Computational Approaches to Fire-structure Interaction and Real-time Fire Monitoring

by

Paul A. Beata

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Civil Engineering and Scientific Computing)
in the University of Michigan
2017

Doctoral Committee:

Associate Professor Ann E. Jeffers, Chair
Professor Sherif El-Tawil
Professor Krishnakumar R. Garikipati
Professor Vineet R. Kamat

Paul A. Beata

pbeata@umich.edu

ORCID iD: 0000-0002-0445-4558

Dedication

This dissertation is dedicated to my fiancée and my inspiration: Lucia. I would not have been able to complete this research without her love and support for the last five years. I want to thank her for understanding me and working together to keep our long-distance relationship strong.

Acknowledgements

My advisor Dr. Jeffers has influenced my research, education, and life in so many positive ways. I want to thank her as my advisor and mentor, especially for the patience, understanding, and guidance through the difficult times of graduate school. Most importantly, Dr. Jeffers allowed me to explore my own interests in research and coursework and encouraged independent growth. I would also like to thank my committee: Dr. Kamat, Dr. El-Tawil, and Dr. Garikipati. They have been very helpful in this time leading up to the defense and have always been so nice to me over the years. Additionally, Dr. Kochunas, who guided me through his class on scientific computing and encouraged me during labs and office hours, has provided a major influence in my career decisions and goals. Dr. Prevatt at the University of Florida helped me develop as an undergraduate researcher and prepared me for graduate school. It was a pleasure working and learning alongside all of my friends at Michigan from CEE, everyone from our 1267 office, and our awesome research team (Qianru, Ha, and Alyssa). Also from our team, I want to thank my research brothers and best friends, Ning and Jason, for our strong friendship and for all their support over the years; we shared the best memories of grad school together. Finally, my parents and sister (Paul, Michelle, and Gianna), who supported my decision to leave Florida and pursue my goals. They provided me with the love and care that I needed to finish my degree, especially when things were the most challenging. I especially want to thank my parents for their lifelong guidance and encouragement. I would not have been able to complete this degree without my professors, friends, family, and my fiancée, Lucia; I am extremely grateful for all of their support.

Table of Contents

Dedication ..... **ii**

Acknowledgements ..... **iii**

List of Figures ..... **viii**

List of Tables ..... **xv**

Abstract ..... **xvi**

Chapter 1 Introduction **1**

1.1 Task 1: The Fire-structure Interaction Problem 2

1.2 Task 2: Computational Framework for Real-time Fire Monitoring 5

1.3 Organization of the Dissertation 7

1.4 References 9

Chapter 2 Spatial Homogenization Algorithm for Bridging Disparities in Scale between the Fire and Solid Domains **10**

2.1 Introduction 11

2.2 Governing Equations 15

2.2.1 2D Formulation 17

2.2.2 3D Formulation 20

2.3 Implementation 24

2.3.1 2D Implementation 25

2.3.2 3D Implementation 26
2.4 Results 27
2.4.1 2D Application 27
2.4.2 3D Application 32
2.5 Conclusions 39
2.6 Acknowledgements 40
2.7 References 40
Chapter 3 Thermo-mechanical Shell Element for Coupled Fire-structure Analysis 42
3.1 Introduction 42
3.2 Coupled thermo-mechanical shell element 46
3.2.1 Thermal shell element 47
3.2.2 Coupling approach 49
3.2.3 Mechanical shell element 52
3.2.4 Coupled shell element 55
3.2.5 Through-thickness integration 59
3.3 Numerical results 62
3.3.1 Thermal stress in a cylinder 63
3.3.2 Simply supported heated plate 65
3.4 Conclusions 69
3.5 Acknowledgements 70
3.6 References 70
Chapter 4 Applications of the Thermo-mechanical Shell Element in Coupled Fire-structure Analyses 72

4.1 Introduction 72
4.2 Fire-structure Coupling using Shell Elements 76
4.3 Plate Exposed to Local Fire 78
4.4 Beam Exposed to Local Fire 84
4.5 Analysis of the Time Step and Mesh Size 86
4.6 Discussion of Results 94
4.7 References 96
Chapter 5 Real-Time Fire Monitoring for the Post-Ignition Fire State in a Building 97
5.1 Introduction 97
5.2 LCM-Based Computing Infrastructure 105
5.3 Data Collection for Fire Monitoring 110
5.4 Event Detection Model 116
5.4.1 Hazard 1: Smoke Toxicity 119
5.4.2 Hazard 2: Burn Threats 123
5.4.3 Hazard 3: Fire Status 125
5.4.4 Summary and Testing 128
5.5 Visualization using BIM 129
5.6 System Testing with Simulated Fire 138
5.7 Performance Testing for the Real-time Requirement 148
5.8 Conclusion 154
5.9 Acknowledgements 155
5.10 References 155
Chapter 6 Conclusion 159

6.1 The Fire-structure Interaction Problem 160

6.2 Computational Framework for Real-time Fire Monitoring 161

6.3 Limitations and Future Work 163

Appendix A 165

List of Figures

Figure 2-1: Differences in scale for a sequentially coupled analysis (the extreme case of a fire impinging on a structure is shown) 12

Figure 2-2: Macro heat transfer elements: (a) fiber heat transfer element [15] and (b) plate/shell heat transfer element [16] 14

Figure 2-3: 2D elements exposed to non-uniform surface flux: (a) non-uniform flux and (b) equivalent nodal fluxes 17

Figure 2-4: Overview of the four methods developed for 2D spatial homogenization 18

Figure 2-5: Isoparametric element with non-uniform boundary condition 20

Figure 2-6: 3D shell heat transfer element with non-uniform thermal load 21

Figure 2-7: Mapping between real and natural coordinates 21

Figure 2-8: A fine mesh for measuring fluxes in the CFD domain superimposed on a single finite element with nine nodes in this case 22

Figure 2-9: Integration over a surface using the trapezoid rule 22

Figure 2-10: Relation between integration point sampling schemes and the relative mesh sizes in the CFD and FEA domains 23

Figure 2-11: Flowchart for sequentially coupled analysis 25

Figure 2-12: 2D application involving a solid exposed to a random heat flux 27

Figure 2-13: Heat flux at the integration points: (a) averaging method, (b) sampling method, (c) least squares method, and (d) trapezoid rule 30

Figure 2-14: Heat flux at the integration points: (a) averaging method, (b) sampling method, (c) least squares method, (d) trapezoid rule method 31
Figure 2-15: Fire simulation in FDS 33
Figure 2-16: Mesh configurations used in the test: (a) shell element models of 2×1, 4×2, 8×4, and 16×8 elements, each containing five layers through the thickness (not depicted); (b) solid element model with four elements through the thickness 34
Figure 2-17: Incident heat flux over time for a sensor at the center of the plate 35
Figure 2-18: Contours of the temperature field at the mid-surface of the plate after 12 minutes of fire exposure (a) 2×1 shell model, (b) 4×2 shell model, (c) 8×4 shell model, and (d) 16×8 shell model, and (e) 80×40 solid element model 36
Figure 2-19: Section for contour plots taken through the thickness 37
Figure 2-20: Temperature field through the thickness at 6 min (top) and 12 min (bottom): (a) coarse shell mesh of 2×1, (b) fine shell mesh of 16×8, and (c) the solid element model 38
Figure 3-1: Overview of the sequentially coupled fire-structure interaction problem using CFD-to-FEA coupling methods 43
Figure 3-2: Layered thermal shell element, as originally presented in [5] 48
Figure 3-3: Corner node at the mid-surface of the shell element, as originally presented in [5] . 49
Figure 3-4: Verification problem for analyzing thermal stress in a cylinder 64
Figure 3-5: Simply supported rectangular plate geometry and boundary conditions; the linear temperature through the thickness *h* is shown as well 66
Figure 3-6: Simply supported plate with displacement boundary conditions explicitly defined for the 3D FEA model employing shell elements 66

Figure 3-7: Comparison of the results for the out-of-plane displacement of the plate along the two centerlines of the domain using thermo-mechanical shell elements 68
Figure 4-1: Model details and material properties for the flat plate exposed to a local fire 79
Figure 4-2: Temperature and displacement results at the center of the plate; temperatures provided at the top, middle, and bottom (highest temperatures) of the plate at this central point 80
Figure 4-3: Temperatures on the heated surface along the two centerlines of the plate at 3 min and 10 min using four different meshes 80
Figure 4-4: Displacement along the two centerlines of the plate at 3 min and 10 min 81
Figure 4-5: The meshes used for each test (top), the resulting temperature field for the heated surface at the final time step (middle), and the corresponding out-of-plane displacement at this time (bottom) 82
Figure 4-6: Convergence plot for temperature and displacement; computed using the two-norm relative error between each mesh with the finest mesh (16×32) 83
Figure 4-7: Beam model exposed to a local fire with dimensions and material properties provided; the fire is off center by 0.5 m in the CFD model 84
Figure 4-8: Temperature and displacement results at the final time step along the neutral axis (middle of the web) and along the bottom flange centerline; the right image shows the displacement from the centerline (i.e., displacement U1 along the x-axis in the Abaqus model) 85
Figure 4-9: FDS model for a particular configuration used in the mesh size and time step study 87
Figure 4-10: Comparison of the various mesh sizes used in the study and the corresponding boundary data grids; the number above each mesh is the ratio of the FEA element size (green outline of each element) to the CFD cell size (red dot for the center of each cell) 89

Figure 4-11: Diagram showing the locations of points on the plate selected for the temperature-time plots 90
Figure 4-12: Temperature-time curve at Point A for the uniform boundary condition 91
Figure 4-13: Temperature-time curve at Point B for the bilinear boundary condition 91
Figure 4-14: Temperature-time curve at Point A for the FDS-based boundary conditions 91
Figure 4-15: Temperature results at the final step for either point A or B (as in previous figure) with respect to the FEA mesh size for the three cases of (a) uniform, (b) bilinear, and (c) FDS boundary conditions 93
Figure 4-16: Temperature results at the final step for either point A or B (as in previous figure) with respect to the subcycling time step for the three cases of (a) uniform, (b) bilinear, and (c) FDS boundary conditions 94
Figure 5-1: The use of data measured at the scene of the fire event could provide an informed response using real-time computation augmented with the traditional improvised response 99
Figure 5-2: A three-component view of the major features used for real-time fire monitoring; data is collected within the building during the fire event and used for computation and visualization in the proposed system 103
Figure 5-3: Overview of the proposed LCM-based monitoring system for real-time fire monitoring; the various colors of the links between different system components represent unique channels of communication in the LCM framework 107
Figure 5-4: The LCM data structure used as a common data type between the main monitoring program and the event detection sub-model (file `sim_sensor.lcm`) 108
Figure 5-5: Abbreviated version of the main fire monitoring program `main_rtfm.py` 109
Figure 5-6: Abbreviated version of the main event detection model program `main_edm.cpp` 110

Figure 5-7: The timeline of progression of a fire in a building; the aim for the current work is to provide a monitoring tool for the post-ignition state 113
Figure 5-8: The multi-threaded model for simulating multiple sensors with nominal time steps listed on the timeline to the left; each sensor is mapped to its own thread which accesses its own unique data file to read and push new data to the main monitoring program using an LCM data structure..... 116
Figure 5-9: The post-processing step was included for use after fire monitoring in order to create XML-based schedules for animation in ABD; the tasks in the schedule were automatically created using start and finish times of each threat level based on the event detection model output 133
Figure 5-10: Screenshot of the 3D BIM exterior walls and colored floor plates, highlighted here in magenta, used for visualizing threat levels; tasks in the XML schedules were tied to the magenta-colored floor elements..... 134
Figure 5-11: Screenshot of the Animation Producer in Bentley ABD with the imported XML tasks present in the *Schedule* tree..... 135
Figure 5-12: The four possible tasks are shown on the left with their corresponding colors; the actual representation of a "warning" task is shown to the right for Room 1 in XML format, where the start/finish times and colors were automatically generated from output generated by the event detection model..... 136
Figure 5-13: Sample live data captured and plotted in real-time automatically in the monitoring simulation; these plots were generated on the visualization device (Windows) while the incoming data was being processed on the computing workstation (Ubuntu) 137
Figure 5-14: The floor plan of the fire simulation is shown with the locations of the two fires (blue square in Room 1 and red square in Room 2); the sensor measurements used in this test were

recorded at the centers of each room, specifically at the ceiling level, and are marked with green circles in the image 138
Figure 5-15: Time history of the gas temperature output from the FDS fire simulation for the four-room model; subsequently used as input for the fire monitoring system 140
Figure 5-16: Time history of the radiative heat flux output from the FDS fire simulation for the four-room model; subsequently used as input for the fire monitoring system 140
Figure 5-17: Time history of the concentrations of carbon monoxide, oxygen, and carbon dioxide output from the FDS fire simulation for the four-room model; subsequently used as input for the fire monitoring system 141
Figure 5-18: Results of the real-time FED computation for smoke toxicity and burn threats due to heat; results were computed using measurements at one sensor location per room every 2.0 seconds as received by the event detection model for real-time calculation 142
Figure 5-19: The FDS simulation is shown at 201 seconds, just after the warning for fire status in Rooms 1 and 2 had increased to Level 1 143
Figure 5-20: Results from the event detection model for each of the three hazards: a) smoke toxicity, b) burn threats, and c) fire status; the warning levels of {0, 1, 2} are whole integers at every increment in the monitoring simulation using a 2.0-second time step 144
Figure 5-21: Sample results using visualization in BIM; each stage of the threat progression in Room 2 is provided, where the FDS model is shown on the left and the Bentley ABD BIM with imported schedules appears on the right 147
Figure 5-22: Computational cost quantified by the average time required for the event detection model to process the data; the maximum observed cost for a single data message is also provided as the upper bound 153

Figure A-1: The pyramid of software development 173

Figure A-2: A view of the top-level repository for `fire_main`, which is the main repository hosting the research project 175

Figure A-3: A view of the top-level repository for `fire_tpls` which hosts the required TPLs for the research project 177

Figure A-4: Terminal output from the automatic TPL installation process..... 181

Figure A-5: The build script for the main TriBITS project with flags for including LCM..... 183

Figure A-6: Results of `ctest` in the `packages/RTFM` subdirectory 184

Figure A-7: A sample of the `install_tpls.sh` demonstrating the TPL installation logic. 185

List of Tables

Table 2-1: Mesh properties for the 2D case 28

Table 2-2: Vector norm for relative differences in temperature 32

Table 2-3: Mesh properties for the 3D case 34

Table 2-4: Comparison between the shell model and the solid element model 37

Table 4-1: Properties of the FEA models and the required simulation times; note that the temperatures and displacements are the results at the final time step 83

Table 4-2: Overview of the variables used in the testing matrix 88

Table 5-1: Thresholds and conditions used to define the event detection model 128

Table 5-2: Temperature, heat flux, and species concentration input for the event detection model test based on the SFPE Handbook example [40] 129

Table 5-3: Results computed from the event detection model for the example in the SFPE Handbook [40] 129

Table 5-4: Start times, in seconds, for each of the three threat levels based on event detection model output; the XML schedule generator determined these start times automatically and created the corresponding tasks for schedule simulation 145

Table 5-5. Quantification of the system performance with respect to increasing number of sensors; the average computational cost, maximum observed single cost, and the standard deviation are all provided as well as the percentage of measurements that required ≤ 1.0 ms 152

Abstract

Structural fires in buildings are a persistent hazard in modern infrastructure and a potential threat to occupants and firefighters alike. In the pre-construction phases of building design, there has been a growing interest in the use of simulated natural fires as a direct input for modeling the fire-structure interaction problem. One common method to simulate the natural fire is through the use of computational fluid mechanics (CFD), where the fire development is treated as a fluid-flow problem. The structure is typically modeled using finite element analysis (FEA), as in other applications of structural engineering. However, the link between the fire and solid domains has not been standardized for structural fire engineering purposes in research or practice. In the post-construction phases of the building lifecycle, for example, fire safety engineering is used to reduce the potential for fire and minimize threats to building occupants. While there have been many improvements in materials and design to attempt to limit the ignition of new fires in buildings over the last several decades, the fact remains that fire events still pose significant risks to firefighters. With this in mind, methods must be developed to provide new technology and solutions for the modern firefighter by using advances in computation and visualization.

