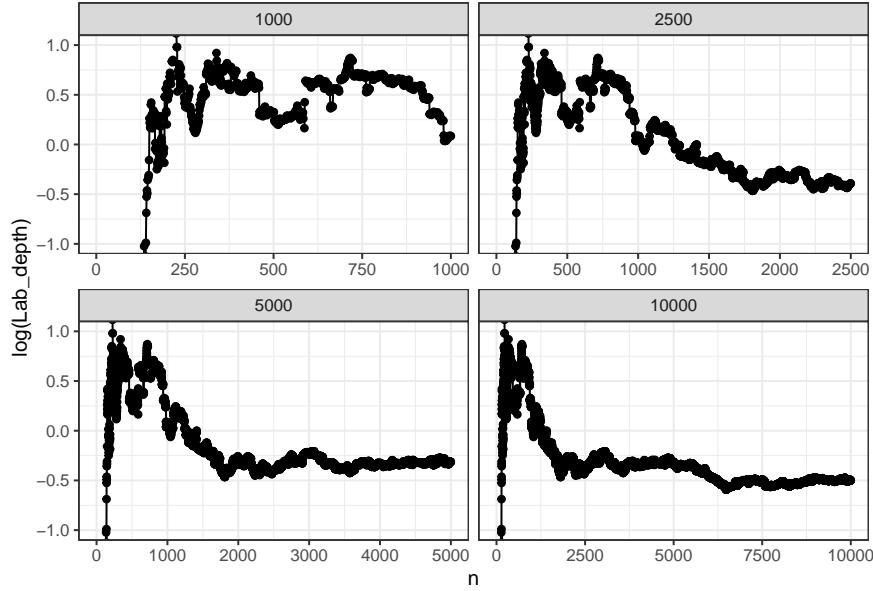


Figure 4.10: Convergence by the number of runs.

#### 4.3.4 Discussion

First we look at the sensitivity of maximum temperature FED in the Foyer. Selected results are displayed in Table 4.3. The single most significant factor is the depth of the lab space—where the fire occurred. A one-percent change in the lab depth will produce a 0.53 % decrease in the maximum heat FED for the Foyer. Similarly a one percent change in the height of the Foyer and halls produces a 0.47 % decrease in the heat FED for the Foyer. At the other end of the scale changes in the Foyer width or the depth of office 1 produce minimal changes to the heat FED for the Foyer

In Table 4.4, we look at just those variables for which a one-percent change produces a change of greater than 0.2 % in the time to non-viability for the Foyer, plus the scaling factors for the fire. Those variables include variables related to the room of fire origin (Lab\_depth), factors related to the space itself (Foyer\_width, Front\_Door\_Width), factors related to the spaces through which the fire must travel (Even\_Hallway\_depth, Odd\_Hallway\_width, Front\_Hall\_depth) and some additional miscellaneous factors (Office\_.2\_height Office\_.5\_height Gyp\_Conductivity.1 Gyp\_Density.1).

Table 4.3: Selected result for sensitivity of maximum temperature FED.

<b>Variable</b>	<b>Value</b>	<b>Std. Error</b>	<b>t value</b>	<b>Pr(&gt; t )</b>	
log(Lab_depth)	-0.5328	0.16	-3.33	0.0009	***
log(Front_Hall_depth)	0.1206	0.16	0.75	0.4551	
log(Foyer_width)	0.0053	0.16	0.03	0.9735	
log(Office_#1_depth)	-0.0042	0.16	-0.03	0.9791	
log(Office_#2_height)	0.4525	0.16	2.84	0.0045	**
log(Foyer_and_Halls_height)	-0.4661	0.16	-2.94	0.0033	**
log(FrontHall2EvenHall_width)	-0.0054	0.16	-0.03	0.9727	
log(Gyp_Emissivity.1)	0.1223	0.16	0.76	0.4458	
log(Office_#4_Door_Height)	-0.0029	0.16	-0.02	0.9856	
log(Office_#6_Door_Height)	0.1209	0.16	0.75	0.4512	
log(Fire_HRR_scaling_factor)	0.0398	0.16	0.25	0.8051	
log(Fire_time_scaling_factor)	-0.3920	0.16	-2.45	0.0141	*

Table 4.4: Selected results for sensitivity of time to non-viability for Foyer Heat FED.

<b>Variable</b>	<b>Value</b>	<b>Std. Error</b>	<b>t value</b>	<b>Pr(&gt; t )</b>	
log(Lab_depth)	0.3460	0.14	2.40	0.0162	*
log(Even_Hallway_depth)	0.3043	0.14	2.12	0.0341	*
log(Odd_Hallway_width)	0.2359	0.14	1.64	0.1011	
log(Front_Hall_depth)	-0.2318	0.15	-1.59	0.1116	
log(Foyer_width)	-0.2778	0.15	-1.91	0.0555	.
log(Office_#2_height)	-0.2103	0.14	-1.46	0.1431	
log(Office_#5_height)	0.2564	0.14	1.78	0.0753	.
log(Gyp_Conductivity.1)	0.2454	0.14	1.70	0.0889	.
log(Gyp_Density.1)	0.2217	0.14	1.54	0.1233	
log(Front_Door_Width)	0.3057	0.14	2.13	0.0333	*
log(Fire_HRR_scaling_factor)	-0.0462	0.15	-0.32	0.7510	
log(Fire_time_scaling_factor)	0.1927	0.14	1.34	0.1790	

# Chapter 5

## Summary

This publication introduces the CFAST Fire Data Generator, CData, that has been developed as a tool to help users do Monte Carlo analyses. The various parts of CData are described and then a very simple analysis is first presented to demonstrate the different parts of the analysis that CData can assist in performing. This is followed by three slightly more complex analyses to further demonstrate the uses of CData.

CData supports several statistical analyses with the statistical software R to provide several built-in tools to analyze the data. Three of the tools are useful in determining that enough cases have been run to draw justifiable conclusions from the analysis. The first, convergence of the mean, allows the user to make a determination if enough cases have been run. Two other tools, the histogram generator and the empirical probability density function, allow users to look at the distributions of input variables to determine that distributions are as expected. Both can also be used to look at the distributions of output variables. Finally, the decision trees model can be used to look for the most important variables in determining the value of a particular output variable. These tools as well as the R software distributed with CData will allow users to do a significant amount of analysis of data that is generated.

Three simple examples were also explored using CData. While these examples do not demonstrate all the capability of CData it is hoped that they do give a flavor of the kinds of problems that can be addressed as well as examples of some of the basic analysis that should always be done. It is important to always check that enough runs have been done so that some confidence can be had in the results. However, it is not very useful to run many times the number of runs needed to draw statistically justifiable conclusions. For one thing the computational time and storage space, especially the storage space, can overwhelm any system if the number of cases is large enough.

The plan with CData moving forward is to provide support and address bugs as they arise while monitoring the kinds of problems users use the program to solve as well as note other comments about the software. Once a picture of how people are using CData and what needs they have emerges, a plan for a next version of the tool will be made.



# References

- [1] American Society for Testing and Materials, West Conshohocken, Pennsylvania. *ASTM E 1355-04, Standard Guide for Evaluating the Predictive Capabilities of Deterministic Fire Models*, 2004. [iii](#), [2](#), [45](#)
- [2] Wikipedia contributors. Monte carlo method. [https://en.wikipedia.org/wiki/Monte\\_Carlo\\_method](https://en.wikipedia.org/wiki/Monte_Carlo_method). Accessed: 2020-08-21. [1](#)
- [3] N. Metropolis. The beginning of the monte carlo method. *Los Alamos Science*, pages 125–130, 1987. 1987 Special Issue dedicated to Stanislaw Ulam. [1](#)
- [4] R.W. Bukowski. Evaluation of furniture-fire hazard using a hazard-assessment computer model. *Fire and Materials*, 9(4):159–166, 1985. [1](#), [2](#)
- [5] F.W. Clarke, R.W. Bukowski, R.W. Steifel, and J.R. Hall. The National Fire Risk Assessment Project - Final Report. Report, National Fire Protection Research Foundation, 1990. [1](#), [2](#)
- [6] NFPA 5000, *Building and Construction Code*. National Fire Protection Association, 2018. [1](#), [2](#)
- [7] M.J. Hurley and E.R. Rosenbaum. *Performance-Based Design*, volume 2, book section 37. Society of Fire Protection Engineers, New York, NY, fifth edition edition, 2016. [1](#), [2](#)
- [8] K. A. Notarianni. *The Role of Uncertainty in Improving Fire Protection Regulation*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 2000. [1](#)
- [9] M.C. Bruns. Predicting the effects of barrier fabrics on residential upholstered furniture fire hazard. Technical Note 1920, National Institute of Standards and Technology, Gaithersburg, Maryland, July 2016. [1](#), [2](#)
- [10] P.A. Reneke, M.C. Bruns, S.W. Gilbert, C. MacLaren, R.D. Peacock, T.G. Cleary, and D.T. Butry. Toward a Process to Quantify the Hazard of Fire Protection Design Alternatives. Report NIST TN 2041, National Institute of Standards and Technology, 2018. [1](#)
- [11] P.A. Reneke, M.C. Burns, R.D. Peacock T.G. Cleary, and C. MacLaren. Conceptual system for evaluation the impact on residential life safety of new fire protection technology. In *The 2017 Suppression, Detection, and Signaling Research and Applications Conference/ 16th International Conference on Fire Detection*. National Fire Protection Association, 2017. [1](#)