In this dissertation, modeling the fire-structure interaction problem and providing a real-time system for fire monitoring are the two main focuses. Both contributions serve to improve the two different ends of the structural fire spectrum: research, analysis, and design on one end and sensor-assisted firefighting on the other.

First, to address the problem of modeling the fire-structure interaction, an approach is provided which used CFD-based boundary conditions from a fire simulation as input for the FEA-

based model of the structure. The main components of this research are (i) the development of fire-to-structure coupling methods linking the fluid and solid domains, (ii) the extension of a layered thermal shell element to include mechanical degrees of freedom (DOF) and provide a coupled thermo-mechanical shell element, and (iii) the use of these two contributions together to analyze structures exposed to local fires. The trapezoidal rule for numerical integration was used to represent spatially non-uniform heat fluxes computed in the CFD fire simulation as equivalent nodal fluxes in the FEA model of the structure. The thermo-mechanical shell element was coupled from its individual formulations using virtual work methods and ensuring consistency in the use of the layered representation of the governing equations for conduction heat transfer and structural deformation in the element.

Second, a proposed system for real-time fire monitoring in the post-ignition fire state of a building was developed to improve the technology of sensor-assisted firefighting from a computing perspective. The software system was designed to coordinate various data streams from a simulated wireless sensor network (WSN) to sub-models responsible for performing real-time calculations using the data measured by sensors. An event detection model for assessing smoke toxicity, burn threats, and fire spread was implemented as the first sub-model and its real-time performance was analyzed. Results from the event detection model were used to present an example of visualization using a building information model (BIM) as the platform for communicating hazard warnings to the incident commander and firefighters in this study.

Chapter 1
Introduction

At the intersection of fire safety and structural engineering is the field of structural fire engineering. The researchers who exist in this space are interested in providing solutions to the communities within fire protection engineering, which include members from civil, mechanical, and chemical engineering as well as material science. The variety of research at the intersection is reflected in the contributions presented here as this dissertation aims to provide solutions for two different groups in these fields. The overarching theme of this work is the development and use of computational methods and software tools to provide a better understanding of structural fires: from the mechanics and simulation aspects to firefighting and decision making.

The common thread connecting fire safety (sprinkler design, evacuation, fire science, etc.) and structural fire engineering (for example, designing structures for fire resistance) is the desire to make the built environment safer for the occupants during the lifetime of the building. In the case of analysis and design, the goal is to understand the behavior of the fire as a hazard to the structure and occupants, as well as to provide a safer infrastructure for the public. Passive forms of fire suppression may be included in an attempt to limit the ignition of new fires and the subsequent effects of accidental fires. However, much of fire safety science trends into the post-ignition state as well: the situation in which a fire is present and needs to be handled through the use of active fire suppression technology and human intervention by the fire department.

The goal of this dissertation is to provide a contribution to both the pre-design phase (research, analysis, and simulation) as well as to the post-ignition state (firefighting intervention)

with two main tasks. In the interest of researchers and analysts, a thermo-mechanical shell element for coupled fire-structure simulation is presented first. Then, for the purpose of occupant and firefighter safety, a computational system for sensor-assisted firefighting is proposed. These two tasks each seek to increase safety in the built environment from the two different aspects of structural fire engineering through computational mechanics and fire safety through intelligent firefighting using computational tools. The first task is focused on fire-structure coupling using CFD-FEA simulation with thermo-mechanical shell finite elements and the representation of thermal boundary conditions as equivalent nodal fluxes. The second task is focused on real-time fire monitoring using multiple fire signatures to assess hazards and provide graphical warnings to firefighters during the post-ignition fire state in the building.

Although these topics focus on different areas of the problem of structural fires in buildings, they both serve to provide new technology and approaches to one field that has traditionally relied on prescriptive building codes (structural fire engineering) and another which has not received the benefits of sensor networks at the same pace as other engineering fields (fire safety and firefighting). The following introduction to the individual topics describes the computational-mechanics-based approaches used for modeling the fire-structure interaction problem and then the methods used for providing a software framework for real-time fire monitoring and sensor-assisted firefighting.

1.1 Task 1: The Fire-structure Interaction Problem

In the field of structural fire engineering, there has been a growing interest over the last decade in the use of CFD to model natural fire scenarios in structures. The primary software used for this purpose is Fire Dynamics Simulator (FDS) [1]. However, one of the persistent challenges

for this field is the ability to employ such high-resolution fire simulations in the analysis of structures exposed to fires: the software tool exists, but its feasibility for use in research and analysis at the structural level is prohibitive. The difficulty lies in the fact that FDS is mainly used for modeling the combustion, chemistry, and corresponding fire growth of the fuel source using CFD. Also note that other researchers and analysts interested solely in smoke spread may also use FDS as well.

In the context of the thermal and mechanical analysis of structural components, structures in the flow field of the fire, which is modeled as a fluid, are represented as solid obstacles with specific material properties, but the conduction model of the solid domain is not normally considered in the simulation. Thus, to perform a thermal analysis of the structure with non-uniform heating from a natural fire, some representation of the flow field acting on the structure must be transferred from the fire domain to the solid domain, where the solid model may be represented by a finite element model typically. This transfer of data from the fire simulation (FDS) to the solid model in a separate finite-element modeling software (e.g., Abaqus, ANSYS, deal.II, etc.) is challenging because of the different time increments used in each domain as well as the different spatial meshes.

Approaches to the fire-structure coupling problem using CFD and FEA can be divided into two particular methods: flux-based and temperature-based methods. The temperature-based methods typically include the use of an adiabatic surface temperature (AST), which is used to represent the non-uniform boundary conditions at specific points in the CFD domain on the surfaces of obstacles in the fire flow field as surface temperatures in the corresponding FEA model. The flux-based methods are designed to use incident, net, radiative, and/or convective heat fluxes measured on the surface of the obstacle in the CFD model as input for the FEA model through the

use of the corresponding non-uniform surface flux boundary conditions. In either case, from the analyst's perspective, these various forms of output from a fire simulation are readily available in the FDS software.

In this dissertation, the class of flux-based methods for coupling the fire to the structure were developed for mapping net and incident heat flux output from the FDS fire simulation to the appropriate locations in the domain of the FEA thermal model in order to provide representations of the temporally and spatially varying non-uniform boundary conditions on exposed surfaces in the solid domain. The differences in the time increments between the two domains were handled in a previous study [2] and the methods used for linking the two dissimilar spatial domains were developed here.

Specifically, the surface fluxes were measured along a fine grid in the CFD domain on the exposed exterior of the structure's surface and transferred to the FEA model. Then, the trapezoidal rule was used to compute equivalent nodal fluxes as a forcing term to use in the conduction heat transfer model (FEA). The proposed method is compared to other approximating techniques, including averaging, sampling, and least squares methods, for a 2D heat transfer problem. The results demonstrate that the proposed homogenization algorithm for handling the spatial non-uniformity in the boundary conditions provides solutions for the temperature field that converge rapidly due to the energy-equivalent representation of the thermal boundary condition. The homogenization algorithm is then implemented in a 3D heat transfer model that uses thermal shell elements to model conduction through the solid.

Once the fire-structure coupling technique was established in the conduction model, the coupled thermo-mechanical shell elements were formulated and implemented for use in the fire-structure interaction problem through the use of virtual work methods. The layered formulation

for the thermal shell was developed first by Jeffers [3] to provide the finite element for conduction heat transfer analysis. Then, a mechanical shell element with displacement DOF was chosen from the literature to represent the deformation in the coupled element [4]. Using the method of virtual work, the two physical responses of heat transfer and structural deformation were combined in the coupled thermo-mechanical shell element formulation. This formulation was then implemented as a Fortran subroutine for use with the commercial FEA software Abaqus, which provides for the inclusion of user-defined subroutines as custom elements.

Employing the thermo-mechanical shell elements in the coupled fire-structure simulation problem allows the analyst to perform this multi-step analysis in a single FEA model without re-meshing and by solving for temperatures and displacements simultaneously. The current study demonstrates the accuracy and convenience of this approach for calculating deformations in the structure due to non-uniform heating from a simulated fire source.

1.2 Task 2: Computational Framework for Real-time Fire Monitoring

Sensor-assisted firefighting and real-time computation are two main goals for the future of active fire intervention in the post-ignition state in buildings. In their pioneering work on sensor-assisted firefighting, Cowlard et al. [5] warns against some of the challenges of providing firefighting assistance in the form of data assimilation and visualization. The risk of *information overload* is ever-present and should be respected by any new technology designed for the purpose of improving the experience of extinguishing building fires. However, the call for research to provide solutions which assimilate data, real-time computation, and visualization was posed as a challenge to the field when this work was first published in 2010. To this end, the second topic of the dissertation is real-time fire monitoring and a focus on the proposed use of sensor

measurements to capture the progression of a fire in the building. Specifically, this topic is related to the post-ignition state, as opposed to the detection of new fires in a building. The intended end user for the fire-monitoring system described herein is the firefighter, where the work aims to provide a new contribution in the direction of sensor-assisted firefighting technology: a delicate topic requiring careful consideration of the intended user.

The range of topics in the real-time monitoring space include hardware-related research such as the development of robust wireless-sensor technology for extreme conditions and the invention of other supplemental data acquisition tools, communications research for transmitting data from the scene of the fire to individual firefighters or a centralized repository, and the life-safety and occupant-rescue aspects of hazard intervention. In particular, the real-time monitoring features covered in this dissertation are the components of computation and visualization for a distributed deployment system: a necessary supplemental technology for the true implementation of a future monitoring system. The problem is larger than one particular academic solution and the contribution of many researchers is necessary to provide field-ready tools for improving the practice of firefighting, including the necessary feedback from actual firefighters.

The chapter on fire monitoring serves to establish a new research branch into the field, focusing on the sound computational aspects required for such a technology. As a new endeavor, the fire monitoring software presented here was established from the start by using proven software engineering techniques that are critical for research code development and reproducibility: version control, installation scripts, and a build system. The purpose of this effort was to provide the software foundation upon which the present work was built and future work can continue through inter-university collaboration. By establishing the use of high-quality software engineering tools early on in this project, developers will be able to provide the eventual future users with robust

scientific software to perform the overarching goal of monitoring fire events in real time using reliable computational tools.

In its present state, the main contributions to the real-time fire monitoring problem are presented here with the appropriate focus on fire safety. The computational system was developed to handle incoming data from multiple sensor measurements and subsequently process this data in real time for a multi-room fire model. An event-detection model was designed to monitor the common hazards of smoke toxicity, burn threats, and fire spread that may occur during a real fire event. This model was linked to the main computing system using Lightweight Communications and Marshalling (LCM) [6] to coordinate data transfer between dissimilar applications. Finally, the potential for real-time visualization is demonstrated through the use of schedule simulation based on the output of the event detection model. Building information modeling software called AECOsim Building Designer (ABD) was used in a distributed system to create visualizations of the event detection model results as a post-processing feature. Integrating real-time measurements from sensors into the fire intervention strategy may provide a new technological advancement to the future practice of firefighting.

1.3 Organization of the Dissertation

The structure of this dissertation is based on the manuscript-style format. Each chapter that follows has either been published in a peer-reviewed journal, will be submitted to a journal, or exists as a stand-alone technical report; the distinction will be made clear in the following descriptions. Chapter 1, the current chapter, serves as the introduction and has provided an overview of the included material.

Contributions to the *fire-structure interaction problem* are described in three chapters. Chapter 2 presents the methods and applications of a fire-structure coupling technique from a journal article published in the Fire Safety Journal. These methods use the trapezoidal rule for numerical integration in two spatial dimensions to compute non-uniform thermal boundary conditions as the equivalent nodal forcing terms in a conduction heat transfer analysis. The heat transfer in the solid was modeled using the previously developed thermal shell element and the thermal boundary conditions were applied as non-uniform surface fluxes (temporally and spatially). Chapter 3 extends the formulation of the thermal shell element to include displacement DOF for use in coupled thermo-mechanical analyses; this manuscript will be submitted to Finite Elements in Analysis and Design. Additionally, the fire-structure coupling methods in Chapter 2 were employed in the new thermo-mechanical shell element of Chapter 3 for use in coupled fire-structure simulations. The final chapter on this topic, Chapter 4, is a technical report which describes this supplemental study to analyze the effects of relative time steps and mesh sizes on the solutions of the thermo-mechanical shell element models. Additionally, the coupled framework was used in the application of a structural I-beam exposed to a local fire.

The second topic of the dissertation is the *computational framework for real-time fire monitoring*, and it consists of one main chapter and a supplemental appendix. First, Chapter 5, is a manuscript which will be submitted to the Fire Technology journal and provides the fire safety contributions to the proposed monitoring system. In this chapter, the development of an event detection model, the computational framework for fire monitoring, and the subsequent use of real-time computing for visualization are presented. The corresponding appendix that follows is a technical report providing details on the initial establishment of the project as a new research-level software contribution. Here, the foundations of the computational fire-monitoring system were

developed through the use of modern scientific software engineering tools, such as version control and build systems.

Finally, Chapter 6 serves as the conclusion and summary of the work, including the limitations and future outlook for this line of research. References used in each of the chapters are provided with the individual chapters as they appear in the dissertation, as opposed to being collected at the end of the document.

1.4 References

[1] McGrattan, K., Hostikka, S., Floyd, J., Baum, H., Rehm, R., Mell, W., & McDermott, R. (2004). Fire dynamics simulator (version 5), technical reference guide. *NIST special publication, 1018*(5).

[2] X. Yu and A. E. Jeffers, "A comparison of subcycling algorithms for bridging disparities in temporal scale between the fire and solid domains," *Fire Saf. J.*, vol. 59, pp. 55–61, 2013.

[3] A. E. Jeffers and P. A. Beata, "Generalized shell heat transfer element for modeling the thermal response of non-uniformly heated structures," *Finite Elem. Anal. Des.*, vol. 83, pp. 58–67, Jun. 2014.

[4] W. Kanok-nukulchai, "A simple and efficient finite element for general shell analysis," *Int. J. Numer. Methods Eng.*, vol. 14, no. 2, pp. 179–200, 1979.

[5] A. Cowlard, W. Jahn, C. Abecassis-Empis, G. Rein, and J. L. Torero, "Sensor Assisted Fire Fighting," *Fire Technol.*, vol. 46, no. 3, pp. 719–741, Jul. 2010.

[6] A. S. Huang, E. Olson, and D. C. Moore, "LCM: Lightweight communications and marshalling," in *Intelligent robots and systems (IROS), 2010 IEEE/RSJ international conference on*, 2010, pp. 4057–4062.