- [12] P.A. Reneke, M.C. Burns, R.D. Peacock, T.G. Cleary, and S.W. Gilbert. Development of Tools to Evaluate the Impact of New Residential Fire Protection Technology. In *Suppression, Detection and Signaling Research and Applications Symposium*. NFPA, 2018. 1
- [13] T.G. Cleary., Reneke P.A., and R.D. Peacock. Assessment of wireless smoke alarm interconnect signal delay on safe egress. In *Suppression, Detection and Signaling Research and Application Symposium*, 2019. 1, 37
- [14] R. D. Peacock, P. A. Reneke, and G. P. Forney. CFAST – Consolidated Fire And Smoke Transport (Version 7) Volume 2: User’s Guide. Technical Note 1889v2, National Institute of Standards and Technology, Gaithersburg, Maryland, November 2015. 1
- [15] R.D. Peacock, K.B. McGrattan, G.P. Forney, and P.A. Reneke. CFAST – Consolidated Fire And Smoke Transport (Version 7) Volume 1: Technical Reference Guide. Technical Note 1889v1, National Institute of Standards and Technology, Gaithersburg, Maryland, November 2015. 2
- [16] R. D. Peacock and P. A. Reneke. CFAST – Consolidated Fire And Smoke Transport (Version 7) Volume 3: Software Development and Model Evaluation Guide. Technical Note 1889v3, National Institute of Standards and Technology, Gaithersburg, Maryland, November 2015. 2
- [17] R.D. Peacock. CFAST – Consolidated Fire And Smoke Transport (Version 7) Volume 4: Configuration Management Guide. Technical Note 1889v4, National Institute of Standards and Technology, Gaithersburg, Maryland, November 2015. 2
- [18] American housing survey 2015, 2017. 3, 7
- [19] C. Wade, G. Baker, K. Frank, A. Robbins, R. Harrison, M. Spearpoint, and C. Fleischmann. B-risk user guide and technical manual. Technical Report SR 282, Branz, Ltd., 2013. 4
- [20] V. Babrauskas and R. D. Peacock. Heat Release Rate: The Single Most Important Variable in Fire Hazard. *Fire Safety Journal*, 18:255, 1992. 8
- [21] H. A. Sturges. The choice of a class interval. *Journal of the American Statistical Association*, 21(153):65–66, 1926. 27
- [22] Trevor Haste, Robert Tibshirani, and J. H. Friedman. Springer Series in Statistics. Springer, New York, NY, 2 edition, 2009. 28, 30
- [23] W.D. Walton, P.H. Thomas, and Y. Ohmiya. *SFPE Handbook of Fire Protection Engineering*, chapter Estimating Compartment Fire Temperatures, pages pp. 996 – 1023. Society of Fire Protection Engineering, 2016. 33
- [24] V. Babrauskas. Estimating room flashover potential. *Fire Technology*, 16(2):pp. 94–103, 1980. 33
- [25] B. J. McCaffrey, J. G. Quintiere, and M. F. Harkleroad. Estimating Room Temperatures and the Likelihood of Flashover Using Fire Test Data Correlations. *Fire Technology*, 17(2):98–119, 1981. 33

- [26] P. H. Thomas. Testing Products and Materials for Their Contribution to Flashover in Rooms. *Fire and Materials*, 5:103–111, 1981. [33](#)
- [27] B. Hägglund. Estimating Flashover Potential in Residential Rooms. FOA Rapport C 202369-A3, Forsvarets Forkningsanstalt, 1980. [33](#)
- [28] R. D. Peacock, P. A. Reneke, R. W. Bukowski, and V. Babrauskas. Defining Flashover for Fire Hazard Calculations. *Fire Safety Journal*, 32(4):331–345, 1999. [34](#), [35](#)
- [29] V. Babrauskas, R. D. Peacock, and P. A. Reneke. Defining Flashover for Fire Hazard Calculations: Part II. *Fire Safety Journal*, 38:613–622, 2003. [34](#), [35](#)
- [30] W. Chow. Predicability of Flashover by Zone Models. *Journal of Fire Sciences*, 16:335, September/October 1988. [35](#)
- [31] M. Luo, Y. He, and V. Beck. Application of Field Model and Two-Zone Model to Flashover Fires in a Full-Scale Multi-Room Single Level Building. *Fire Safety Journal*, 29:1, 1997. [35](#)
- [32] P. Collier. Fire in a Residential Building: Comparisons Between Experimental Data and a Fire Zone Model. *Fire Technology*, 32:195, August/September 1996. [35](#)
- [33] D. White, C. Beyler, J. Scheffrey, and F. Williams. Modeling the Impact of Post-Flashover Shipboard Fires on Adjacent Spaces. *Journal of Fire Protection Engineering*, 10, 2000. [35](#)
- [34] V. Babrauskas. Upholstered Furniture Room Fires—Measurements, Comparison with Furniture Calorimeter Data, and Flashover Predictions. *Journal of Fire Sciences*, 2(1):5–19, January/February 1984. [35](#)
- [35] S. Deal and C. Beyler. Correlating Preflashover Room Fire Temperatures. *Journal of Fire Protection Engineering*, 2(2):33–48, 1990. [35](#)
- [36] D. Madrzykowski and C. Weinschenk. Impact of Fixed Ventilation on Fire Damage Patterns in Full-Scale Structures. Technical report, Underwriters Laboratories Firefighter Safety Research Institute, Columbia, MD, April 2019. [37](#)
- [37] T.G. Cleary and R.D. Peacock. A statistical model for smoke alarm activation in upholstered furniture fires. In *Suppression, Detection and Signaling Research and Applications Conference SUPDET 2017*, 2017. [38](#)
- [38] J. Tobin. Estimation of relationships for limited dependent variables. *Econometrica*, 26(1):24–26, 1958. [48](#)



# Appendix A

## Nomenclature

Note that the units associated with a given symbol are sometimes changed upon input to and output from the program. In particular, temperatures are typically input in degrees Celsius, converted to Kelvin, and then converted back to Celsius on output. Energy units involving Joules or Watts are typically input as kJ or kW, converted to J or W, then converted back to kJ or kW.

$A$	area, vent area, $\text{m}^2$
$A_T$	total surface area of a compartment, $\text{m}^2$
$c_p$	heat capacity of air at constant pressure, $\text{J}/(\text{kg}\cdot\text{K})$ or $\text{kJ}/(\text{kg}\cdot\text{K})$
$D$	fire diameter, m
$g$	gravitational constant, $9.8 \text{ m/s}^2$
$h$	vent height, m
$\dot{Q}$	total heat release rate of the fire, kW
$\dot{Q}^*$	Fire Froude Number, $\dot{Q}^* = \frac{\dot{Q}}{\rho_\infty c_p T_\infty \sqrt{g D D^2}}$
$T$	temperature, K
$T_\infty$	ambient gas temperature, K
$\rho$	density, $\text{kg/m}^3$
$\rho_\infty$	ambient density of air, $1.2 \text{ kg/m}^3$



# Appendix B

## CData/CFAST Text-based Input File

The operation of CData and CFAST is based on a single ASCII<sup>1</sup> text file containing parameters organized into *namelist*<sup>2</sup> groups. The input file provides CFAST with all of the necessary information to describe the scenario. The graphical user interface, CEdit, writes this file. This appendix details all the parameters, which are organized into groups that roughly coincide with the tabs in the graphical user interface.

### B.1 Naming the Input File

The input file is saved with a name such as `job_name.in`, where `job_name` is any character string that helps to identify the simulation. All of the output files associated with the calculation will then have this common prefix name.

There should be no blank spaces in the job name. Instead use the underscore character to represent a space.

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

### B.2 Namelist Formatting

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

```
&TIME  SIMULATION = 3600., PRINT = 50., SMOKEVIEW = 50., SPREADSHEET = 50. /
```

sets various values of parameters contained in the TIME namelist group. The meanings of these various parameters is explained in this guide. The namelist records can span multiple lines in the

---

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

<sup>2</sup>A *namelist* is a Fortran input record.

input file, but just be sure to end the record with a slash or else the data will not be understood. Do not add anything to a namelist line other than the parameters and values appropriate for that group. Otherwise, CFAST will stop immediately upon execution.

Parameters within a namelist record can be separated by either commas, spaces, or line breaks. It is recommended that you use commas or line breaks, and never use tab stops because they are not explicitly defined in the namelist data structure. CFAST and CEdit expect the first character of the file to be an ampersand, &, and by convention the first namelist is the `HEAD` namelist but any namelist can be the first. Comments and notes can be written into the file between namelists so long as nothing comes before the ampersand except a space and nothing comes between the ampersand and the slash except appropriate parameters corresponding to that particular namelist group. However, it is important to note that any comments in an input file that is opened by CEdit and saved will be lost.

The parameters in the input file can be integers, reals, character strings, or logical parameters. A logical parameter is either `.TRUE.` or `.FALSE.` – the periods are a Fortran convention. Character strings that are listed in this User’s Guide must be copied exactly as written – the code is case sensitive and underscores *do* matter. The maximum length of most character input parameters is 60.

Most of the input parameters are simply real or integer scalars, like `PRINT = 50.`, but sometimes the inputs can be arrays.