Chapter 2
Spatial Homogenization Algorithm for Bridging Disparities in Scale between the Fire and Solid Domains

The analysis of structures exposed to non-uniform heating from localized fires is a challenging task due to the spatially varying boundary conditions and the differences in scale between the fire simulation and solid heat transfer model. This chapter presents a spatial homogenization algorithm for capturing non-uniform boundary conditions from a high-resolution fire simulation in a low-resolution finite element heat transfer model of a structure. The homogenization algorithm uses numerical integration by the trapezoid rule to calculate the equivalent thermal flux vector in the finite element heat transfer model for a spatially varying surface flux. The proposed method is compared to other approximating techniques, including averaging, sampling, and least squares methods, for a 2D heat transfer problem. The results demonstrate that the proposed homogenization algorithm converges rapidly due to the energy-equivalent representation of the thermal boundary condition. The homogenization algorithm is then implemented in a 3D heat transfer model that uses macro-level plate elements. For an application involving a horizontal plate exposed to a localized fire, the model is shown to converge to the results obtained by a solid finite element model. The homogenization algorithm combined with the plate heat transfer element proves to be an accurate and highly efficient means for analyzing structures with spatially varying thermal boundary conditions calculated by computational fluid dynamics.

2.1 Introduction

Current methods for the fire-resistant design of structures emphasize the design for post-flashover fire conditions and commonly use parametric fire models such as those found in Eurocode 1 [1]. Post-flashover fire models assume that the gas temperature is uniform within the compartment. Although intending to represent a worst-case structurally significant fire, post-flashover fire models are limited to relatively small compartments with regular geometries. In large open spaces, fires tend to burn locally and oftentimes will travel across the floor plate and even spread across floors [2]. Under these conditions, the fire behavior can be rather complex and may be more accurately represented by a computational fluid dynamics (CFD) model.

CFD models such as Fire Dynamics Simulator (FDS) [3] are well-established and provide an accurate representation of natural fire events. However, challenges arise when coupling the CFD fire model to a solid heat transfer model due to incompatibilities in mesh and disparities in scale between the fire and solid domains. Figure 2-1 illustrates the disparities in spatial and temporal scales for a one-way coupled system. Note that an extreme case of a fire impinging on the structure is shown for contrast. It can be seen that the fire simulation requires a time step and spatial resolution that is considerably smaller than is needed in the structural analysis due to fire dynamics being a “fast” physics. Yu and Jeffers [4] demonstrated that a time-averaged subcycling algorithm can overcome the differences in temporal scales between the fire simulation and solid heat transfer model in an accurate and efficient manner. However, the difference in spatial scale becomes particularly important when simulating the 3D temperature gradients in the structure.

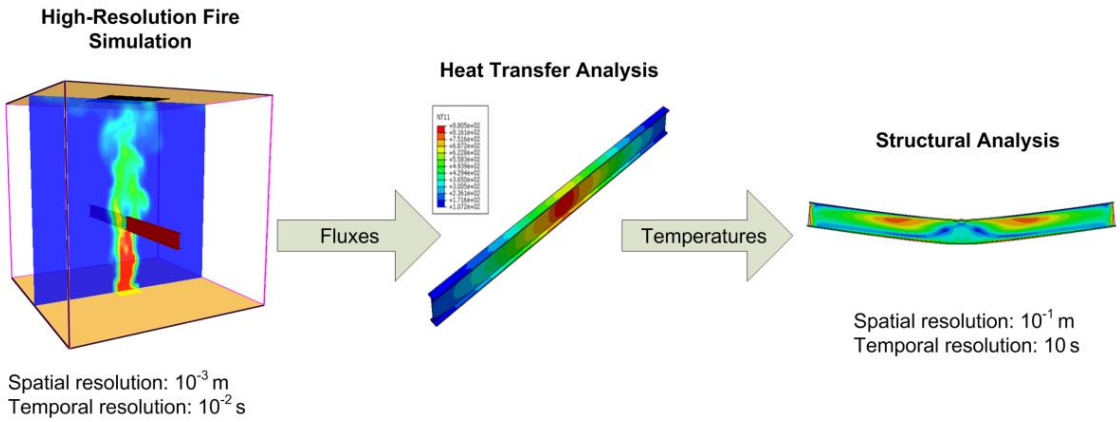


Figure 2-1: Differences in scale for a sequentially coupled analysis (the extreme case of a fire impinging on a structure is shown)

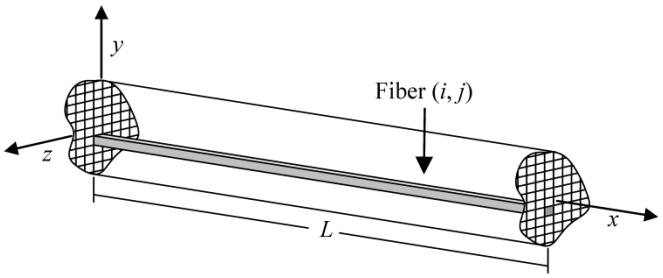
There have been a few efforts in recent years to couple a CFD fire model to a structural model in order to study structural response under natural fire effects. The NIST investigations into the World Trade Center collapses employed high-fidelity CFD fire models in conjunction with macro finite element structural models [5]. In the NIST investigations, the boundary conditions were expressed as a locally uniform gas temperature that was calculated by the CFD analysis. In a similar vein, Tondini et al. [6-7] established a framework to couple a CFD fire model to a structural model based on a locally uniform gas temperature. Wickstrom et al. [8] demonstrated that the boundary condition at the fire-structure interface could be expressed in terms of adiabatic surface temperature, which decouples the fire exposure from the structural surface temperature. These approaches are most suitable for situations in which the fire heats the structure by remote radiation or by heat transfer associated with an optically thick gas, in which case the fire exposure is relatively uniform over the structure's surface. Temperature gradients along the lengths of members can be captured by discretizing the structural members into more elements [6]. However, temperature gradients along the lengths of members were not the focus of prior studies.

Non-uniform heating in general is known to have a significant effect on the structural response [9-12]. Additionally, it has been shown that localized heating associated with a fire

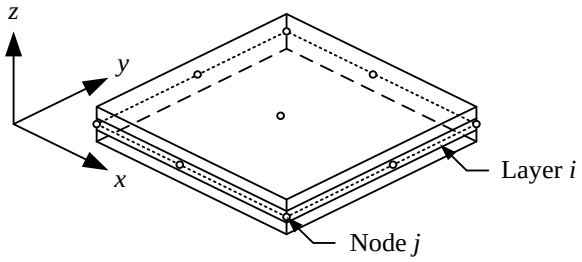
impinging on a structure can be detrimental to the structural performance because it can produce structural effects (e.g., local buckling) that are not seen when the structure is heated uniformly [13]. It has not yet been determined whether fluctuations in boundary conditions due to a real fire impinging on a structure play a significant role in the thermo-mechanical response of the structure. It is clear that unprotected structures will be more sensitive to such localized effects, but even for unprotected members the effects have not been quantified. To this end, Chen et al. [14] established a coupling interface between a CFD fire model and finite element heat transfer model of a structure based on convection between the structure and a gas of non-uniform temperature. While able to simulate non-uniform heating along the lengths of members, the work by Chen et al. used 3D solid elements in the finite element model, resulting in a structural model that required excessive computational expense due to the fact that a fine mesh was needed to capture the cross-section response. Thus, although the CFD analysis continues to drive the total computing time in coupled fire-structure models, it is apparent that 3D solid finite elements are unnecessarily inefficient for thin-walled structures, leaving room for further improvement.

To improve computational efficiency in the thermo-mechanical analysis of beams, plates, and shells, recent efforts by Jeffers et al. [15-17] have led to a class of macro heat transfer elements that can simulate the response of non-uniformly heated structures in an accurate and computationally efficient manner. Specifically, beam, plate, and shell heat transfer elements (Fig. 2-2) were formulated for calculating the 3D thermal response of non-uniformly heated structures. A combination of finite element and control volume methods was used in the element formulations to solve the 3D conduction heat transfer equations in an accurate and efficient manner. The elements have a fiber-based or layered discretization to account for large temperature gradients over the cross-section or through the thickness, respectively. Temperatures along the length or in

plane are approximated by linear or quadratic shape functions. One advantage to the use of macro heat transfer elements is that the elements facilitate the transfer of data from a thermal analysis to a structural analysis because both models can have the same mesh.



(a)



(b)

Figure 2-2: Macro heat transfer elements: (a) fiber heat transfer element [15] and (b) plate/shell heat transfer element [16]

The macro elements have been proven to provide computational savings that are more than an order of magnitude over 3D solid elements. In the case of the plate heat transfer element, for example, it was shown in a 3D verification study that the element provided the same level of accuracy as a 3D solid model and required 1.3 to 2.6 s to perform an analysis that required 830.7 seconds for a 3D solid model [16]. The computational savings are related to the significant reduction in the number of temperature degrees of freedom that are needed in the heat transfer model. The macro elements also allow for a much coarser mesh along the length or in plane in the heat transfer analysis. In the analysis of a 4.0-m steel beam exposed to a localized fire, it was

shown that an element size of 50 cm to 100 cm in length provided a prediction of the thermo-mechanical response that was comparable to a 3D solid model [15]. In the analysis of a 6-m×6-m concrete slab exposed to a localized fire, temperatures were accurate for an element that was 37.5 cm in length in the previous study [18].

This chapter concerns a spatial homogenization algorithm for overcoming the differences in spatial scales between the fire simulation and solid heat transfer model, with an emphasis on conduction heat transfer evaluated by macro finite elements. For macro elements in particular, it is necessary to consider the case in which the CFD grid is significantly finer than the finite element grid and requires homogenization of the data from the CFD analysis. A numerical integration scheme based on the trapezoid rule is employed in the calculation of the equivalent nodal flux vector for the heat transfer finite element. The proposed homogenization algorithm is compared to other methods for representing the spatially varying boundary condition, including averaging, sampling, and least squares methods. Following the 2D verification study, this chapter considers a 3D application of a horizontal plate exposed to localized fire, in which the fire exposure is simulated by CFD and the solid heat transfer analysis is conducted using the plate heat transfer element in [16].

2.2 Governing Equations

The equations governing conduction heat transfer by finite element analysis are expressed as follows:

$$[C]\{\dot{T}\} + [K]\{T\} = \{R\} \tag{2.1}$$

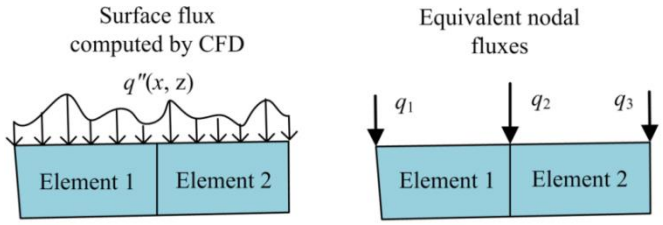
where $\{T\}$ is the field variable (temperature), $[C]$ is the heat capacity matrix, $[K]$ is the thermal conductivity matrix, $\{R\}$ is the vector of thermal loads, which includes heat flux, radiation, and convection boundary conditions [19]. For the applications considered here, it is assumed that incident heat fluxes q'' are calculated by CFD analysis and losses by convection and radiation are separately accounted for. Thus, the homogenization algorithm is presented for a flux boundary condition, although the methodology can readily be extended to convection and radiation boundary conditions. It is presently assumed for combustible solids that the pyrolysis gases flow outward from the surface, in which case there is no forcing term related to mass conservation that is passed into the solid model from the CFD analysis.

The vector of heat fluxes is obtained by summing the heat flux vectors for each of the elements in the structure. For an incident heat flux boundary condition, the vector $\{r\}$ for an element is expressed as

$$\{r\} = \int_S \lfloor N \rfloor^T \, q'' \, dS \tag{2.2}$$

where $\{N\}$ is an array containing the element's shape functions and q'' is the heat flux acting over surface S. Under natural fire conditions, q'' may vary greatly over the surface S due to turbulent flow as illustrated in Fig. 2-3a. Thus, the objective is to compute Eq. (2.2) as an equivalent nodal flux vector (Fig. 2-3b) that appropriately accounts for the non-uniformity in the boundary condition. This can be done by (i) replacing q'' with a function that approximates the randomly varying boundary condition with a smooth (e.g., linear) function, or (ii) expressing Eq. (2.2) in a discrete form that preserves the non-uniformity in q''. The former approach might be chosen for convenience due to compatibility with the standard finite element formulation, whereas the latter approach may be more precise, as shown in this chapter. Several methods were investigated for representing the non-uniform boundary condition for a 2D planar problem. Of the methods

investigated for the planar model, only the highest performing approach was used in the extension to a formulation for use with 3D finite elements.



(a) **(b)**

Figure 2-3: 2D elements exposed to non-uniform surface flux: (a) non-uniform flux and (b) equivalent nodal fluxes

2.2.1 2D Formulation

In each method explored in the current study, the goal was to approximate Eq. (2.2) given the randomly varying heat flux *q*". For a 2D isoparametric element, Eq. (2.2) can be expressed as

$$\{r\} = \int_{-1}^{+1} \lfloor N(\xi) \rfloor^T q''(\xi)\, tJ\, d\xi \tag{2.3}$$

where ξ is the natural coordinate ranging from -1 to 1; $\{N\}$ is an array containing the element shape functions, which are expressed in terms of ξ; *q*" is the heat flux, which varies in ξ; *t* is the thickness of the element; and *J* is the determinant of the Jacobian matrix relating the natural coordinates $\xi\eta$ to the global *xy* coordinates.

Four homogenization algorithms were investigated for handling a randomly distributed heat flux over the surface of an element, as illustrated in Fig. 2-4. The average value method applies a uniform heat flux equal to the average of the randomized fluxes over the surface of the element. The sampling method takes the flux values at the end points of the element and interpolates linearly between them. The least squares method fits the randomized flux data with a linear least squares

approximation. The trapezoid method preserves the randomized flux data and uses the trapezoid rule to numerically evaluate the integral in Eq. (2.3). The averaging, sampling, and least squares methods aim to smooth the randomized flux with a linear function, enabling the Gaussian quadrature rule to be used for numerical integration, whereas the trapezoid method expresses Eq. (2.3) in a discrete form that captures the randomized flux exactly. It should be noted that the trapezoid rule is superior to Gaussian quadrature for the integration of a random function because Gaussian quadrature is best-suited for the integration of low-order polynomials. However, averaging, sampling, and least squares methods are convenient methods for smoothing the data from a CFD analysis. This chapter considers the implications of smoothing the data vs. integrating the function in a more exact sense by studying the structural temperatures in realistic fire safety engineering applications.

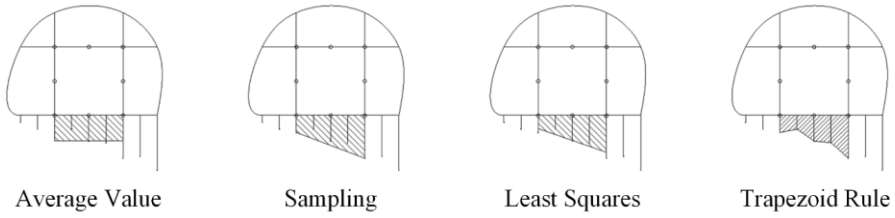


Figure 2-4: Overview of the four methods developed for 2D spatial homogenization

The integral in Eq. (2.3) was evaluated numerically in each of the four methods. Numerical integration of the element's flux vector $\{r\}$ was accomplished using Gaussian quadrature in the averaging, sampling, and least squares methods, whereby the integral in Eq. (2.3) is expressed as

$$\{r\} \approx \sum_{i=1}^{n} W_i \left\lfloor N(\xi_i) \right\rfloor^T q''(\xi_i)\, tJ \tag{2.4}$$

In Gaussian quadrature, W_i represents the weighting constant associated with the sampling point ξ_i, which lies in the domain [-1, 1] along the isoparametric element's edge. The heat flux $q''(\xi)$ is

evaluated at Gauss point ξ_i based on the homogenization method that is used. The summation is carried out over *n*, the number of sampling points based on the quadrature rule that is employed.