Note that character strings can be enclosed either by single or double quotation marks, however CEdit only recognizes the single quotation mark. Be careful not to create the input file by pasting text from something other than a simple text editor, in which case the punctuation marks may not transfer properly into the text file. Some text file encodings may not work on all systems. If file reading errors occur and no typographical errors can be found in the input file, try saving the input file using a different encoding. For example, the text file editor Notepad works fine on a Windows PC, but a file edited in Notepad may not work on Linux or Mac OS X because of the difference in line endings between Windows and Unix/Linux operating systems. The editor Wordpad typically works better, but try a simple case first.

## B.3 Input File Structure

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

```
&TAIL /
```

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

The general structure of an input file is shown below, with many lines of the original input file<sup>3</sup>

---

<sup>3</sup>The actual input file, `Users_Guide_Example.in`, is part of the CFAST software distribution

removed for clarity.

```

&HEAD VERSION = 7600, TITLE = 'CFAST Simulation' /
&MHDR NUMBER_OF_CASES = 3 WRITE_RANDOM_SEEDS = .TRUE. /

!! Scenario Configuration
&TIME SIMULATION = 3600 PRINT = 60 SMOKEVIEW = 15 SPREADSHEET = 15 /
&INIT PRESSURE = 101325 RELATIVE_HUMIDITY = 50 INTERIOR_TEMPERATURE = 20 EXTERIOR_TEMPERATURE = 20 /

!! Material Properties
&MATERIAL ID = 'CONCRETE' MATERIAL = 'Concrete Normal Weight (6 in)',
  CONDUCTIVITY = 1.75 DENSITY = 2200 SPECIFIC_HEAT = 1, THICKNESS = 0.15 EMISSIVITY = 0.94 /

!! Compartments
&COMP_ID = 'Comp 1'
  DEPTH = 3 HEIGHT = 3 WIDTH = 3
  CEILING_MATL_ID = 'CONCRETE' CEILING_THICKNESS = 0.15 WALL_MATL_ID = 'CONCRETE' WALL_THICKNESS = 0.15
  ORIGIN = 0, 0, 0 GRID = 50, 50, 50 /

!! Wall Vents
&MRND_ID = 'Vent Width Generator' DISTRIBUTION_TYPE = 'UNIFORM' VALUE_TYPE = 'REAL'
  MINIMUM = 0.25 MAXIMUM = 2.0/
&MFLD_ID = 'Wall Vent Width' FIELD_TYPE = 'VALUE' RAND_ID = 'Vent Width Generator' FIELD = 'Wall Vent' 'WIDTH' /
&MRND_ID = 'Vent Height Generator' DISTRIBUTION_TYPE = 'UNIFORM' VALUE_TYPE = 'REAL'
  MINIMUM = 1.5 MAXIMUM = 2.5/
&MFLD_ID = 'Wall Vent Height' FIELD_TYPE = 'VALUE' RAND_ID = 'Vent Height Generator' FIELD = 'Wall Vent' 'TOP' /

&VENT TYPE = 'WALL' ID = 'Wall Vent' COMP_IDS = 'Comp 1' 'OUTSIDE', BOTTOM = 0 TOP = 2, WIDTH = 1
  FACE = 'FRONT' OFFSET = 1 /

!! Fires
&MRND_ID = 'Peak HRR', DISTRIBUTION_TYPE = 'UNIFORM' VALUE_TYPE = 'REAL' MINIMUM = 500000 MAXIMUM = 3000000 /
&MRND_ID = 'End of Fire HRR' DISTRIBUTION_TYPE = 'CONSTANT' VALUE_TYPE = 'REAL' REAL_CONSTANT_VALUE = 0 /
&MRND_ID = 'Peak HRR Time Interval' DISTRIBUTION_TYPE = 'CONSTANT' VALUE_TYPE = 'REAL' REAL_CONSTANT_VALUE = 900 /
&MRND_ID = 'Fire Time Interval' DISTRIBUTION_TYPE = 'CONSTANT' VALUE_TYPE = 'REAL' REAL_CONSTANT_VALUE = 10 /

&MFIR_ID = 'Fire_generator' FIRE_ID = 'Fire' FIRE_TIME_GENERATORS = 'Fire Time Interval'
  'Peak HRR Time Interval' 'Fire Time Interval' FIRE_HRR_GENERATORS = 'Peak HRR' 'Peak HRR' 'End of Fire HRR' /

&FIRE_ID = 'Fire' COMP_ID = 'Comp 1', FIRE_ID = 'Constant Fire' LOCATION = 1.5, 1.5 /
&CHEM_ID = 'Constant Fire' CARBON = 1 CHLORINE = 0 HYDROGEN = 4 NITROGEN = 0 OXYGEN = 0
  HEAT_OF_COMBUSTION = 50000 RADIATIVE_FRACTION = 0.35 /
&TABL_ID = 'Constant Fire' LABELS = 'TIME', 'HRR', 'HEIGHT', 'AREA', 'CO_YIELD', 'SOOT_YIELD', 'HCN_YIELD', 'HCL_YIELD',
  'TRACE_YIELD' /
&TABL_ID = 'Constant Fire', DATA = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 /
&TABL_ID = 'Constant Fire', DATA = 10, 100, 0, 0.113798159261744, 0, 0, 0, 0, 0, 0 /
&TABL_ID = 'Constant Fire', DATA = 990, 100, 0, 0.113798159261744, 0, 0, 0, 0, 0, 0 /
&TABL_ID = 'Constant Fire', DATA = 1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 /

!! Special Outputs
&DUMP_ID = 'Time to Upper Layer 600 C' FILE = 'COMPARTMENTS' TYPE = 'trigger_greater'
  FIRST_FIELD = 'Time' 'Simulation Time' SECOND_FIELD = 'Comp 1' 'Upper Layer Temperature' CRITERION = 600 /
&DUMP_ID = 'Actual HRR at Upper Layer 600 C' FILE = 'COMPARTMENTS' TYPE = 'trigger_greater'
  FIRST_FIELD = 'Fire' 'HRR Actual' SECOND_FIELD = 'Comp 1' 'Upper Layer Temperature' CRITERION = 600 /
&DUMP_ID = 'Time to Layer Height 1.5 m' FILE = 'COMPARTMENTS' TYPE = 'trigger_lesser'
  FIRST_FIELD = 'Time' 'Simulation Time' SECOND_FIELD = 'Comp 1' 'Layer Height' CRITERION = 1.5 /
&DUMP_ID = 'Maximum Upper Layer Temp' FILE = 'COMPARTMENTS' TYPE = 'maximum'
  FIRST_FIELD = 'Comp 1' 'Upper Layer Temperature' /
&DUMP_ID = 'Minimum Layer Height' FILE = 'COMPARTMENTS' TYPE = 'minimum'
  FIRST_FIELD = 'Comp 1' 'Layer Height' /
&DUMP_ID = 'Maximum Actual HRR' FILE = 'COMPARTMENTS' TYPE = 'maximum'
  FIRST_FIELD = 'Fire' 'HRR Actual' /

&MSTT_ID = 'Width of Vent' FYI = 'I am a firm believer in checking the inputs to make sure that you are getting what you expect'
  ANALYSIS_TYPE = 'HISTOGRAM' OUTPUT_FILENAME = 'simple_width.jpg' COLUMN_TITLE = 'Wall Vent_WIDTH' /
&MSTT_ID = 'Top of Vent' FYI = 'I am a firm believer in checking the inputs to make sure that you are getting what you expect'
  ANALYSIS_TYPE = 'HISTOGRAM' OUTPUT_FILENAME = 'simple_top.jpg' COLUMN_TITLE = 'Wall Vent_TOP' /
&MSTT_ID = 'Peak HRR' FYI = 'I am a firm believer in checking the inputs to make sure that you are getting what you expect'
  ANALYSIS_TYPE = 'HISTOGRAM' OUTPUT_FILENAME = 'simple_peak_hrr.jpg' COLUMN_TITLE = 'Fire_HRR_PT_2' /
&MSTT_ID = 'Time to FO' FYI = 'I am a firm believer in checking the inputs to make sure that you are getting what you expect'
  ANALYSIS_TYPE = 'HISTOGRAM' OUTPUT_FILENAME = 'Simple_Time_to_FO.jpg' COLUMN_TITLE = 'Time to Upper Layer 600 C' /
&MSTT_ID = 'Max Upper Temp' FYI = 'I am a firm believer in checking the inputs to make sure that you are getting what you expect'
  ANALYSIS_TYPE = 'HISTOGRAM' OUTPUT_FILENAME = 'Simple_MaxUpperTemp.jpg' COLUMN_TITLE = 'Maximum Upper Layer Temp' /
&MSTT_ID = 'Convergence of Layer Height Reaching 1.5' OUTPUT_FILENAME = 'Simple_time_to_1p5.jpg' ANALYSIS_TYPE =
  'CONVERGENCE_OF_MEAN'
  COLUMN_TITLE = 'Time to Layer Height 1.5 m' /
&MSTT_ID = 'Convergence of Max Temp' OUTPUT_FILENAME = 'Simple_max_temp.jpg' ANALYSIS_TYPE = 'CONVERGENCE_OF_MEAN'
  COLUMN_TITLE = 'Maximum Upper Layer Temp' /
&MSTT_ID = 'Correlation Tree on Temp' OUTPUT_FILENAME = 'Simple_tree_temp.jpg' ANALYSIS_TYPE = 'CORRELATION TREES'
  COLUMN_TITLE = 'Maximum Upper Layer Temp' /

&TAIL /

```

It is recommended that when looking at a new scenario, first select a pre-written input file that resembles the case, make the necessary changes, then run the case 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. As you learn how to write input files, you will continually run and re-run your case as you add in complexity.

Table B.1 provides a quick reference to all the namelist parameters used by CData and where you can find the reference to where it is introduced in the document and the table containing all of the keywords for each group.

Table B.1: CFAST Input File Keywords

Group Name	Namelist Group Description	Reference Section	Parameter Table
MFIR	Monte Carlo fire definitions	2.4	B.6
MFLD	Monte Carlo field definitions	2.3	B.4
MHDR	Monte Carlo header	2.1	B.2
MRND	Monte Carlo random distributions	2.2	B.3
MSTT	Monte Carlo statistical outputs	2.5	B.7

Examples of each of the Monte Carlo inputs are included in the sections that follow. A complete list of all CFAST inputs are included in the CFAST User’s Guide. All examples are taken from the sample input file `Simple.in` listed above. Following are some general rules about the CFAST input file:

- The `&HEAD` input identifies the version of CFAST for which the input file was created and is typically the first line in the input file. Use `&TAIL` as the last line at the end of the input file. This completes the file from `&HEAD` to `&TAIL`. CFAST does not even look for this last line. It just forces the “end of file” character past relevant input.
- Many of the listed keywords are mutually exclusive. Repeated entry of some keywords can cause the program to either fail or run in an unpredictable manner.
- Use of some keywords triggers the code to operate in a certain mode/condition. For example, specifying `ADIABATIC` to be `.TRUE.` triggers the code to treat all compartment surfaces to be perfectly insulated.
- Multiple inputs are required whenever the keyword is in plural form — keywords ending with an `s`. For example, the keyword parameter, `TEMPERATURES`, within the namelist group, `INIT`, requires two temperature values (in this case, one for exterior ambient temperature and one for interior ambient temperature). In the case of missing inputs, an error message will be generated to assist users in troubleshooting any errors.
- Default values to inputs are assigned to some of the keywords to facilitate the set up of an input file. Users should review the applicability of any default values for their simulation.

## B.4 Monte Carlo Header, Namelist Group MHDR

Table B.2: For more information see Section 2.1.

MHDR (Monte Carlo Header Parameters)				
Parameter	Type	Reference	Units	Default Value
NUMBER_OF_CASES	Integer	Section 2.1		
OUTPUT_DIRECTORY	Character	Section 2.1		Local directory
PARAMETER_FILENAME	Character	Section 2.1		<project>_parameters
SEEDS	Integer Pair	Section 2.1		
WORK_DIRECTORY	Character	Section 2.1		Local directory
WRITE_SEEDS	Logical	Section 2.1		.TRUE.

Examples:

```
&MHDR NUMBER_OF_CASES = 3 WRITE_RANDOM_SEEDS = .TRUE. /
```

## B.5 Monte Carlo Random Input Generators, Namelist Group

### MRND

Table B.3: For more information see Section [2.2](#).

MRND (Monte Carlo Random Generators)				
Parameter	Type	Reference	Units	Default Value
ID	Character	Section <a href="#">2.2</a>		
FYI	Character	Section <a href="#">2.2</a>		
TYPE	Selection List	Section <a href="#">2.2</a>		
RANDOM_SEEDS	Integer Pair	Section <a href="#">2.2</a>		
VALUES	Character array	Section <a href="#">2.2</a>		
PROBABILITIES	Real array	Section <a href="#">2.2</a>		
MINIMUM	Real	Section <a href="#">2.2</a>		
MAXIMUM	Real	Section <a href="#">2.2</a>		
PEAK	Real	Section <a href="#">2.2</a>		
MEAN	Real	Section <a href="#">2.2</a>		
STDEV	Real	Section <a href="#">2.2</a>		
ALPHA	Real	Section <a href="#">2.2</a>		
BETA	Real	Section <a href="#">2.2</a>		
MINIMUM_FIELD	Character Pair	Section <a href="#">2.2</a>		
MAXIMUM_FIELD	Character Pair	Section <a href="#">2.2</a>		

Examples:

```
&MRND ID = 'Vent Width Generator' DISTRIBUTION_TYPE = 'UNIFORM' VALUE_TYPE =
  'REAL'
  MINIMUM = 0.25 MAXIMUM = 2.0/

&MRND ID = 'Fire Time Interval' DISTRIBUTION_TYPE = 'CONSTANT' VALUE_TYPE =
  'REAL'
  REAL_CONSTANT_VALUE = 10 /
```

## B.6 Monte Carlo Input Field Generators, Namelist Group MF LD

Table B.4: For more information see Section 2.3.

MF LD (Monte Carlo Input Field Generators)				
Parameter	Type	Reference	Units	Default Value
ID	Character	Section 2.3		
FYI	Character	Section 2.3		
TYPE	Selection List	Section 2.3		
FIELD	Character Pair	Section 2.3		
FIELD_LABELS	Character	Section 2.3		
RAND_ID	Character	Section 2.3		
ADD_TO_PARAMETERS	Logical	Section 2.3		
PARAMETER_COLUMN_LABEL	Character	Section 2.3		
VALUES	Character array	Section 2.3		
BASE_SCALING_VALUE	Real	Section 2.3		

Examples:

```
&MFLD ID = 'Wall Vent Width' FIELD_TYPE = 'VALUE' RAND_ID = 'Vent Width
Generator'
FIELD = 'Wall Vent' 'WIDTH' /
```

Table B.5: &MFLD Inputs That Can be Varied Based on User-Defined Distributions

Category	Input	&MFLD Input	Units
Ambient Conditions	Interior Temperature	INTERIOR_TEMPERATURE	°C
	Exterior Temperature	EXTERIOR_TEMPERATURE	°C
	Pressure	PRESSURE	Pa.
	Relative Humidity	REALATIVE_HUMIDITY	%
Thermal Properties	Thermal Conductivity	CONDUCTIVITY	kW/(m °C)
	Specific Heat	SPECIFIC_HEAT	kJ/(kg °C)
	Density	DENSITY	kg/m <sup>3</sup>
	Default Thickness	THICKNESS	m
	Emissivity	EMISSIVITY	
Compartments	Width	WIDTH	m
	Depth	DEPTH	m
	Height	HEIGHT	m
	Ceiling Material	CEILING_MATERIAL_ID_n	Selection List
	Wall Material	WALL_MATERIAL_ID_n	Selection List
	Floor Material	FLOOR_MATERIAL_ID_n	Selection List
	Ceiling Material Thickness	CEILING_THICKNESS	m
	Wall Material Thickness	WALL_THICKNESS	m
	Floor Material Thickness	FLOOR_THICKNESS	m
	Wall Leak Area Ratio	WALL_LEAK_AREA_RATIO	m <sup>2</sup> /m <sup>2</sup>
Wall Vents	Floor Leak Area Ratio	FLOOR_LEAK_AREA_RATIO	m <sup>2</sup> /m <sup>2</sup>
	Sill	TOP	m
	Soffit	BOTTOM	m
	Width	WIDTH	m
	Open/Close Fraction	F_nm	0-1
	Open/Close Time	T_nm	s
	Setpoint	SETPOINT	°C, or kW/m <sup>2</sup>
Pre-Activation Fraction		PRE_ACTIVATION_FRACTION	0-1
Post-Activation Fraction		POST_ACTIVATION_FRACTION	0-1

Table B.5: Continued

Category	Input	& MFLD Input	Units
Ceiling / Floor Vents	Cross-Sectional Area Open/Close Fraction Open/Close Time	AREA F_mm T_mm	m <sup>2</sup> 0-1 s
Setpoint	SETPOINT		°C, or kW/m <sup>2</sup>
Pre-Activation Fraction	PRE_ACTIVATION_FRACTION		0-1
Post-Activation Fraction	POST_ACTIVATION_FRACTION		0-1
Mechanical Vents	From Compartment Area From Compartment Height To Compartment Area To Compartment Height Flow Rate Begin Dropoff End Dropoff Open/Close Fraction Open/Close Time Setpoint Pre-Activation Fraction Post-Activation Fraction Filter Efficiency Begin Filtering Time	FROM_AREA FROM_HEIGHT TO_AREA TO_HEIGHT FLOW FIRST_CUTOFF SECOND_CUTOFF F_mm T_mm SETPOINT PRE_ACTIVATION_FRACTION POST_ACTIVATION_FRACTION FILTER EFFICIENCY FILTER_TIME	m <sup>2</sup> m m <sup>2</sup> m m <sup>3</sup> /s Pa Pa 0-1 s °C, or kW/m <sup>2</sup> 0-1 0-1 % s
Fire Location	Compartment Width Position Depth Position	COMPARTMENT X_POSITION Y_POSITION	Selection List m m
Fuel Composition	Carbon Molecules Hydrogen Molecules Oxygen Molecules Nitrogen Molecules Chlorine Molecules Heat of Combustion	CARBON HYDROGEN OXYGEN NITROGEN CHLORINE HEAT_OF_COMBUSTION	> 0 ≤ 0 ≤ 0 ≤ 0 ≤ 0 kJ/kg

Table B.5: Continued

Category	Input	&MFLD Input	Units
Ignition Criteria	Radiative Fraction Ignition Criterion Setpoint	RADIATIVE_FRACTION IGNITION_CRITERION SETPOINT	0 - 1 Selection List s, °C, or kW/m <sup>2</sup>
Targets	Width Target Position Depth Target Position Height Target Position Width Normal Vector Depth Normal Vector Height Normal Vector Target Points To Thickness Internal Temperature Location	X_LOCATION Y_LOCATION Z_LOCATION X_DIRECTION Y_DIRECTION Z_DIRECTION SURFACE_ORIENTATION THICKNESS INTERNAL_TEMPERATURE_DEPTH	m m m 0-1 0-1 0-1 Selection List m m
Detection / Suppression	Width Position Depth Position Height Position Activation Temperature Activation Obscuration RTI Spray Density	X_LOCATION Y_LOCATION Z_LOCATION SETPOINT_TEMPERATURE SETPOINT_OSCURATION RTI SPRAY_DENSITY	m m m °C %/m (m s) <sup>1/2</sup> m/s

## B.7 Monte Carlo Input Fire Generators, Namelist Group `MFIR`

Table B.6: For more information see Section 2.4.