The trapezoid rule was explored as an alternative to the methods based on Gaussian quadrature. This method was intentionally selected because it uses all of the data associated with the element in calculating the equivalent nodal flux vector (as opposed to sampling at the Gauss points). In this manner, the contribution of each measured heat flux from the fire simulation is accounted for as opposed to a sampling technique which could fail to catch peaks in the heat flux. By integrating the random flux over the surface, the nodal flux vector that is calculated by the trapezoid rule is equivalent in energy to the random heat flux that exists at the element's surface.

Figure 2-5 shows the application of the trapezoid rule in 2D for a non-uniform heat flux $q''(\xi)$ at the $\eta = 1$ edge of the element. The data points refer to sensor data from the CFD fire simulation for incident heat fluxes. As shown in the figure, five heat flux data points are used to approximate the integral with four trapezoidal segments. In the general case for 2D elements using this method, $n + 1$ points are used for n trapezoidal segments along a given element's edge. When using data from CFD, the width of each trapezoid segment is dependent on the spacing of the sensors in the fire simulation (i.e., the fidelity of the CFD mesh). It is assumed that the number of data points between end nodes is large in comparison to the number of nodes (a reflection of the differences in scale between the fire and solid heat transfer models). The integral in Eq. (2.3) is evaluated by applying the trapezoid rule using incident heat flux data from the fire simulation. For $n + 1$ data points,

$$\{r\} \approx \lfloor N(\xi_0) \rfloor^T \, q''(\xi_0) \, tJ + 2\sum_{i=1}^{n-1} \lfloor N(\xi_i) \rfloor^T \, q''(\xi_i) \, tJ + \lfloor N(\xi_n) \rfloor^T \, q''(\xi_n) \, tJ \qquad (2.5)$$

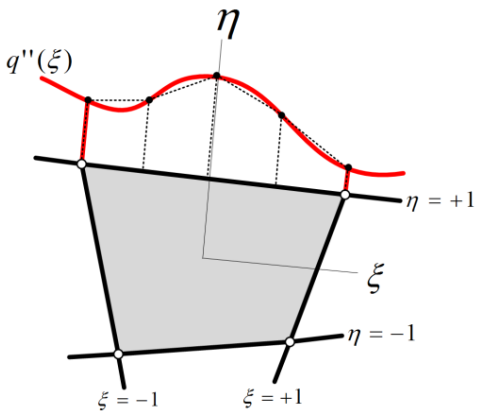
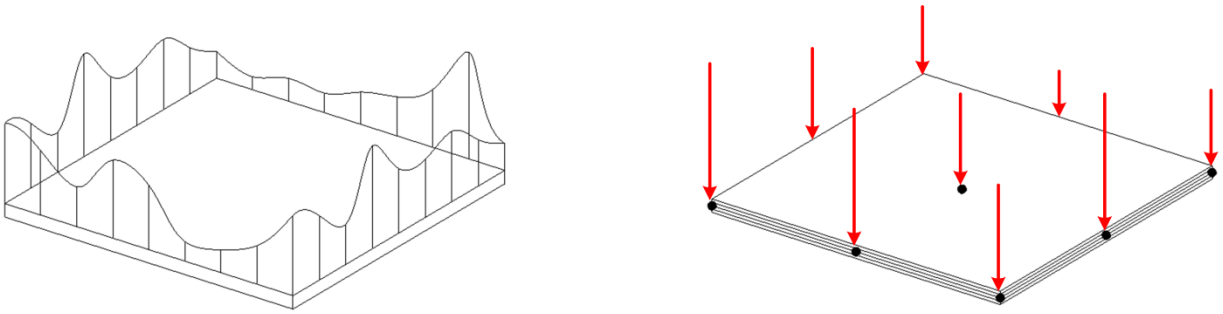


Figure 2-5: Isoparametric element with non-uniform boundary condition

2.2.2 3D Formulation

The extension to 3D is essential for using the spatial homogenization algorithm in the calculation of equivalent nodal fluxes for non-uniform thermal boundary conditions applied over surfaces as opposed to edges of the element. Fig. 2-6a shows a spatially varying heat flux over the surface of a macro heat transfer element. As in the 2D case, equivalent nodal fluxes are to be calculated from the discrete data and applied at the nodes, as shown in Fig. 2-6b. The homogenization algorithm based on the trapezoid rule may be extended to a second dimension by simply applying the trapezoid rule to several slices of data in one dimension (for example, along the ξ-direction) and then subsequently applying the trapezoid rule in the second dimension (along the η-direction).



(a) **(b)**

Figure 2-6: 3D shell heat transfer element with non-uniform thermal load

The heat flux vector of Eq. (2.3) can be re-written by adding another dimension to the expression of the heat flux vector for integrating over the surface:

$$\{r\} = \int_S \lfloor N \rfloor^T \, q'' \, dS = \int_{-1}^{+1} \int_{-1}^{+1} \lfloor N(\xi,\eta) \rfloor^T \, q''(\xi,\eta) \, J \, d\xi \, d\eta \tag{2.6}$$

Note that the shape functions N, heat flux q", and Jacobian J are now functions of two coordinates, ξ and η, representing the surface S over which the heat flux is applied. The Jacobian is used to map between the real space and the natural coordinates of the isoparametric element, as shown in Fig. 2-7. Note that the heat transfer finite element can have arbitrary geometry due to the isoparametric formulation. Thus, it is not necessary that the finite element grid overlay the CFD grid, provided that the CFD model can handle complex solid boundaries.

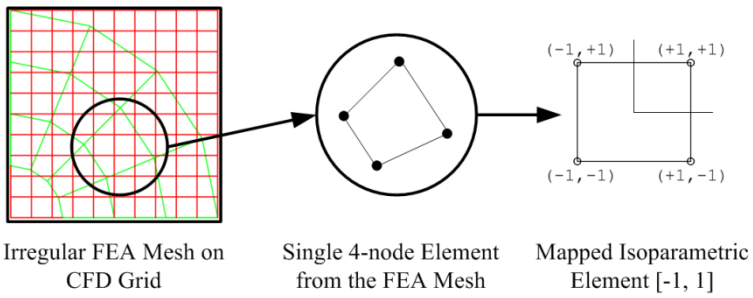


Figure 2-7: Mapping between real and natural coordinates

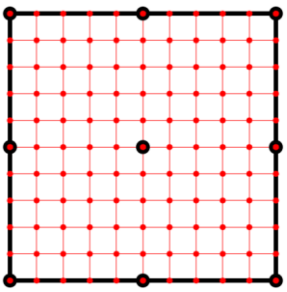


Figure 2-8: A fine mesh for measuring fluxes in the CFD domain superimposed on a single finite element with nine nodes in this case

For a rectangular element, heat fluxes are measured over a uniform grid in the CFD domain as shown in Fig. 2-8. Equation (2.6) is used to calculate the heat flux vector for the element. First, integration is performed along the straight lines of the CFD grid in one dimension, as shown in Fig. 2-9, resulting in $m + 1$ integrals along the first dimension (say, in the ξ-direction as before). The integral at a particular line in natural coordinates along the element's surface will be denoted as the integral j for $j = 0, 1, \ldots, m$.

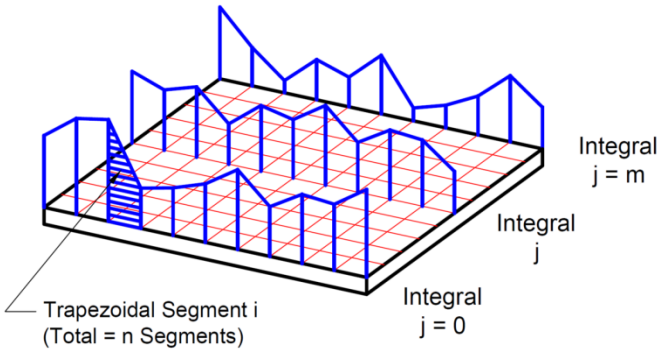


Figure 2-9: Integration over a surface using the trapezoid rule

From Eq. (2.6), it can be seen that each of the j integrals results in an array with the same size as the number of nodes in the element. The vector representing slice j is defined with a dummy variable $\{I\}^{(j)}$, where

$$\{I\}^{(j)} = \frac{1}{n}\left[\left(\lfloor N \rfloor^T q'' J\right)\Big|_{(\xi_0,\eta_j)} + 2\sum_{i=1}^{n-1}\left(\lfloor N \rfloor^T q'' J\right)\Big|_{(\xi_i,\eta_j)} + \left(\lfloor N \rfloor^T q'' J\right)\Big|_{(\xi_n,\eta_j)}\right] \qquad (2.7)$$

Note that $\{I\}^{(j)}$ is based on the 1D trapezoid rule for ξ ranging from -1 to 1 and $\eta = \eta_j$. In total, there are m vectors $\{I\}^{(j)}$ produced by integrating over the $j = 0, 1, \ldots, m$ slices of data. There are a total of $n + 1$ integration points in the ξ-direction.

The integration over the surface of the element is computed by applying the trapezoid rule to vectors $\{I\}^{(j)}$ for η ranging for -1 to 1, i.e.,

$$\{r\} \approx \frac{1}{m}\left(\{I\}^{(0)} + 2\sum_{j=1}^{m-1} \{I\}^{(j)} + \{I\}^{(m)} \right) \tag{2.8}$$

This integration method requires calculation of the element flux vector by the trapezoid rule using the number of data points available at the surface of each element. It is not required that the CFD grid matches the FEA grid, although compatibility between meshes facilitates the transfer of data between the two models. For the case in which the CFD grid does not match the FEA grid, integration points are defined on the element in isoparametric coordinates (e.g., trapezoid rules of order h, $h/2$, and $h/4$ are shown in Fig. 2-10). The heat fluxes from the CFD grid points that are adjacent to the integration point are interpolated linearly to get the heat flux data that is passed into the trapezoid rule, as shown in Fig. 2-10.

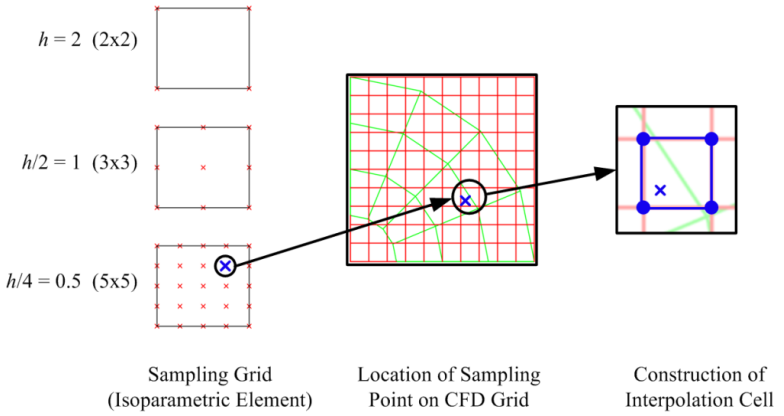


Figure 2-10: Relation between integration point sampling schemes and the relative mesh sizes in the CFD and FEA domains

2.3 Implementation

With an efficient numerical method for calculating energy-equivalent nodal fluxes from spatially varying incident heat flux data obtained by a CFD fire simulation, the non-uniform thermal boundary conditions produced in a natural fire scenario can be passed into a finite element heat transfer analysis model. The averaging, sampling, and least squares methods were compared to the trapezoid method for spatial homogenization in two dimensions. The trapezoid method was extended for a 3D case and implemented in a macro heat transfer element [16] that has a coarse grid in relation to the CFD grid. It should be noted that the homogenization technique for handling non-uniform thermal boundary conditions is particularly useful for macro-level finite elements, which have a coarser element mesh in relation to the CFD grid.

This chapter considers a sequentially coupled analysis in which the fire affects heat transfer to the structure but the structure does not affect the fire dynamics. A flow chart is provided in Fig. 2-11, which shows the transfer of data through the analysis. In Fig. 2-11, the disparities in spatial scale between the fire and solid domains are handled using a homogenization algorithm based on sampling, averaging, least squares, or trapezoid methods. To conduct the analysis, a CFD simulation is performed to compute the incident heat flux acting on the surfaces of the structure. The homogenization algorithm transforms large sets of data describing boundary conditions at a structure's surface into equivalent heat fluxes that act at the nodes of the finite element in the conduction heat transfer analysis. An input file is generated for the finite element heat transfer analysis, with the equivalent nodal heat fluxes specified as boundary conditions. The heat transfer model is then analyzed to determine the temperatures within the solid. For a two-way coupled system, surface fluxes and temperatures computed in the heat transfer analysis would be calculated at the CFD grid points by interpolation using the element's shape functions. The methodology

presumes that the CFD grid is finer than the element mesh used in the solid heat transfer model, which is generally the case when macro heat transfer elements are used.

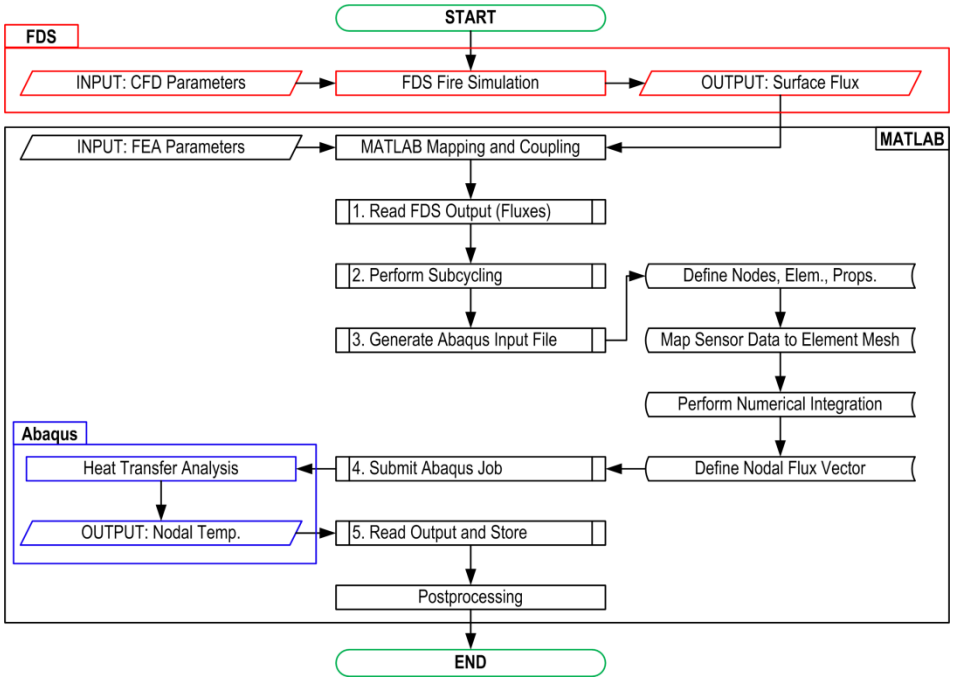


Figure 2-11: Flowchart for sequentially coupled analysis

2.3.1 2D Implementation

A 2D application was considered for the evaluation of the homogenization algorithms based on averaging, sampling, least squares, and the trapezoid rule. In the 2D application, the conduction heat transfer to the solid was evaluated using eight-node (quadratic) solid elements with the equivalent nodal fluxes calculated by Eq. (2.4) for the averaging, sampling, and least squares methods, or Eq. (2.5) for the trapezoid method. On the heated surface, the boundary condition was expressed as a net heat flux, i.e.,

$$q'' = q''_{net} \tag{2.9}$$

which was assumed to be constant in time so as to avoid challenges associated with differences in time scale that would otherwise occur in a coupled fire-structure simulation. To simplify the problem, a random heat flux was generated in MATLAB as opposed to performing a CFD fire simulation for the 2D analysis. It was also assumed that the heat flux variation out of plane was negligible such that the problem could be treated as 2D. The 2D heat transfer analysis was performed in a special-purpose code that was written in MATLAB.

2.3.2 3D Implementation

As will be shown in the following sections, the trapezoid method was found to exhibit superior performance over the averaging, sampling, and least squares methods. The algorithm based on the trapezoid rule was therefore extended to 3D heat transfer in Section 2.2 and implemented with the layered plate heat transfer element in Fig. 2-2b [16] for an application involving a plate exposed to a localized fire. In the plate heat transfer element, a heat flux boundary condition is applied to a layer at the top or bottom of the plate using Eq. (2.2). For homogenization of a spatially varying heat flux, the trapezoid rule is applied according to Eqs. (2.7-2.8). The present analysis considers a nine-node quadratic plate element, although the formulation in Section 2.2 is general and can be applied to any type and order of element. In the 3D case, the spatially varying heat flux was calculated by CFD analysis in Fire Dynamics Simulator (FDS). The boundary conditions are expressed in terms of an incident heat flux with losses to the surroundings by convection and radiation.

The 3D conduction heat transfer analysis was performed in Abaqus. A special purpose code was written in MATLAB to resolve the spatially varying surface fluxes into equivalent nodal fluxes according to the spatial homogenization algorithm in Eqs. (2.7-2.8). The plate heat transfer

element was implemented in Abaqus as a user-defined element (i.e., UEL subroutine). The equivalent nodal fluxes were specified using the *DFLUX command in the input file after they were calculated in a preprocessing program in MATLAB (see flowchart in Fig. 2-11).

2.4 Results

2.4.1 2D Application

The homogenization algorithms based on averaging, sampling, least squares, and the trapezoid rule were evaluated by considering a plate exposed to a random (i.e., non-uniform) net heat flux along one edge, as shown in Fig. 2-12. The average heat flux varied linearly from 10 kW/m^2 at the ends of the plate to 40 kW/m^2 at the center of the plate. About the mean, the heat flux followed a random distribution that was bounded by a range of 20 kW/m^2. The random heat flux can be seen in Fig. 2-12 along with the boundaries. The heat flux was assumed to be constant over the thickness of the plate (i.e., out of plane) and the plate was insulated out of plane such that the heating was restricted to two dimensions.

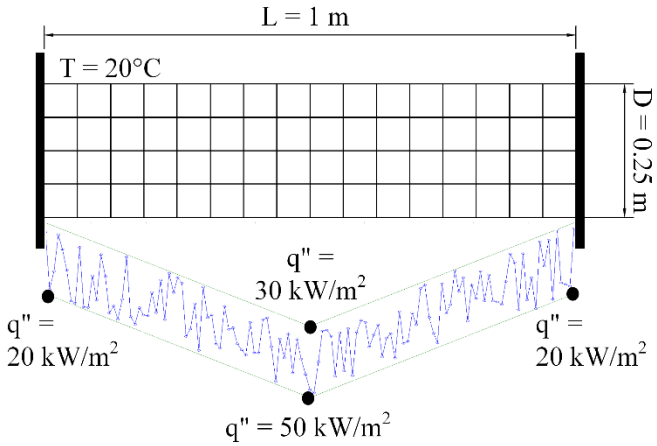


Figure 2-12: 2D application involving a solid exposed to a random heat flux

The plate was 1.0 m long and 0.25 m deep with a uniform thickness of 0.1 m. The plate was made of steel (specific heat = 465 J/kg·K, density = 7,850 kg/m^3, thermal conductivity = 54

W/m·K), and it was assumed that the material properties were independent of temperature for simplicity. The plate had an initial temperature of 20°C. At the start of the analysis, the bottom edge was instantaneously exposed to the non-uniform net heat flux, while a convective cooling condition was prescribed on the top surface with a heat transfer coefficient of 35 $W/m^2 \cdot K$ and fluid temperature of 20°C. It was assumed that the bottom surface was exposed to the non-uniform heat flux and did not interact with the surroundings (i.e., there were no losses due to convection or radiation). A duration of 2,000 seconds was chosen for the analysis.

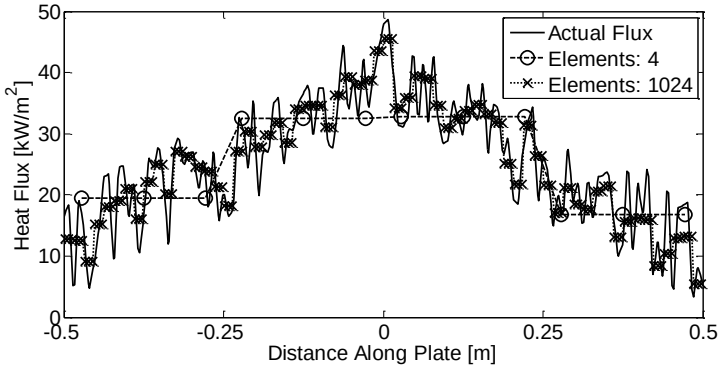
The analysis was completed using several different mesh configurations to evaluate the accuracy of each method as the mesh was refined. A summary of the mesh configurations is provided in Table 2-1. The "exact" solution was obtained using a very fine mesh that provided a one-to-one correlation between the heat flux data and the finite element mesh. The results from the exact solution were used to calculate errors in the temperatures associated with each of the homogenization algorithms.

Table 2-1: Mesh properties for the 2D case

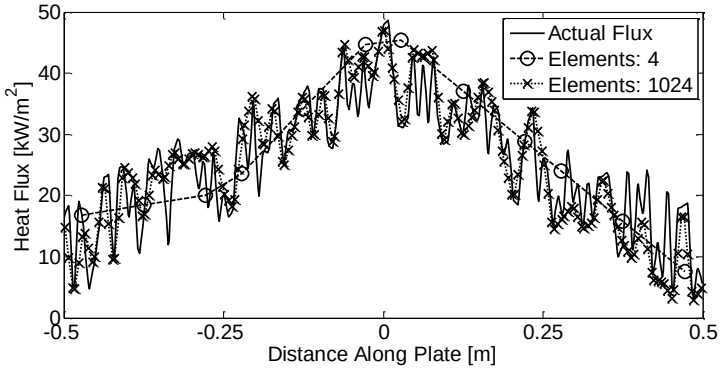
	Number of Elements		
Element Size [m]	**Vertical**	**Horizontal**	**Total**
0.25	1	4	4
0.125	2	8	16
0.0625	4	16	64
0.03125	8	32	256
0.015625	16	64	1024

The heat flux at the integration points for the averaging, sampling, least squares, and trapezoid methods is plotted in Fig. 2-13 for mesh configurations of 4 and 1024 elements. The actual heat flux is also plotted for comparison and is represented with a heavier line-weight. It can be seen that the averaging method (Fig. 2-13a) assumes a uniform heat flux based the average heat flux along the element's length, whereas the sampling (Fig. 2-13b) and least-squares methods (Fig.

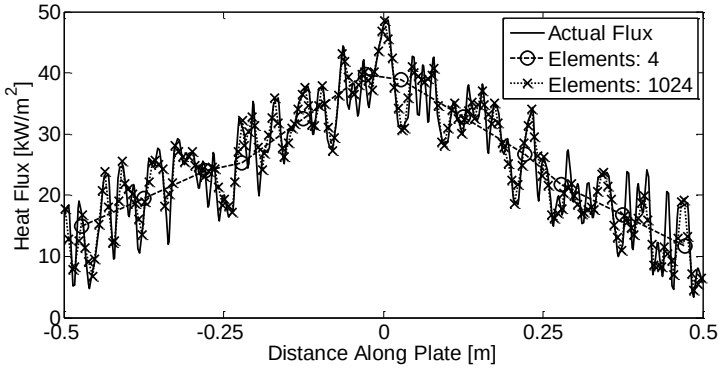
2-13c) fit the data with a linear function. The trapezoid rule method (Fig. 2-13d) uses the exact heat flux data in the calculation of the equivalent nodal heat flux vector.



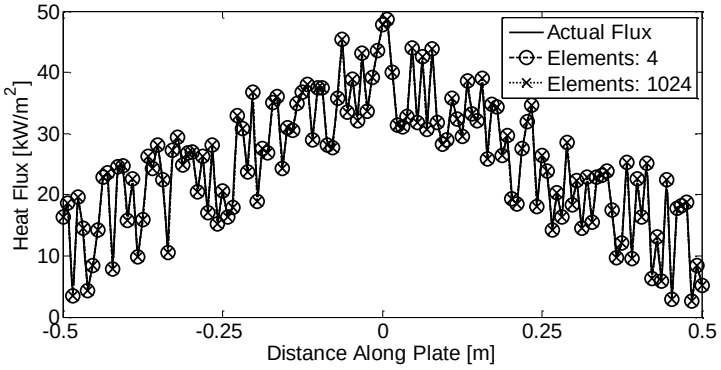
(a)



(b)



(c)



(d)

Figure 2-13: Heat flux at the integration points: (a) averaging method, (b) sampling method, (c) least squares method, and (d) trapezoid rule

The approximated heat fluxes shown in Fig. 2-13 were used in the heat transfer finite element analysis based on Eq. (2.4) or Eq. (2.5). The convergence of the solution was investigated based on the mesh configurations given in Table 2-1. The nodal temperatures at the heated surface of the plate were calculated for each method and compared to the "exact" (i.e., converged) finite element solution. The "exact" solution was defined as the solution obtained with an element size that corresponded to the spacing of flux data points (i.e., 4,096 elements arranged in a 32×128 grid). Comparisons between the coarsest and finest element meshes are shown in Fig. 2-14 for each of the methods. Note that the nodal temperatures are shown to be connected by straight lines, although the temperatures are actually interpolated using the quadratic shape functions for the 8-node element. From Fig. 2-14, it can be seen that least-squares (Fig. 2-14c) and trapezoid rule (Fig. 2-14d) methods provide a high degree of accuracy for both the coarse and fine finite element meshes, although the least-squares method results in a noticeable difference in temperature prediction at the ends of the plate for the fine mesh. The averaging method gives reasonable results for the fine mesh, although some variations in the calculated temperatures can be seen at the end of the plate (Fig. 2-14a). The sampling method produces significant differences in the calculated temperatures when compared to the expected values for both the coarse and fine meshes, indicating poor convergence (Fig. 2-14b).

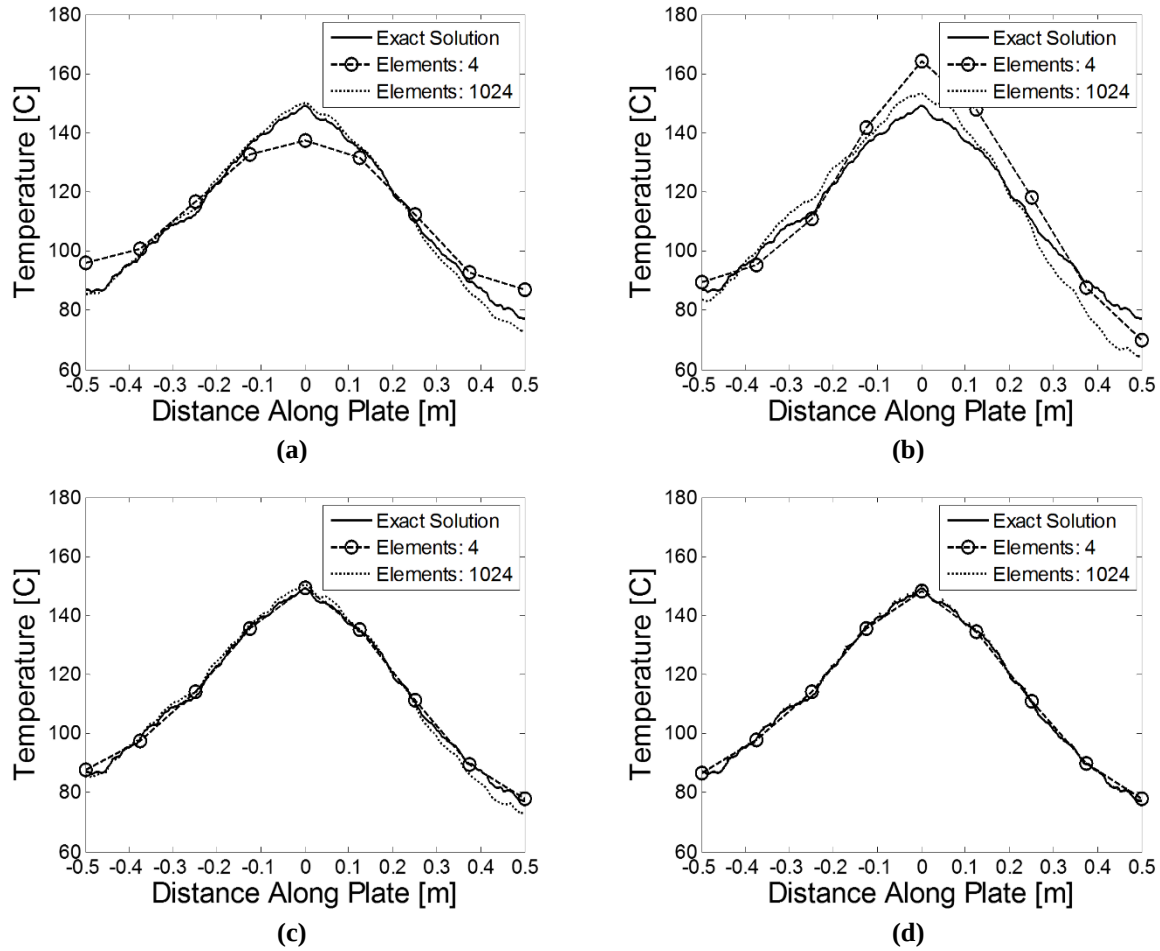


Figure 2-14: Heat flux at the integration points: (a) averaging method, (b) sampling method, (c) least squares method, (d) trapezoid rule method

To better gauge the accuracy of the simulation, relative errors were calculated based on the l^2-norm of the difference in temperature between the coarser element mesh employing the trapezoid rule and the converged finite element solution, as shown in Table 2-2. The data used for the comparison include all of the nodal temperatures along the heated edge of the plate. In general, the methods tend to converge to the exact solution as the number of elements increases. However, a consistent rate of convergence (i.e., consistently decreasing differences in temperatures between the reference solution and the increasingly fine mesh density) is not attainable with the sampling methods and, to some extent, the least squares methods. The increasing value of the norm with mesh refinement that is observed for the least-squares method is caused by a situation in which

too few data points exist over an element. The simulation time for each mesh size was comparable regardless of chosen approximation method. Overall, the trapezoid method leads to the smallest relative norm values in all cases and exhibits consistent convergence; it is therefore recommended for spatial homogenization.