Parameter	Type	Reference	Units	Default Value
ID	Character	Section 2.4		
FYI	Character	Section 2.4		
FIRE_ID	character	Section 2.4		
MODIFY_FIRE_AREA_TO_MATCH_HRR	Logical	Section 2.4		
FIRE_COMPARTMENT_RANDOM_GENERATOR_ID	Character	Section 2.4		
FIRE_COMPARTMENT_IDS	Character Array	Section 2.4		
ADD_FIRE_COMPARTMENT_TO_PARAMETERS	Logical	Section 2.4		
FIRE_COMPARTMENT_PARAMETER_COLUMN_LABEL	Character	Section 2.4		
INCIPIENT_TYPE_RANDOM_GENERATOR_ID	Character	Section 2.4		
INCIPIENT_FIRE_TYPES	Character array	Section 2.4		
INCIPIENT_GROWTH_TYPE	Character	Section 2.4		
FLAMING_INCIPIENT_DELAY_RANDOM_GENERATOR_ID	Character	Section 2.4		
FLAMING_PEAK_INCIPIENT_RANDOM_GENERATOR_ID	Character	Section 2.4		
SMOLDERRING_INCIPIENT_DELAY_RANDOM_GENERATOR_ID	Character	Section 2.4		
SMOLDERRING_PEAK_INCIPENT_RANDOM_GENERATOR_ID	Character	Section 2.4		
ADD_INCIPENT_TYPE_TO_PARAMETERS	Logical	Section 2.4		
ADD_INCIPENT_TIME_TO_PARAMETERS	Logical	Section 2.4		
ADD_INCIPENT_PEAK_TO_PARAMETERS	Logical	Section 2.4		
BASE_FIRE_ID	Character	Section 2.4		
SCALING_FIRE_HRR_RANDOM_GENERATOR_ID	Character	Section 2.4		
SCALING_FIRE_TIME_RANDOM_GENERATOR_ID	Character	Section 2.4		
ADD_HRR_SCALE_TO_PARAMETERS	Logical	Section 2.4		
HRR_SCALE_COLUMN_LABEL	Character	Section 2.4		
ADD_TIME_SCALE_TO_PARAMETERS	Logical	Section 2.4		
TIME_SCALE_COLUMN_LABEL	Character	Section 2.4		

Table B.6: Continued

MFIR (Monte Carlo Fire Generators)			
FIRE_TIME_GENERATORS	Character array	Section 2.4	
FIRE_HRR_GENERATORS	Character array	Section 2.4	
NUMBER_OF_GROWTH_POINTS	Integer	Section 2.4	
NUMBER_OF_DECAY_POINTS	Integer	Section 2.4	
GROWTH_EXPONENT	Real	Section 2.4	
DECAY_EXPONENT	Real	Section 2.4	
ADD_HRR_TO_PARAMETERS	Logical	Section 2.4	
HRR_COLUMN_LABEL	Character	Section 2.4	
ADD_TIME_TO_PARAMETERS	Logical	Section 2.4	
TIME_COLUMN_LABEL	Character	Section 2.4	

**Examples:**

```
&MFLD ID = 'Wall Vent Width' FIELD_TYPE = 'VALUE' RAND_ID = 'Vent Width  
Generator'  
FIELD = 'Wall Vent' 'WIDTH' /
```

## B.8 Monte Carlo Statistical Plots, Namelist Group MSTT

Table B.7: For more information see Section [2.5](#).

MSTT (Monte Carlo Statistics)				
Parameter	Type	Reference	Units	Default Value
ID	Character	Section <a href="#">2.5</a>		
FYI	Character	Section <a href="#">2.5</a>		
ANALYSIS_TYPE	Selection List	Section <a href="#">2.5</a>		
INPUT_FILENAME	Character Pair	Section <a href="#">2.5</a>		<project>_accumulate.csv
OUTPUT_FILENAME	Character	Section <a href="#">2.5</a>		
ERROR_FILENAME	Character	Section <a href="#">2.5</a>		<project>.err
LOG_FILENAME	Logical	Section <a href="#">2.5</a>		<project>.log
COLUMN_LABEL	Character	Section <a href="#">2.5</a>		

Examples:

```
&MSTT ID = 'test 1' ANALYSIS_TYPE = 'HISTOGRAM'  
    INPUT_FILENAME = 'Andy_accumlate.csv'  
    OUTPUT_FILENAME = 'Andy_bedroom_1_window.jpg'  
    COLUMN_TITLE = 'Bedroom 1 Front Window_SETPOINT' /
```



# Appendix C

## Running CData from a Command Prompt

CData is run from a Windows command prompt from the folder where the base input file is located.

```
[drive1:\][folder1\]cdata project.in
```

The project name will have extensions appended as needed (see below). For example, to run a test case when the CFAST executable is located in c:\firemodels\cfast7 and the input data file is located in c:\data, the following command could be used:

```
c:\firemodels\cfast7\cfast c:\data\testfire0    <<< note that the extension is optional.
```

One or more command line options can follow the name of the file to be run as follows:

- -A - *Accumulate* - combine individual summary files from all simulations in a data set into a single comma-separated spreadsheet file.
- -P - *Preprocessor* - Create a set of individual CFAST inputs files for Monte-Carlo analysis as specified in the base Monte-Carlo input file
- -S - *Statistics* - Create summary statistical outputs as specified in the base Monte-Carlo input file.



# Appendix D

## R Scripts for Example Cases

The scripts listed below are in R<sup>1</sup> and are in the form of R Markdown<sup>2</sup> files.

### D.1 R Script for Interconnected Alarms

```
title: "Interconnected Alarms"
author: ""
date: "`r format(Sys.Date(), '%e %B %Y')`"
output:
  pdf_document:
    keep_tex: true
params:
  save_file: "example3"
  seed: 737826061
  detail: FALSE
  max_t: 1000
---

```{r setup, include=FALSE}
library(rpart)
# The package above is part of base R. The packages below are not.
# The script can easily be rewritten to work without them.
# I use them here because they work well for the purpose.
library(magrittr)
library(foreach)
library(ggplot2)
library(rpart.plot)

set.seed(as.integer(params$seed))
knitr::opts_chunk$set(echo=params$detail, results="hold", warning=FALSE, message=FALSE)

# These lines allow me to run the script multiple times without having to
# repeat some of the more time-intensive portions.
keep <- union(ls(), c("ex3", "reserved", "vars", "cvg"))
do.ex3   <- ! exists("ex3")
do.find  <- do.ex3 | ! exists("vars")
do.cvg   <- do.ex3 | ! exists("cvg")
```

---

<sup>1</sup>R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.

<sup>2</sup>Yihui Xie and J.J. Allaire and Garrett Grolemund (2018). R Markdown: The Definitive Guide. Chapman and Hall/CRC. ISBN 9781138359338. URL <https://bookdown.org/yihui/rmarkdown>

```

```
## Data

The information on doors is not really in a usable form, so for this analysis
the door information is recoded into a door state for each door. The state
can be one of "closed" "one-fourth", "one-half", "three-fourth", and "open."
The recoding for doors is what is used throughout this analysis.

```{r data, eval=do.ex3}
ex3 <- read.csv("Detectors_accumulate3.csv",
# Note that the column names as supplied are not R-compliant names.
# It is more complicated to work with the names as supplied, but
# it makes reporting easier. So I use `check.names=FALSE` to import
# the names as supplied, and deal with the complications.
#
# check.names=FALSE,
# stringsAsFactors = FALSE)
#####
# There are a few columns that are empty, so they are removed.
tmp <- sapply(ex3, function(x) all(is.na(x)))
ex3 <- ex3[,!tmp]

# In addition, there may be rows that are empty as well, and they are removed.
tmp <- apply(ex3, 1, function(x) all(is.na(x) | trimws(x) == ""))
ex3 <- ex3[!tmp,]

# The "File.Names" Column does me no good and can cause problems with some of
# the analyses. For this work I just delete it from the data.
# The column is saved in the 'reserved' data frame.
tmp <- grep("File[.]Names", names(ex3))
reserved <- NULL
if(length(tmp) > 0){
  if(length(tmp) == 1){
    reserved <- data.frame(ex3[,tmp])
    names(reserved) <- names(ex3)[tmp]
  } else {
    reserved <- cbind(reserved, ex3[,tmp])
  }
  ex3 <- ex3[,-tmp]
}

# Take any text columns and convert them to factors
tmp <- sapply(ex3, function(x) class(x) == "character")
if(length(tmp[tmp]) > 0){
  for(i in names(ex3[tmp])) ex3[[i]] <- factor(trimws(ex3[[i]]))
}

#####
## This section is unique to this data set.
# Each of the doors has two variables, but they represent only five
# states for each door. The problem is that those two variables
# are effectively perfectly correlated, which causes problems
# for the analysis.
# The simplest approach--and probably the best--is to convert those
# values to a factor. This does that.
# The approach is to identify Door columns (which have either
# "WIDTH" or "TOP" in the name).
x <- grep("WIDTH", names(ex3), value=TRUE) %>% sub("WIDTH", "[[:LETTERS:]*]", .)
doors <- foreach(i=x, .combine="cbind") %do% {
  u <- ex3[,grep(i, names(ex3))]
  width <- u[[grep("WIDTH", names(u))]]
  top   <- u[[grep("TOP",  names(u))]]
  z <- c(3, 5, 7) * max(width) / 8
  y <- rep(NA, nrow(u)) %>% factor(levels=c("closed", "one-fourth", "one-half", "three-fourth",
  "open"))
  y[width < z[1]] <- "one-fourth"
  y[width < z[2] & width > z[1]] <- "one-half"
}

```

```

y[width < z[3] & width > z[2]] <- "three-fourth"
y[width > z[3] & top == max(top)] <- "open"
y[width > z[3] & top == min(top)] <- "closed"
data.frame(y)
}
names(doors) <- sub("[_]*[[[]:LETTERS:][*]]","", x)
x <- names(doors) %>% grep("DR|DOOR", ., ignore.case=TRUE, invert=TRUE)
names(doors)[x] <- paste0(names(doors)[x], "_DOOR")

doors2 <- ex3[,grep("WIDTH|TOP", names(ex3))]
ex3 <- cbind(ex3[,-grep("WIDTH|TOP", names(ex3))], doors)
reserved <- cbind(reserved, doors, doors2)
rm(doors, doors2, u, i, x, y, z, width, top)
```

```{r data_setup, eval=do.find}
# Note that independent variables are not necessarily pre-stated.
# So I pick out the independent variables as those with no NAs.
# NAs result from run failures, which happen about 10% of the time.
# Any column with an NA is a dependent variable.
#
# I also identify any columns that are obviously singular. Any column
# with only one value is separated out from the others.

a <- sapply(ex3, function(x) any(is.na(x)))
b <- sapply(ex3, function(x) length(unique(x)) < 2)
vars <- list(IV=(1:length(a))[!a & !b],
            DV=(1:length(a))[ a & !b],
            NO=(1:length(a))[ b])

# Generate a random "column" name. If it is in the table, try again until you
# find a name that is not in the table.
# This seems more complicated than necessary, but it GUARANTEES that my index
# name is not already in the table.
idx <- names(a)[1]
while(idx %in% names(ex3)){
  idx <- c(sample(letters, 1), sample(c(letters, 0:9), 9, replace=TRUE)) %>% paste0(collapse="")
}
```

## Number of Cases

```

As a first step, we do a preliminary check to see if we have enough cases. To be meaningful we have to look at one of the output columns; here we look at the Bedroom 1 Alarm data. Looking at the data it becomes very quickly apparent that the results for Flaming type fires and Smoldering type fires are very different. Also, nearly half the cases have no activation within the simulation time. So the evaluation of convergence is evaluated separately for flaming type fires and smoldering type fires. Then the time to activation for cases where activation occurs is evaluated separately from the percent of cases where activation occurs. If any of these do not converge, then we do not have enough cases.

```

```{r converg}
# This does the convergence tests and graphs.
# The algorithm used here is the same as that used as part of the CDATA
# package. The only difference is that the output is printed using
# ggplot rather than R's native graphics engine.
#
# This section builds the tables containing the means versus number of runs.
# It creates a column for each of the dependent variables so a plot can be
# generated for any of them as desired.
#
converg <- foreach(i=levels(ex3$Random_ignition_type), .combine="rbind") %do% {
  foreach(j=c("times", "alerts"), .combine="rbind") %do% {
    t2 <- foreach(k=vars$DV, .combine="data.frame") %do% {
      t3 <- ex3[[k]][ex3$Random_ignition_type == i]
      if(j == "times"){
        
```

```

        t3[t3 < 0] <- NA
    } else {
        t3 <- ifelse(t3 > 0, 1, 0)
    }
    t3[! is.na(t3)] <- cumsum(t3[! is.na(t3)]) / 1:length(t3[! is.na(t3)])
    t3
}
names(t2) <- names(ex3)[vars$DV]
cbind(g2=j, n=1:nrow(t2), t2)
} %>% cbind(gl=i, .)
}

# This section builds the tables containing the standard deviation versus
# number of runs. Again, this is the same algorithm as the one used in
# CDATA, just using ggplot instead of the native R graphics engine.
#
# Also as above, it creates a column for each of the dependent
# variables so a plot can be generated for any of them as desired.
#
converg <- list(mean=converg,
                  sd=tmp <- foreach(i=levels(ex3$Random_ignition_type), .combine="rbind") %do% {
t0 <- ex3[ex3$Random_ignition_type == i, vars$DV]
t4a <- data.frame(n=c(nrow(t0),
                      floor(exp(seq(from=log(nrow(t0)), to=log(10), length.out=10)[2:9])),
                      10),
len=NA) %>% cbind(foreach(j=names(t0), .combine="cbind") %do% rep(NA, 10))
t4a$len <- nrow(t0) / t4a$n
names(t4a)[-c(1:2)] <- names(t0)
t4b <- t4a
t4a[1,3:ncol(t4a)] <- sapply(t0, sd, na.rm=TRUE)
t4b[1,3:ncol(t4b)] <- sapply(t0, function(x) sd(ifelse(x > 0, 1, 0), na.rm=TRUE))
for(j in 2:10){
    for(k in names(t0)){
        t4a[[k]][j] <- sd(sapply(split(t0[[k]],
                                      cut(runif(nrow(t0)), t4a$n[j]),
                                      drop=TRUE),
                                      mean, na.rm=TRUE), na.rm=TRUE)
        t4b[[k]][j] <- sd(sapply(split(ifelse(t0[[k]] > 0, 1, 0),
                                      cut(runif(nrow(t0)), t4b$n[j]),
                                      drop=TRUE),
                                      mean, na.rm=TRUE), na.rm=TRUE)
    }
}
rbind(cbind(g2="times", t4a),
      cbind(g2="alert", t4b)) %>% cbind(gl=i, .)
})

# These are example plots.
print(converg$mean %>% ggplot() + aes(x=n, y='Bedroom 3 Alarm') +
  geom_line() + geom_point() + theme_bw() +
  facet_wrap(~ g2 + gl, scales="free_y") + expand_limits(y=0))

print(converg$sd %>% ggplot() + aes(x=len, y='Bedroom 3 Alarm') +
  geom_line() + geom_point() + theme_bw() +
  facet_wrap(~ g2 + gl, scales="free_y") + expand_limits(y=0))

```

```

The preliminary results, charted above, suggest that we have enough cases. The averages all appear to have converged, and the standard errors are small.

## ## Analysis

The question explored here is the benefit gained by connecting alarms. The effect of interconnecting alarms is to reduce the amount of time before people

in the house are notified of a fire even though they may be far away from the location of the fire. In this example, there are fires in the dining room, kitchen, or living room, and we will assume that people are in the bedrooms. Further assume the people in the bedrooms will not hear an alarm unless it is the one in the bedroom. So the effect of interconnecting alarms is to notify people to a fire when the alarm in one of the front rooms goes off rather than waiting until one of the bedroom alarms sounds.

Assume for this example that there is one alarm in the Living Room and one alarm in Bedroom 1 (which will proxy for all the bedroom alarms). This example, then, examines the difference in activation times between the Living Room Alarm and the Bedroom 1 Alarm.

Exploratory analysis of the data is an important part of any analysis. Here, initial exploration indicated that in a substantial number of cases one or more of the alarms never activated.

```
```{r exploratory}
data.frame(`Living Room` = ifelse(ex3$`Living Room Alarm` > 0, "Yes", "No"),
           `Bedroom 1` = ifelse(ex3$`Bedroom 1 Alarm` > 0, "Yes", "No")) %>%
  table(useNA="ifany") %>% ftable() %>% pander::pander()
```

The table above displays the number of cases where each of the alarms activated. Cases marked NA, are cases where the model itself did not converge. That was a very small number of cases for this example.

For more than half the cases, both alarms activated. For most of the remaining cases the Living Room alarm activated but the Bedroom alarm did not. For less than 10% of the cases neither alarm activated. There were no cases where the bedroom alarm activated but the Living Room alarm did not.