Table 2-2: Vector norm for relative differences in temperature

	Number of elements				
Method	**4**	**16**	**64**	**256**	**1024**
Averaging	4.36	1.37	1.07	1.01	1.77
Sampling	7.14	6.75	9.52	4.73	5.32
Least squares	1.04	0.82	1.20	0.99	1.78
Trapezoid rule	0.85	0.59	0.41	0.24	0.26

2.4.2 3D Application

To evaluate the performance of the spatial homogenization algorithm based on the trapezoid rule, a 3D application was considered involving a horizontal plate subjected to a localized fire. The plate measured 2 × 1 m with a thickness of 5 cm and had constant thermal material properties, with a specific heat of 1,000 J/kg·K, density of 2,000 kg/m^3, and thermal conductivity of 2 W/m·K. The plate was located 1 m above a heptane pool fire, which was characterized by a peak heat release rate of 500 kW within a fixed burn area of 1,600 cm^2. The fire was modeled in FDS with a grid size of 5 cm for a total duration of 15 minutes. The FDS fire simulation displayed in Fig. 2-15 features sensors placed at 5-cm intervals along the top and bottom surfaces of the plate to measure the incident heat flux at each surface, resulting in 800 sensors on each of the top and bottom surfaces of the plate structure. Convection and radiation losses to ambient (T_∞ = 20 °C) were modeled, with a heat transfer coefficient of 25 W/m^2·K and an emissivity of 0.8.

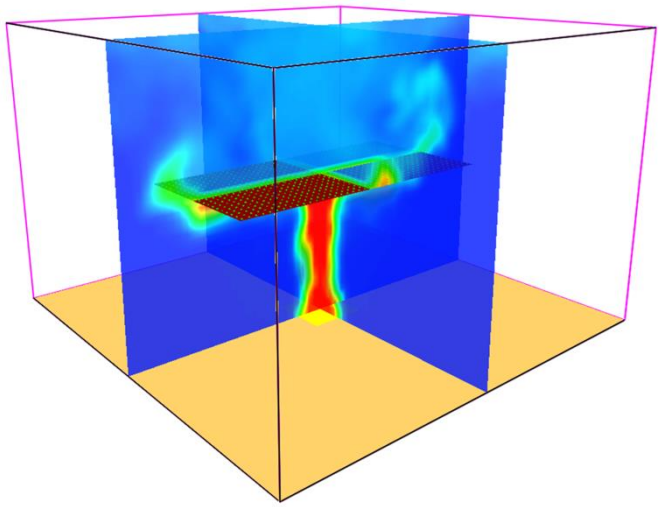
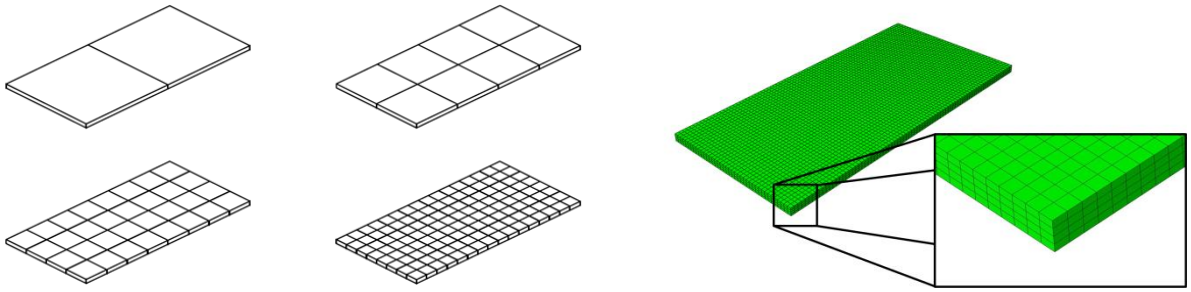


Figure 2-15: Fire simulation in FDS

The homogenization algorithm was used in conjunction with the 4-node linear shell heat transfer element. A convergence study was performed to measure the performance of the shell element in conjunction with the homogenization algorithm. Four mesh configurations were considered for the plate model: 2×1, 4×2, 8×4, and 16×8, as shown in Fig. 2-16a. The shell element used five equally spaced layers over the thickness. For comparison, a solid element model was generated in Abaqus using eight-node (linear) brick elements (i.e., DC3D8 elements), as shown in Fig. 2-16b. Four elements were required through the thickness in order to calculate the temperature gradient through the thickness. An element size of 1.25 cm × 2.5 cm × 2.5 cm was chosen in order to preserve an appropriate aspect ratio. Thus, the mesh for the solid element model contained 40 elements across the width of the plate, 80 elements along the length of the plate, and 4 elements through the thickness of the plate, for a total of 12,800 elements and 16,605 temperature degrees of freedom. An overview of the mesh details for each of the models is given in Table 2-3.



(a) (b)

Figure 2-16: Mesh configurations used in the test: (a) shell element models of 2×1, 4×2, 8×4, and 16×8 elements, each containing five layers through the thickness (not depicted); (b) solid element model with four elements through the thickness

Table 2-3: Mesh properties for the 3D case

Element Type	Mesh	DOF	Edge Size [cm]
Shell UEL	2 × 1	30	100
Shell UEL	4 × 2	75	50
Shell UEL	8 × 4	225	25
Shell UEL	16 × 8	765	12.5
Solid DC3D8	80 × 40	16,605	2.5

Surface flux data from the CFD fire simulation in FDS was written to the output file in one-second intervals. Rather than limiting the time step in the solid heat transfer model to one second, the time-averaged subcycling algorithm [4] was used with a time step of 10 seconds to increase the efficiency of the analysis. Subcycling was performed in MATLAB prior to the generation of the input files for the solid heat transfer models. The time-averaged heat flux measured at the sensor in the center of the plate is compared to the actual sensor data from FDS in Fig. 2-17. After subcycling, the homogenization algorithm based on the trapezoid rule was applied to calculate the equivalent nodal heat fluxes in the shell element models. Homogenization was not needed for the solid element model due to the fact that the mesh in the solid heat transfer model was finer than the CFD mesh.

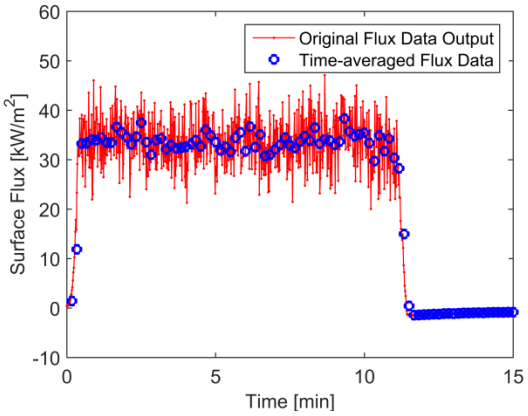
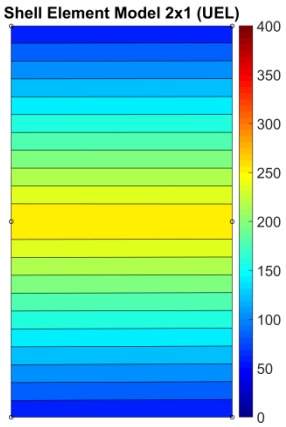
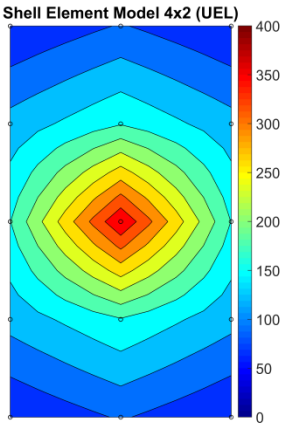


Figure 2-17: Incident heat flux over time for a sensor at the center of the plate

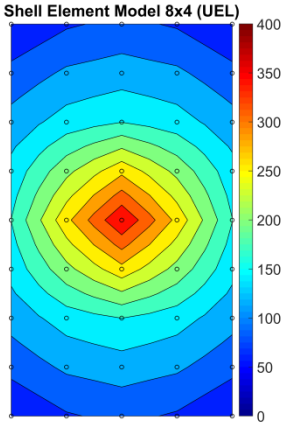
Accuracy was measured by comparing the shell models to the solid element model. The spatial homogenization algorithm performed well when used with the shell heat transfer elements. Contour plots of the mid-surface temperatures at 12 minutes into the simulation are shown in Fig. 2-18. Temperatures calculated by the shell models with spatial homogenization are shown in Fig. 2-18 (a-d) while the temperatures calculated by the solid element model are shown in Fig. 2-18e. It can be seen that the shell model converges to the solution calculated by the solid element model. Temperatures are also plotted through the thickness to demonstrate the accuracy of the shell model in predicting cross-sectional temperatures. As illustrated in Fig. 2-19, a slice through the plate's thickness was taken at the middle of the plate to illustrate the temperature gradients through the thickness. Temperatures are plotted for the solid element model (Fig. 2-20c) and for the coarsest (2×1) and finest (16×8) shell models (Figs. 2-20a and 2-20b, respectively). Temperature contours are shown at 6 and 12 minutes into the simulation. It can be seen that the shell model converges to the solid element model as the number of elements is increased.



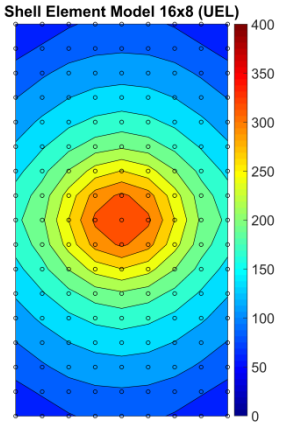
(a)



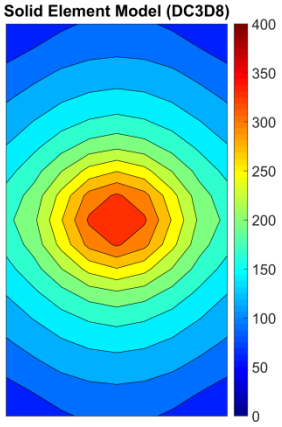
(b)



(c)



(d)



(e)

Figure 2-18: Contours of the temperature field at the mid-surface of the plate after 12 minutes of fire exposure (a) 2×1 shell model, (b) 4×2 shell model, (c) 8×4 shell model, and (d) 16×8 shell model, and (e) 80×40 solid element model

A measure of the relative difference in the computed temperatures at the mid-surface of the plate is provided in the form of the relative l^2-norm values, as given in Table 2-4. The reference solution for computing the norm values was based on the converged solid element model. The relative l^2-norm values were calculated by interpolating between nodal temperatures in the shell models to retrieve the temperatures in the locations of nodes in the solid element model at the mid-surface of the plate. The comparison in Table 2-4 was prepared using temperatures at the mid-surface of the plate only. However, similar convergence properties were exhibited at the top and bottom surfaces of the plate as well.

Table 2-4: Comparison between the shell model and the solid element model

Model Details			Percent Difference (Relative Norm) [%]				Computing Time
Element	**Mesh**	**DOF**	**3 min**	**6 min**	**9 min**	**12 min**	
Shell UEL	2 × 1	30	15.6	20.2	20.3	19.7	11.5 sec
Shell UEL	4 × 2	75	3.2	4.0	4.0	3.8	13.8 sec
Shell UEL	8 × 4	225	3.0	3.9	3.8	3.6	20.1 sec
Shell UEL	16 × 8	765	1.2	1.5	1.5	1.4	47.4 sec
DC3D8	80 × 40	16,605	--	--	--	--	48.5 min

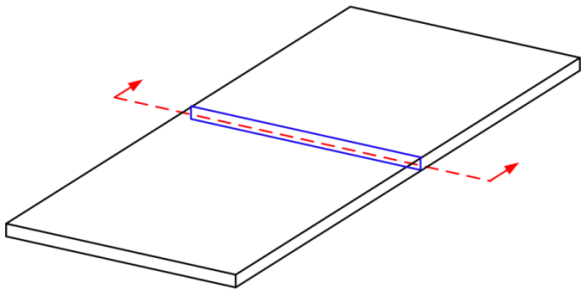
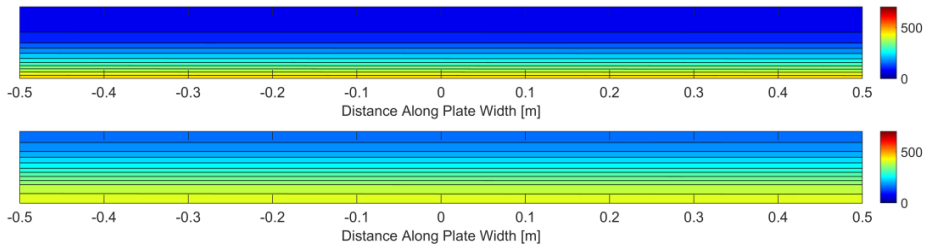
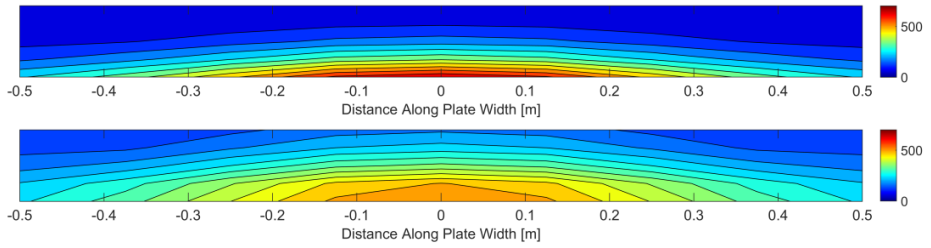


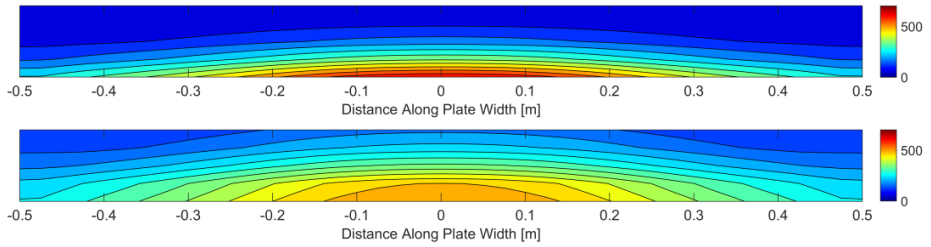
Figure 2-19: Section for contour plots taken through the thickness



(a)



(b)



(c)

Figure 2-20: Temperature field through the thickness at 6 min (top) and 12 min (bottom): (a) coarse shell mesh of 2×1, (b) fine shell mesh of 16×8, and (c) the solid element model

The computing times are also reported in Table 2-4. The computing time shown in Table 2-4 includes the subcycling process, the mapping of fluxes to the element surfaces, the computation of equivalent nodal fluxes, and the actual heat transfer analysis. The subcycling process required a mere 0.08 seconds for each model. The time required for the other components of the simulation was directly dependent on the number of degrees of freedom and the numerical integration scheme. As shown in Table 2-4, the 4×2 shell model employing homogenization by the trapezoid rule was within 4 percent of the solid element model and required only 13.8 seconds to

complete the analysis. The 16×8 shell model provided temperatures that were within 1.5 percent of the solid element model and required 47.4 seconds. For comparison, the solid element model that was used as the reference solution required a total of 48.5 minutes and required significantly more degrees of freedom.

2.5 Conclusions

A spatial homogenization algorithm was formulated for capturing non-uniform thermal boundary conditions associated with a CFD fire simulation. Energy-equivalent nodal fluxes were calculated for use with macro-level finite elements for heat transfer analysis, which have a coarser mesh in relation to the CFD grid. The method for calculating the nodal fluxes is based on the trapezoid rule for numerical integration. The proposed method was compared to other homogenization techniques including sampling, averaging, and least squares methods for a 2D application. The trapezoid rule approach offers superior performance because it more closely enforces conservation of energy by accounting for the variations in heat flux over the surface of the structure.

The homogenization algorithm was extended to 3D analyses and implemented in a macro heat transfer element based on a shell formulation. The homogenization algorithm combined with the shell heat transfer element resulted in an extremely efficient and accurate solution that led to considerable time savings when compared to a solid-element model. Relative errors of less than 1.5% were reached using 128 layered shell elements in a 16×8 configuration, requiring less than one minute (47.4 seconds) of computing time as opposed to the 48.5 minutes that were needed to complete the solid-element model. This chapter does not consider the cost of the CFD fire simulation, which still requires significant computational resources in a coupled fire-structure simulation. Additionally, the work described herein only considered a flat rectangular plate in the

3D application and additional work is needed to extend the methodology to tilted and curved geometries.

The applications considered here involved unprotected structures subjected to localized fires with impinging flames. The situation is intended to represent a worst-case scenario in which the structure is highly sensitive to variations in surface fluxes and in which the surface fluxes vary considerably over small distances. It is acknowledged that protected structures will be less sensitive to variations in surface fluxes. Additionally, members heated by remote radiation and by optically thick gases may experience less severe fluctuations in surface heat fluxes and therefore may be suitably modeled by other means that assume uniform temperature. In this chapter, we advocate for an algorithmically consistent manner for representing the thermal boundary conditions, as achieved by the homogenization algorithm presented here.

2.6 Acknowledgements

This work was supported by the United States Office of Naval Research under contract number N00014-13-C-0373. Any opinions, findings, conclusions, or recommendations are those of the authors and do not necessarily reflect the views of the sponsoring agency.

2.7 References

[1] Eurocode 1: Actions: General actions – Actions on structures exposed to fire, BS EN 1991-1-2, British Standards Institution, London, 2009.
[2] J. Stern-Gottfried, G. Rein, Travelling fires for structural design - Part I: Literature review, Fire Safety J. 54 (2012) 74-85.
[3] K. McGrattan, H. Baum, W. Mell, R. McDermott, Fire Dynamics Simulator (Version 5) Technical Reference Guide – Volume 1: Mathematical Model, NIST Special Publication 1018-5, National Institute of Standards and Technology, Gaithersburg, MD, 2010.
[4] X. Yu, A.E. Jeffers, A comparison of subcycling algorithms for bridging disparities in temporal scale between the fire and solid domains, Fire Safety J. 59 (2013) 55-61.

[5] K. Prasad, H. Baum, Federal building and fire safety investigation of the World Trade Center Disaster—Fire structure interface and thermal response of World Trade Center, NIST NCSTAR 1-5G, National Institute of Standards and Technology, Gaithersburg, MD, 2005.
[6] N. Tondini, O. Vassart, J.M. Franssen, Development of an interface between CFD and FE software, in: Fontana, M., Frangi, A., and Knobloch, M. (Eds.), Proceedings of the 7th International Conference on Structures in Fire, ETH Zurich, Zurich, 2012.
[7] N. Tondini, A. Morbioli, O. Vassart, S. Lechene, J.M. Franssen, An integrated modelling strategy between FDS and SAFIR: the analysis of the fire performance of a composite steel-concrete open car park, in: G.Q. Li et al. (Eds.), Proceedings of the 8th International Conference on Structures in Fire, Shanghai, China, 2014.
[8] U. Wickström, D. Duthinh, K. McGrattan, Adiabatic surface temperature for calculating heat transfer to fire exposed structures, Proceedings of the 11th Interflam Fire Science and Engineering Conference, Interscience Communications, London, 2007.
[9] C. Culver, Steel column buckling under thermal gradients, J. Struct. Div. 92 (1972) 1853-1865.
[10] P. Ossenbruggen, V. Aggarwal, C. Culver, Steel column failure under thermal gradients by member, J. Struct. Div. 99 (1973) 727-739.
[11] J. Kruppa, Some results on the fire behavior of external steel columns, Fire Saf. J. 4 (1981) 247-257.
[12] J. Witteveen, L. Twilt, A critical view on the results of standard fire resistance tests on steel columns, Fire Saf. J. 4 (1981) 259-270.
[13] C. Zhang, G.-Q. Li, A. Usmani, Simulating the behavior of restrained steel beams to flame impingement from localized-fires, J. Constr. Steel Res. 83 (2013) 156-165.
[14] L. Chen, C. Luo, J. Lua, FDS and Abaqus coupling toolkit for fire simulation and thermal and mass flow prediction, in: M. Spearpoint (ed.), Proceedings of the IAFSS 10th International Symposium on Fire Safety Science, International Association for Fire Safety Science, UK, 2011.
[15] A.E. Jeffers, E.D. Sotelino, An efficient fiber element approach for the thermo-structural simulation of non-uniformly heated frames, Fire Safety J. 51 (2012) 18-26.
[16] A.E. Jeffers, Heat transfer element for modeling the thermal response of non-uniformly heated plates, Finite Elements in Analysis and Design 63 (2013) 62-68.
[17] A.E. Jeffers, P.A. Beata, Generalized shell heat transfer element for modeling the thermal responses of non-uniformly heated structures, Finite Element in Analysis and Design 83 (2014) 58-67.
[18] A.E. Jeffers, Triangular shell heat transfer element for the thermal analysis of non-uniformly heat structures, J. Struct. Eng. (2015), to appear.
[19] R.D. Cook, D.S. Malkus, M.E. Plesha, R.J. Witt, Concepts and Applications of Finite Element Analysis, 4th ed, John Wiley and Sons, U.S., 2002.

Chapter 3
Thermo-mechanical Shell Element for Coupled Fire-structure Analysis

One modern approach to understanding the fire-structure interaction problem involves using a computational fluid dynamics (CFD) simulation to model the natural fire evolution within a structure. Using the output from the CFD-based fire simulation, one can then sequentially couple the fire boundary conditions to a thermal model of the structure for subsequent heat transfer analysis. A problem arises when the analyst needs to model the deformation in the structure due to the thermal exposure, which would require re-meshing in the finite element analysis (FEA), thus implying the need for another coupling between the thermal and mechanical FEA models. However, employing thermo-mechanical shell elements in the coupled fire-structure simulation problem allows for this multi-step analysis in a single FEA model without re-meshing thus solving for temperatures and displacements simultaneously. The current study demonstrates the accuracy and convenience of this approach for calculating deformations in the structure due to non-uniform heating from a simulated fire source. Performing the thermo-mechanical simulation in a single step after the completion of the fire simulation removes one phase of data transfer from the workflow, allowing for a streamlined two-step method using heat fluxes to compute temperatures and displacements in the structure based on measured heat fluxes from the CFD fire simulation.

3.1 Introduction

The aim of fire-structure simulation is to characterize the temperature and displacement fields within the fire-exposed structure in an accurate and efficient manner. For thin-walled

structures, such as floor slabs in buildings or structural panels in naval vessels, finite-element approaches based on shell theory are the preferred means for analyzing the deformation response. The benefit of shell elements from a computational perspective is the reduction of the degrees of freedom (DOF) in structures with larger planar dimensions than the thickness dimension. Shells overcome the computational restrictions posed by 3D continuum-based elements, which may demand an excessively fine mesh of elements to capture bending behavior and maintain acceptable aspect ratios for the elements. This benefit is well known for the use of general shell elements, and a similar situation has been encountered in the 3D conduction heat transfer in thin-walled structures. In the fire-structure simulation problem, there are three key components requiring the selection of a set of analysis tools. Figure 3-1 presents a high-level view of the coupled fire-structure interaction problem using data from a CFD-based fire simulation.

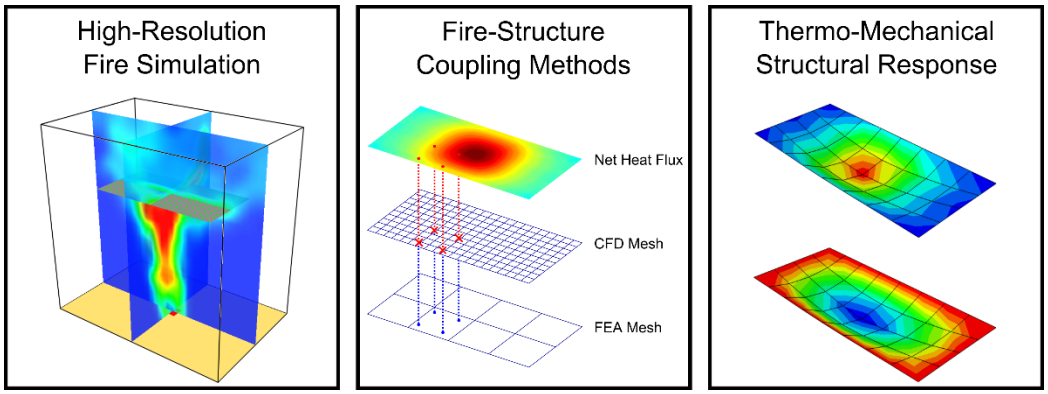


Figure 3-1: Overview of the sequentially coupled fire-structure interaction problem using CFD-to-FEA coupling methods

The three main stages of analysis are shown in Fig. 3-1, i.e., (1) the high-resolution fire simulation using CFD software, (2) the fire-to-solid coupling for mapping the fire boundary conditions for non-matching meshes in the fire and solid domains, and (3) the thermo-mechanical response calculation using FEA. The simulation of fires using CFD is well-established as an accurate representation of natural fires. For example, the CFD-based fire simulation software is

Fire Dynamics Simulator (FDS), which was developed at the National Institute of Standards and Technology (NIST) [1], has been extensively validated against a wide range of fire experiments. However, limited work has been done on the coupling of the CFD fire simulation to the thermo-mechanical simulation of structural shells, particularly with regards to the two phases of Fig. 3-1 beyond the fire simulation. The primary focus of the current study is capturing the thermo-mechanical response in the structure using coupled thermo-mechanical shell elements (the image on the right in Fig. 3-1). The work builds on prior research by the authors regarding fire-structure coupling methods [2,3] and the thermal analysis of shells [4–6].

Research in the field of general shell theory is expansive but the thermal shell elements available in the literature are limited to a few researchers. For example, the work by Surana's team [7–12] has produced a series of foundational papers on topics ranging from axisymmetric thermal shells to three-dimensional hierarchical shells. One common theme among that line of work was to represent the through-thickness temperature field using a temperature DOF and its gradient computed at the mid-surface of the shell as a secondary DOF. Noor and Burton [13] gave a predictor-corrector method for the calculation of heat fluxes and temperature distributions in thick multilayered composite shells and plates. Mukherjee and Sinha [14] presented a formulation based on a Galerkin approach applied to the heat conduction problem in laminated composite plates as well. Noack, Rolfes, and Tessmer [15] formulated a layer-wise theory for heat conduction of hybrid structures and sandwich panels. The authors [4–6] formulated heat transfer shell elements for 3D transient conduction analysis in thin-walled structures. The approach provides a layered method for computing the temperature field through the thickness of the shell while providing bilinear or biquadratic temperature distributions in the plane of the element. Temperature DOF are

stacked at each node such that a single *n*-node thermal shell element with *NL* temperature layers would have $n \cdot NL$ total temperature DOF in the element.

In the present work, a fully coupled thermo-mechanical shell element was developed to simultaneously solve for the temperature and displacement fields using the same finite-element mesh (i.e., made up of thermo-mechanical shell elements) to model the coupled response of thin-walled structures. The element matrices and vectors associated with conduction heat transfer in the solid were computed using the previous formulation [4,5]. For the mechanical shell element formulation, a general-purpose shell was selected to couple with the thermal shell element. The theory presented by Kanok-Nukulchai [16] for a degenerated bilinear shell element was adapted here and used in conjunction with the thermal shell element; together, these formulations provided the basis for coupled temperature-displacement analyses. This formulation was selected for its specific use of a torsional stiffness term to resist rotations about the normal vector to the shell surface. Other recent approaches [17,18] have also addressed the drilling rotation problem using this method [16].

The thermal shell element presented by the authors [4,5] and the classical mechanical shell element presented by Kanok-Nukulchai [16] were combined using a standard virtual work approach [19] for finite-element coupled thermoelasticity. The process for formulating the coupled-element system of equations was used to develop non-symmetric element arrays of the coupled thermoelastic solid by starting from a temperature-dependent constitutive law. In this approach, layer-wise temperatures were considered DOF on the left-hand side of the system of equations as opposed to thermal strain forcing terms on the right-hand side. The coupled thermo-mechanical shell element was subsequently implemented as a user-defined element in Abaqus through the UEL subroutine [20]. Numerical examples employing the thermo-mechanical shell

element are provided to demonstrate the performance of the formulation as implemented in Abaqus. The user subroutine UEL was used to include this element in the software.

3.2 Coupled thermo-mechanical shell element

In the current study, two finite-element theories for shell elements were coupled to provide a thermo-mechanical shell element formulation for use in thermoelastic analyses. The formulation of this fully coupled shell element was intended for 3D transient temperature-displacement analyses and was implemented as a user element in Abaqus through the UEL subroutine, which was specifically designed for future use in one-way coupled fire-structure interaction problems. The main formulation for each of the individual shell theories used here has been published in previous work. Specifically, the *thermal* shell comes from a layered formulation for 3D conduction heat transfer analysis [5], while the original *mechanical* shell element is based on an equivalent single layer theory in which the element properties and arrays are computed only at the mid-surface of the element [16]. The main features of these original formulations will be discussed in the following subsections.

In the coupled formulation presented here, the main consideration for linking the two theories was properly handling the various temperature layers through the thickness from the thermal shell model. The coupled formulation was based on the principle of virtual work and uses descriptions of the finite-element approximation of the shell geometry and DOF from each individual formulation directly. The fully coupled shell element employs a bilinear displacement field approximation in the mid-surface plane and a bilinear temperature field within the plane of each thermal layer. The temperature field through the thickness is incorporated in the coupling stiffness terms by integrating in this direction with a piecewise-quadratic polynomial

approximation using Simpson's quadrature rule. As a result, only four nodes need to be defined by the user at the corners of the mid-surface for each element and the calculation of nodal direction vectors was handled automatically in the Fortran implementation (i.e., UEL subroutine).

The virtual work approach to generate the coupled-element matrices was essential for solving for the temperatures and displacements simultaneously in the shell (i.e., in the same time iteration of the analysis) using the non-symmetric matrix solver in Abaqus. The result was a fully coupled thermo-mechanical shell finite element capable of any (odd) number of temperature layers. In the fire-structure simulation setting, some researchers have used heat transfer elements to calculate temperatures and then mapped the temperature field to the structural shell elements (e.g., [21]). Others (including the authors) have explored this route of coupled thermo-mechanical shell formulations that capture the 3D conduction and deformation in the structure simultaneously.

The total number of DOF is dependent on the number of layers selected for the analysis; the default value of five layers is typical [20]. The transfer process to map nodal temperatures from the thermal model to mechanical model, as seen in other attempts, was eliminated with this solution to the coupled thermo-mechanical problem. Currently, the element is limited to small-displacement analysis and thermoelastic problems for this initial development.

3.2.1 Thermal shell element

The original formulation of the thermal plate provided a layered 3D finite element to model the conduction heat transfer in thin-walled structures [4]. The basis of the theory was the use of a collection of layers in the thickness direction comprising the full plate geometry in which the temperature variation of a given layer could be interpolated within the plane of that layer using 2D basis functions. Thermal plate element was extended to a more general formulation for 3D shells

[5] and later took into consideration triangular element geometry [6]. In general, the thermal shell theory in [4–6] research has proven to be efficient and accurate for modeling 3D thermal conduction in solid media.

The thermal shell element provides a finite-element approach for solving the 3D heat conduction equation within a solid body:

$$\frac{\partial}{\partial x}\left(k_x \frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k_y \frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(k_z \frac{\partial T}{\partial z}\right)+Q=\rho c \frac{\partial T}{\partial t} \tag{3.1}$$

Here, k_x, k_y, and k_z represent the thermal conductivity in each principal direction; Q is the internal volumetric heat source; ρ is density; c is the specific heat; and T is the unknown temperature field within the solid medium. Derivatives present in Eq. (3.1) are with respect to the global x, y, and z directions and with respect to time t. The element was discretized into layers (Fig. 3-2) in which the temperature in the thickness direction was lumped at each layer and the temperature in the plane was approximated as bilinear or biquadratic.