```
```{r cart}
# Again, this is the same algorithm as the one built into CDATA.
# I use the package `rpart.plot` to plot the resulting decision
# tree rather than the R graphics engine, mostly because it produces
# a plot that is easier to read and interpret.
bed1 <- rpart(I(`Bedroom 1 Alarm` > 0) ~ `UL Alarm Living Room_TRIGGER` + `UL Alarm Dining
Room_TRIGGER` +
  `UL Alarm Kitchen_TRIGGER` + `UL Alarm Bedroom 1_TRIGGER` + `UL Alarm Bedroom 2_TRIGGER` +
  `UL Alarm Bedroom 3_TRIGGER` + `UL Alarm Living Room_TRIGGER_SMOLDER` +
  `UL Alarm Dining Room_TRIGGER_SMOLDER` + `UL Alarm Kitchen_TRIGGER_SMOLDER` +
  `UL Alarm Bedroom 1_TRIGGER_SMOLDER` + `UL Alarm Bedroom 2_TRIGGER_SMOLDER` +
  `UL Alarm Bedroom 3_TRIGGER_SMOLDER` + `Random_HRR_PT_2` + `Random_T_HRR_PT_2` +
  I(`Random_T_HRR_PT_15` - `Random_T_HRR_PT_3`) + `DR LR 1` + `DR LR 2` + `DR Kit` + `Kit
LR_DOOR` +
  `LR to BR Hall_DOOR` + `Bedroom 1 Door` + `Bedroom 2 Door` + `Bedroom 3 Door` + `Fire Room` +
  Random_ignition_type,
ex3)
rpart.plot::rpart.plot(bed1)
```

To better understand the cases where the bedroom alarm fails to sound, a classification tree was generated on whether the bedroom alarm sounded (above). It is unusual for a tree to produce results as stark as this one. What becomes clear on looking at the tree is that the state of the intervening doors determines whether the bedroom alarm sounds. If the fire is isolated from the bedroom by closed doors then the alarm will not sound. Otherwise it will.

Interconnected alarms do not monitor continuously. Rather they check periodically to see if some other alarm has sounded. For this example, we will assume they check every 60 seconds. The effect is that there is a random delay, with the delay drawn from a uniform random distribution of between zero and 60 seconds. A Kernel Density estimate of the distribution of time savings is plotted below, with fire type on separate charts. Any negative "time delay" means that the bedroom alarm sounds on its own before

```

it gets notification from the Living Room alarm.

```{r kde}
# In the generated data frame below:
# type is either Flaming or Smoldering
# Diff is the difference in activation times between the bedroom alarm
# and the living room alarm. sig.delay is a random value between 0
# and 60 seconds representing the interconnection delay delay is the
# total difference in activation times taking into account the random
# interconnection delay.
#
# Note that I have use Inf (an infinite value) to represent non-activation
# for any alarm.
delay <- data.frame(type=ex3$Random_ignition_type,
                     diff=ifelse(ex3$`Bedroom 1 Alarm` < 0, Inf, ex3$`Bedroom 1 Alarm`) -
                         ifelse(ex3$`Living Room Alarm` < 0, Inf, ex3$`Living Room Alarm`),
                     sig.delay=60 * runif(length(ex3$Random_ignition_type)),
                     delay=NA)
delay$delay <- with(delay, diff - sig.delay)

# This plots a KDE estimate of the delay, EXCLUDING the cases where one or
# both of the alarms did not activate.
# This is equivalent to the Density option contained in CDatas, just using a
# different graphics engine.
delay %>% ggplot() + aes(x=ifelse(is.infinite(delay), NA, delay)) +
  geom_density() + theme_bw() + facet_wrap(~ type, scales="free")
```

```{r dist, eval=do_cvg}
# Again we are doing the mean v. number of cases data. While the base
# algorithm is the same as that contained in CDatas, this applies the
# approach to quantiles of the `delay` data created in the section above.
# The quantiles are not a part of CDatas.
cvg <- foreach(i=1:nrow(delay), .combine="rbind")%do%
{
  if(i < 100){
    c(i, rep(NA, 12))
  } else {
    tmp <- by(delay[1:i], delay$type[1:i],
              function(x){
                with(subset(x, is.finite(delay)),
                     c(mean(pmax(delay, 0), na.rm=TRUE),
                        quantile(delay, c(0.5, 0.75, 0.9, 0.95, 0.99), na.rm=TRUE)))
              })
    c(i, tmp[[1]], tmp[[2]])
  }
} %>% as.data.frame()
names(cvg) <- c("n", "f.mean", "f.50", "f.25", "f.10", "f.05", "f.01",
               "s.mean", "s.50", "s.25", "s.10", "s.05", "s.01")
```

```

With 10,000 data points we can empirically estimate the average time savings that interconnected alarms would provide, as well as median time savings and various quantiles. We are interested in the quantiles because adverse outcomes, like death or injury, are tail events. So we also look at the tails of the distribution. Here the "% Quantile" means that 1% of alarms saved more time than this.

```

```{r table4}
tmp <- cvg[nrow(cvg), 2:13] %>% matrix(ncol=2) %>% as.data.frame()
names(tmp) <- c("Flaming", "Smoldering")
tmp <- cbind(result=c("Mean", "Median", paste0(c(25, 10, 5, 1), "% Quantile")), tmp)
knitr::kable(tmp)
```

```

This looks only at cases where the Bedroom 1 alarm sounded. If this were a serious attempt to identify the effect of interconnected alarms then the percent of non-activations would need to be taken into account. If

```
non-activations were included all the quantiles larger than the median would
fall into the non-activation range.
```

```
## Number of Cases – Revisited
```

The preliminary analysis above suggested that we appeared to have enough cases. However, the values of interest here are complex manipulations of the data. Even though the average of the Bedroom activation time has converged, it is possible that the results of interest to us have not. To finally determine if we have enough cases we look at how the outcomes we analyze change with the number of cases.

```
```{r cvg_plot}
tmp <- cvg[,c(1,2,6, 8, 12)] %>% tidyverse::pivot_longer(2:5, names_to=c("Type", "Quantile"),
  names_sep="[.]")
tmp$Type <- factor(tmp$Type, levels=c("f", "s"), labels=c("Flaming", "Smoldering"))
tmp$Quantile <- factor(tmp$Quantile, levels=c("mean", "05"), labels=c("Mean", "5 % Quantile"))

# Again I use ggplot for plotting instead of the base R graphics engine.
tmp %>% ggplot() + aes(x=n, y=value) + geom_line() + geom_point() +
  facet_wrap(~ Type + Quantile, scales="free_y") + theme_bw()
```

```

While the preliminary look settled down within a couple of thousand cases, these results take longer. In particular the extreme quantiles take somewhat longer to settle down. In particular, the 5 % quantile value doesn't really settle down until we have 7500 cases or so. However, looking at the results it looks like our results have settled down sufficiently by the 10,000 cases we actually use to have reasonable confidence in our results.

```
## Discussion
```

For a study like this it is really important to get the distribution of fire parameters correct. For example, if the proportion of fires starting in the Kitchen were low compared to the real world then all of the time-delay values would be underestimated. If the likelihood of closed doors were overestimated, then there would be far fewer cases of alarm failure for the bedroom alarms, and the benefit of interconnection would be lower.

```
```{r cleanup}
# Does some final cleanup
rm(list=setdiff(ls(), keep))
save.image(file=paste0(params$save_file, ".RData"))
```

```

## D.2 R Script for Sensitivity

```
title: "Sensitivity Analysis"
author: ""
date: "`r format(Sys.Date(), '%e %B %Y')`"
output:
  pdf_document:
    keep_tex: true
params:
  save_file: "example2"
  seed: 244213930
  detail: FALSE
  max_t: 1000
---
```{r setup, include=FALSE}
```

```

# The `survival` library is part of core R.
# The remaining libraries are not.
# The libraries `magrittr` and `foreach` are just easier and more intuitive
# ways of doing some tasks. The library `ggplot2` is a graphics engine
# that is used in lieu of core R's graphics engine. It is possible to rewrite
# this script file to work without those libraries.
library(magrittr)
library(foreach)
library(ggplot2)
library(survival)
# set.seed(as.integer(params$seed))
keep <- union(ls(), c("ex2", "out", "use.vars", "converg"))

knitr::opts_chunk$set(echo=params$detail, results="hold", warning=FALSE, message=FALSE)

do.ex2    <- ! exists("ex2")
do.find   <- do.ex2 | ! exists("use.vars")
do.mdl    <- do.find | ! exists("out")
do.cvg    <- do.find | ! exists("converg")
```
``{r data, eval=do.ex2}
ex2 <- read.csv("Sensitivity_accumulate.csv",
# Note that the column names as supplied are not R-compliant names.
# It is more complicated to work with the names as supplied, but
# it makes reporting easier. So I use `check.names=FALSE` to import
# the names as supplied, and deal with the complications.
# check.names=FALSE,
# stringsAsFactors = FALSE)
# There are a few columns that are empty, so they are removed.
tmp <- sapply(ex2, function(x) all(is.na(x)))
ex2 <- ex2[,!tmp]

# In addition, there are rows that are empty as well, and they are removed.
tmp <- apply(ex2, 1, function(x) all(is.na(x) | trimws(x) == ""))
ex2 <- ex2[!tmp,]

# The "File.Names" Column does me no good and can cause problems with some of
# the analyses. For this work, I just delete it from the data.
#!!!Saving it in case it is needed might be a good idea for future reference.
fn <- grep("File[.]Names", names(ex2))
if(length(fn) > 0){
  ex2 <- ex2[,-fn]
}
```
``{r data_setup, eval=do.find | do.mdl | do.cvg}
# Note that independent variables are not necessarily pre-stated.
# So I pick out the independent variables as those with no NAs.
# NAs result from run failures, which happen about 10% of the time.
# Any column with an NA is a dependent variable.

a <- sapply(ex2, function(x) any(is.na(x)))
vars <- list(IV=(1:length(a))[!a],
            DV=(1:length(a))[ a])

# For sensitivity, no variable is particularly important to the discussion,
# so pick one with high sensitivity to something.

a <- sapply(ex2[,vars$DV], function(x) length(x[! is.na(x) & x > 0]))
n <- max(a)

# Generate a random "column" name. If it is in the table, try again until you
# find a name that is not in the table.
# This seems more complicated than necessary, but it GUARANTEES that my index
# name is not already in the table.
idx <- names(a)[1]
while(idx %in% names(ex2)){

```

```

idx <- c(sample(letters , 1), sample(c(letters , 0:9) , 9, replace=TRUE)) %>% paste0(collapse="")
}

# The first is simply a log-log analysis.
# Since some of these variables are times till an event, where the event may
# not occur within the time of the run, those need to be analyzed using a
# Tobit model. The second sets that up. This is still a log-log analysis, just
# with the tools of Tobit analysis set up.
fmla1 <- paste0("log(`", idx , "`") ~ log(`",
           paste0(names(ex2)[vars$IV], collapse="`") + log(`"),
           "`") )
fmla2 <- paste0("Surv(log(ifelse(`", idx , "` < 0, ", params$max_t, ", `", idx , "`)), `",
           idx , "` > 0, type="right") ~ log(`",
           paste0(names(ex2)[vars$IV], collapse="`") + log(`"), "`") )
```

```{r select, eval=do.find}
# I can use pretty much any of the dependent variables that I want
# for this. So I select the ones with the largest effect to use in the
# actual analysis.
summ <- list(lm=foreach(i=names(a)[a == n], .combine="rbind") %do% {
            do.call("lm", list(formula=sub(idx , i, fmla1) %>% formula(),
                               data=quote(ex2))) %>%
            coef()
        },
        sr=foreach(i=names(a)[a > 10 & a < n], .combine="rbind") %do% {
            do.call("survreg", list(formula=gsub(idx , i, fmla2) %>% formula(),
                                    data=quote(ex2),
                                    dist="gaussian")) %>%
            coef()
        })
row.names(summ$lm) <- names(a)[a == n]
row.names(summ$sr) <- names(a)[a > 10 & a < n]

use.vars <- foreach(i=names(summ), .combine="c") %do% {
    y <- apply(summ[[i]][,-1], 1, function(x) max(x) == max(summ[[i]][,-1]))
    row.names(summ[[i]])[y][1]
}
```

## Approach

```

For this example it is assumed that the output variables of interest are ``r use.vars[1]`` and ``r use.vars[2]``. Notionally, we might say that we are interested in the amount of time they have to escape (represented by the time during which they foyer--through which they must evacuate--remains viable) and the worst temperature they might have to face in the Foyer during that evacuation.

Sensitivity analysis is interested in the sensitivity the output has to each input. This is typically expressed in terms of the percent variation in output produced by a one percent variation in each input. As it turns out, simple linear regression of the \*log\* of the inputs against the \*log\* of the output will give exactly that value.

```

```{r test}
# Percent cases where FED > 1 does not occur.
u <- ex2[[use.vars[2]]]
u <- u[! is.na(u)]
u <- length(u[u < 0]) / length(u)
```

```

The second target variable--`r use.vars[2]`--was selected to illustrate an additional complication in the analysis. This variable expresses the time till the FED reaches the value of 1--assumed to represent the condition where the space is no longer viable. The complication is that in a number of cases nonviability does not occur within the time of the simulation: here some `r formatC(u \* 100, digits=3)`% of cases do not result

in non-viability. That presents a problem because analyzing the data without taking that into account will result in incorrect estimates of the sensitivity.

In this case what is run is a 'Tobit' analysis (Tobin, James (1958). "Estimation of Relationships for Limited Dependent Variables." *Econometrica*. See Bibliography) which properly accounts for the effect of the cases where non-viability did not occur. The 'Tobit' analysis is also performed on a log-log transformation of the data.

```
```{r analyze, eval=do.mdl}
# This section does the analyses
out <- list(lm=do.call("lm", list(formula=sub(idx, use.vars[1], fmla1) %>% formula(),
                                    data=quote(ex2))),
            sr=do.call("survreg", list(formula=gsub(idx, use.vars[2], fmla2) %>% formula(),
   data=quote(ex2),
   dist="gaussian")))
```

```{r converge, eval=do.csv}
# This section does the convergence analysis that is built into CData.
# The code has been modified slightly to fit this analysis, but is otherwise
# identical to what is built into CData.
converg <- foreach(i=1:10000, .combine="rbind") %do% {
  if(i == 1) {
    X <- model.matrix(sub(idx, use.vars[1], fmla1) %>% formula(), ex2)
    y <- model.response(model.frame(sub(idx, use.vars[1], fmla1) %>% formula(), ex2))
    omit <- na.omit(ex2) %>% attr("na.action")
    j <- 99 - length(omit[omit <= 99])
    XX <- t(X[1:j,]) %>% X[1:j,]
    Xy <- t(X[1:j,]) %>% y[1:j]
  } else if(i < 100) {
  } else {
    j <- i - length(omit[omit <= i])
    XX <- XX + X[j,] %>% t(X[j,])
    Xy <- Xy + X[j,] * y[j]
  }
  if(i < 100 | i %in% omit){
    rep(NA, ncol(X))
  } else {
    (solve(XX) %*% Xy) %>% as.numeric()
  }
} %>% as.data.frame()
names(converg) <- colnames(X)
converg <- cbind(n=1:nrow(converg), converg)
```

```{r results}
# Select the variables to include in the table. For the log-log analysis
# I select variables with high, medium and low levels of sensitivity as
# well as the fire scaling factors.
# For the Tobit analysis I select the most sensitive variables as well
# as the fire scaling factors.
vars <- list(lm=coef(out$lm) %>% extract(-1) %>% abs() %>%
              sort(decreasing=TRUE) %>%
              extract(c(1:3, 30:32, 67:70)) %>% names() %>%
              union(grep("scal", names(coef(out$lm)), value=TRUE)),
        sr=coef(out$sr) %>% extract(-1) %>% abs() %>%
              extract(. > .2) %>% sort(decreasing = TRUE) %>% names() %>%
              union(grep("scal", names(coef(out$lm)), value=TRUE)))
```

# Number of Cases
```

An initial run of 1000 cases was evaluated. The chart below shows the coefficient for 'Lab\_depth' for the first model as a function of the number of cases. Examination of the chart shows that the value for the coefficient is still changing substantially at the end of 1000 cases. When the number

of cases were increased to 2500 the results still did not seem very stable. When the cases increased again to 5000 there was a noticeable trend over the last thousand or so cases. When the cases were increased to 10000, the estimated value over the last couple of thousand cases seems to have settled down. So these results are based on 10000 cases.

```
```{r cvg_plot}
tmp <- rbind(cbind(test=1000, converg[1:1000, c("n", "log(Lab_depth)")]),
              cbind(test=2500, converg[1:2500, c("n", "log(Lab_depth)")]),
              cbind(test=5000, converg[1:5000, c("n", "log(Lab_depth)")]),
              cbind(test=10000, converg[1:10000, c("n", "log(Lab_depth)")]))
# This uses the ggplot graphics engine to plot this data instead of the base
# R graphics engine. Otherwise this is the same as what is in CDATA.
print(tmp %>% ggplot() + aes(x=n, y=`log(Lab_depth)`) +
      geom_line() + geom_point() + facet_wrap(~ test, scales="free_x") +
      theme_bw() + coord_cartesian(ylim=c(-1,1)))
```

```

#### # Discussion

First we look at the sensitivity of maximum temperature FED in the Foyer. Selected results are displayed in Table 1. The single most significant factor is the depth of the lab space--where the fire occurred.. A one-percent change in the lab depth will produce a 0.53% decrease in the maximum heat FED for the Foyer. Similarly a one percent change in the height of the Foyer and halls produces a 0.47% decrease in the heat FED for the Foyer. At the other end of the scale changes in the Foyer width or the depth of office 1 produce minimal changes to the heat FED for the Foyer.

```
```{r tb11}
tbl11 <- summary(out$lm)$coefficients %>% as.data.frame()
tbl11$` ` <- cut(tbl11[[4]], 
                  breaks=c(-Inf, .001, .01, .05, .1, Inf),
                  labels=c("***", ***, "*", ".", ""))
tbl11[row.names(tbl11) %in% vars$lm,] %>%
  knitr::kable(align="rrrrc",
                col.names=c("Value", "Std. Error", "t value", "Pr(>|t|)", ""),
                digits=c(4,2,2,4,0))
```

```

In Table 2, we look at just those variables for which a one-percent change produces a change of greater than 0.2% in the time to non-viability for the Foyer, plus the scaling factors for the fire. Those variables include variables related to the room of fire origin (Lab\_depth), factors related to the space itself (Foyer\_width, Front\_Door\_Width), factors related to the spaces through which the fire must travel (Even\_Hallway\_depth, Odd\_Hallway\_width, Front\_Hall\_depth) and some additional miscellaneous factors (Office\_.2\_height Office\_.5\_height Gyp\_Conductivity.1 Gyp\_Density.1).

```
```{r tb12}
tbl12 <- summary(out$sr)$table %>% as.data.frame()
tbl12$` ` <- cut(tbl12[[4]], 
                  breaks=c(-Inf, .001, .01, .05, .1, Inf),
                  labels=c("***", ***, "*", ".", ""))
tbl12[row.names(tbl12) %in% vars$sr,] %>%
  knitr::kable(align="rrrrc",
                col.names=c("Value", "Std. Error", "t value", "Pr(>|t|)", ""),
                digits=c(4,2,2,4,0))
```

```

```
```{r cleanup}
rm(list=setdiff(ls(), keep))
save.image(file=paste0(params$save_file, ".RData"))
```

```