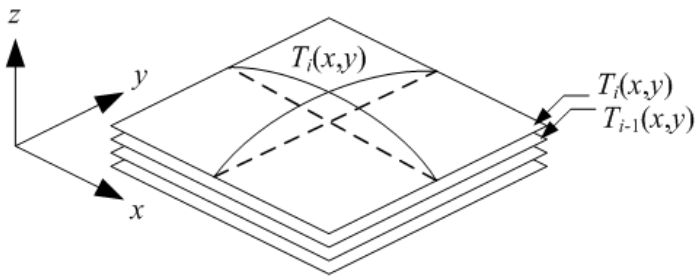


Figure 3-2: Layered thermal shell element, as originally presented in [5]

Using a combination of finite-element and control-volume approaches, the authors showed that the governing equations for the thermal shell element could be expressed in the form [5]:

$$\mathbf{C}_T \dot{\mathbf{U}}_T + \mathbf{K}_T \mathbf{U}_T = \mathbf{R}_T \tag{3.2}$$

Here, $\mathbf{C}_T$ is the specific heat matrix, $\mathbf{K}_T$ is the total conductivity matrix, $\mathbf{R}_T$ contains the thermal forcing terms, and the array $\mathbf{U}_T$ is a vector that contains all the temperature DOF in the system. For

the sake of brevity, the details of the formulation are not repeated here. The subscript *T* was included in Eq. (3.2) in order to distinguish these *thermal* terms from the mechanical terms in the coupled formulation that follows. Equations for $\mathbf{C}_T$, $\mathbf{K}_T$, and $\mathbf{R}_T$ are given in [5]. Readers should refer to the original publications for these details.

The definition of the shell geometry requires only specification of the nodal coordinates at the mid-surface of the shell, as shown for the typical node in Fig. 3-3. The normal direction vectors (i.e., the vectors normal to the mid-surface) were computed internally within the Fortran implementation through the use of a standard algorithm [22].

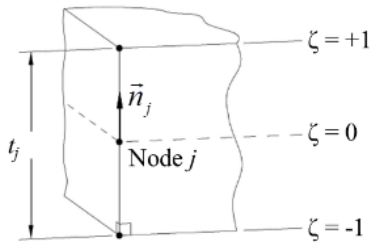


Figure 3-3: Corner node at the mid-surface of the shell element, as originally presented in [5]

3.2.2 Coupling approach

To discuss the mechanical shell element, a coupling approach is needed in order to define the foundation of the formulation. From the coupled formulation approach, the thermal and mechanical components will become clearly distinguishable. The coupled formulation begins with a typical statement of the principle of virtual work for a solid deformable body (or principle of virtual displacements). For brevity, the statement of virtual work is provided here for a deformable elastic body without explicit derivation:

$$\int \delta\boldsymbol{\varepsilon}^T \boldsymbol{\sigma}\, dV = \int \delta\mathbf{u}^T \mathbf{F}\, dV + \int \delta\mathbf{u}^T \boldsymbol{\Phi}\, dS \tag{3.3}$$

The expression given above involves both virtual displacements $\delta\mathbf{u}$ and virtual strains $\delta\boldsymbol{\varepsilon}$, signified by the δ present in each of the terms of the equality. The remaining components are the stresses $\boldsymbol{\sigma}$, the body forces $\mathbf{F}$, and the surface tractions $\boldsymbol{\Phi}$. The left-hand-side term is comprised of the single volume integral, which is integrated over the whole domain of the elastic body. The right-hand-side has one volume integral for the body-force terms as well as a surface integral for capturing the applied traction forces on the boundary surface of the solid. That surface is a particular subset of the full element domain on which a boundary condition, namely the traction forces $\boldsymbol{\Phi}$, has been applied.

The formulation continues with the linear-elastic constitutive law, which takes into consideration thermal strains, and is divided into membrane and shear terms (presented here in a local coordinate system at the individual element level):

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{Bmatrix} = \begin{bmatrix} D & \nu D & 0 \\ \nu D & D & 0 \\ 0 & 0 & G \end{bmatrix} \begin{Bmatrix} \varepsilon_{11} - \alpha\theta \\ \varepsilon_{22} - \alpha\theta \\ \varepsilon_{12} \end{Bmatrix} \quad \Rightarrow \quad \boldsymbol{\sigma}_m = \mathbf{D}_m(\boldsymbol{\varepsilon}_m - \boldsymbol{\varepsilon}_{th}) \tag{3.4}$$

$$\begin{Bmatrix} \sigma_{13} \\ \sigma_{23} \end{Bmatrix} = \begin{bmatrix} fG & 0 \\ 0 & fG \end{bmatrix} \begin{Bmatrix} \varepsilon_{13} \\ \varepsilon_{23} \end{Bmatrix} \quad \Rightarrow \quad \boldsymbol{\sigma}_s = \mathbf{D}_s\boldsymbol{\varepsilon}_s \tag{3.5}$$

These expressions can be written more concisely in matrix-vector form as follows:

$$\begin{Bmatrix} \boldsymbol{\sigma}_m \\ \boldsymbol{\sigma}_s \end{Bmatrix} = \begin{bmatrix} \mathbf{D}_m & 0 \\ 0 & \mathbf{D}_s \end{bmatrix} \begin{Bmatrix} (\boldsymbol{\varepsilon}_m - \boldsymbol{\varepsilon}_{th}) \\ \boldsymbol{\varepsilon}_s \end{Bmatrix} = \begin{Bmatrix} \mathbf{D}_m(\boldsymbol{\varepsilon}_m - \boldsymbol{\varepsilon}_{th}) \\ \mathbf{D}_s\boldsymbol{\varepsilon}_s \end{Bmatrix} \tag{3.6}$$

The σ_{33} stress term is restricted to zero, which is a common assumption in shell element formulations. The constant D can be expressed as $E / (1 - \nu)^2$ where E is the modulus of elasticity or Young's Modulus and ν is Poisson's Ratio in the above equations. G is the shear modulus; f is a shear correction factor of $5/6^{\text{th}}$ which is the typical value for such shell elements; α is the thermal expansion coefficient; and θ is the temperature difference, which is defined as $T - T_0$, where T_0 is

the reference temperature representing a material state with zero thermal stress. In the formulation that follows, all material properties were assumed to be constant and independent of temperature. The thermal strain $\alpha\theta$ is the main connection between the thermal and mechanical shell element formulations and is the basis of the thermo-mechanical coupled-shell approach. Notice that the strain terms associated with $\boldsymbol{\varepsilon}_{th}$ of Eq. (3.6) were not explicitly defined yet; these thermal strains can be expressed compactly as the following:

$$\boldsymbol{\varepsilon}_{th} = \begin{bmatrix} \alpha & \alpha & 0 \end{bmatrix}^T \theta = \mathbf{A}\theta \tag{3.7}$$

Recall θ is a scalar temperature difference at a particular location in the shell volume.

In finite element formulation, the typical strain-displacement matrix **B** is used to relate strains and nodal displacements, i.e., $\boldsymbol{\varepsilon} = \mathbf{BU}$. In the work by Kanok-Nukulchai [16], the strain-displacement relationship was conveniently written as follows:

$$\begin{Bmatrix} \boldsymbol{\varepsilon}_m \\ \boldsymbol{\varepsilon}_s \end{Bmatrix} = \sum_{I=1}^{n} \begin{bmatrix} \mathbf{B}_{1m}^I & (\mathbf{B}_{2m}^I + \zeta\, \mathbf{B}_{3m}^I) \\ \mathbf{B}_{1s}^I & (\mathbf{B}_{2s}^I + \zeta\, \mathbf{B}_{3s}^I) \end{bmatrix} \begin{Bmatrix} \mathbf{u}^I \\ \boldsymbol{\omega}^I \end{Bmatrix} \tag{3.8}$$

The index I represents the I^{th} node in the element and the displacement vector is broken up into $\mathbf{u}^I$ and $\boldsymbol{\omega}^I$, which are the global displacements and rotations, respectively. In accordance with the original formulation, the $\mathbf{B}_{2m}$ term is zero but shown here for completeness. The individual components of the strain-displacement matrix **B** were factored properly such that they only depend on the planar natural coordinates of ξ and η, while the ζ-dependency is explicitly visible in Eq. (3.8). Note that for the mechanical shell element, the array $\mathbf{u}^I$ contains nodal translations and $\boldsymbol{\omega}^I$ contains nodal rotations for the I^{th} node of the element. The two terms are related by the finite-element approximation for defining the displacement field within the element volume as follows:

$$\mathbf{U}(\xi,\eta,\zeta) = \sum_{I=1}^{n} N^I(\xi,\eta) \left\{ \mathbf{u}^I + \frac{1}{2} t\, \zeta\, \boldsymbol{\Phi}^I\, \boldsymbol{\omega}^I \right\} \tag{3.9}$$

The nodal basis functions are defined as each of the N^I terms while the nodal translations and rotations are shown within the summation ranging over all the nodes (totaling n) in the element. The additional factors on the rotations are the element thickness t, the thru-thickness natural coordinate ζ, and a nodal coordinate-transformation matrix $\mathbf{\Phi}^I$.

Following Eq. (3.9), virtual deformations can be similarly defined:

$$\delta\mathbf{U}(\xi,\eta,\zeta) = \sum_{I=1}^{n} N^I(\xi,\eta)\left\{\delta\mathbf{u}^I + \frac{1}{2}t\,\zeta\,\mathbf{\Phi}^I\,\delta\boldsymbol{\omega}^I\right\} \tag{3.10}$$

The virtual strain-displacement relation can therefore be defined as follows for a bilinear displacement field (i.e., $n = 4$):

$$\delta\boldsymbol{\varepsilon} = \begin{Bmatrix}\delta\boldsymbol{\varepsilon}_m \\ \delta\boldsymbol{\varepsilon}_s\end{Bmatrix} = \sum_{I=1}^{n}\begin{bmatrix}\mathbf{B}_m^I \\ \mathbf{B}_s^I\end{bmatrix}\left\{\delta\mathbf{U}^I\right\} \tag{3.11}$$

The membrane strains ($\boldsymbol{\varepsilon}_m$) and shear strains ($\boldsymbol{\varepsilon}_s$) are defined within the shell element domain based on the natural coordinates (ξ, η, ζ). In terms of virtual strains, the strain-displacement relation can be written in a more compact form as $\delta\boldsymbol{\varepsilon} = \mathbf{B}\delta\mathbf{U}$, which incorporates the virtual displacements. Thus, in Eq. (3.11) above, the $\mathbf{U}^I$ term containing nodal displacements of the I^{th} node has become $\delta\mathbf{U}^I$ to reflect the virtual nodal displacements.

3.2.3 Mechanical shell element

By substituting the expressions for the strain-displacement relation and the stress-strain relation (taking into account temperature changes), the mechanical shell stiffness components as well as the thermo-mechanical coupling term can be established. Recall that the volume integral on the left-hand side of Eq. (3.3) is for the full volume of the solid body but note that, in using FEA, the virtual work equation is applied to a subset of the full volume; specifically, to that of a

single element. Using $V^{(e)}$ to denote the element's volume, and substituting Eqs. (3.6-3.11), the internal energy term was transformed into the following form:

$$\int \left(\begin{bmatrix} \mathbf{B}_m \\ \mathbf{B}_s \end{bmatrix} \{\delta \mathbf{U}\} \right)^T \begin{pmatrix} \mathbf{D}_m(\boldsymbol{\varepsilon}_m - \boldsymbol{\varepsilon}_{th}) \\ \mathbf{D}_s \boldsymbol{\varepsilon}_s \end{pmatrix} dV^{(e)} \tag{3.12}$$

As in Eq. (3.10), a summation over the nodes will be needed as well, with indices *I* and *J* ranging from 1, 2, …, *n* nodes, where *n* = 4 for the bilinear element considered here. Substituting the strain-displacement relationship in Eq. (3.8) into the internal energy integral of Eq. (3.12) and using the matrix transpose properties to expand the first product gives the updated matrix-vector expression inside the integrand:

$$\int (\delta \mathbf{U}^I)^T \begin{bmatrix} (\mathbf{B}_m^I)^T & (\mathbf{B}_s^I)^T \end{bmatrix} \begin{bmatrix} \mathbf{D}_m(\mathbf{B}_m^J \mathbf{U}^J - \mathbf{A}\theta) \\ \mathbf{D}_s \mathbf{B}_s^J \mathbf{U}^J \end{bmatrix} dV^{(e)} \tag{3.13}$$

The integrand of Eq. (3.13) now includes the scalar temperature difference, θ, as a variable. This temperature difference will later provide the connection to the actual temperature DOF of the layered thermal shell. Performing matrix multiplication in the above equation produces the following form:

$$\int (\delta \mathbf{U}^I)^T \left[(\mathbf{B}_m^I)^T \mathbf{D}_m (\mathbf{B}_m^J \mathbf{U}^J - \mathbf{A}\theta) \quad + \quad (\mathbf{B}_s^I)^T \mathbf{D}_s \mathbf{B}_s^J \mathbf{U}^J \right] dV^{(e)} \tag{3.14}$$

Next, the individual terms can be separated into different integrals:

$$\int (\delta \mathbf{U}^I)^T \left[(\mathbf{B}_m^I)^T \mathbf{D}_m \mathbf{B}_m^J \mathbf{U}^J \right] dV^{(e)} + \int (\delta \mathbf{U}^I)^T \left[(\mathbf{B}_s^I)^T \mathbf{D}_s \mathbf{B}_s^J \mathbf{U}^J \right] dV^{(e)} - \int (\delta \mathbf{U}^I)^T \left[(\mathbf{B}_m^I)^T \mathbf{D}_m \mathbf{A}\theta \right] dV^{(e)} \tag{3.15}$$

Referring back to the original statement of virtual work presented in Eq. (3.3), the right-hand-side terms can be reintroduced after substituting the above expression for internal energy and considering the finite-element basis functions used similarly for the right-hand side terms. Using the nodal contribution nomenclature from before (namely, the superscripts *I* and *J*), introducing

the right-hand-side, and again specifying the integrals for a typical element, the resulting equation becomes:

$$(\delta\mathbf{U}^I)^T\left\{\left[\int(\mathbf{B}_m^I)^T\mathbf{D}_m\mathbf{B}_m^J\,dV^{(e)}+\int(\mathbf{B}_s^I)^T\mathbf{D}_s\mathbf{B}_s^J\,dV^{(e)}\right]\mathbf{U}^J-\left[\int(\mathbf{B}_m^I)^T\mathbf{D}_m\mathbf{A}\theta\,dV^{(e)}\right]-\mathbf{R}_M^I\right\}=0 \qquad (3.16)$$

Here, $\mathbf{R}_M$ is the right-hand-side forcing term for the mechanical shell element, which is defined as follows for the element (superscript *I* dropped for element-level equation here):

$$\mathbf{R}_M=\int\mathbf{N}^T\mathbf{F}\,dV^{(e)}+\int\mathbf{N}^T\mathbf{\Phi}\,dS^{(e)} \qquad (3.17)$$

Note that a capital *M* is used to signify mechanical stiffness terms, and **N** contains the element's basis functions. From the expression in Eq. (3.16), it is clear that, since the statement must hold for all virtual displacements $\delta\mathbf{U}^I$, all the terms within the main curly braces that right-multiply the virtual displacement term must sum to zero.

The familiar components of the element stiffness equations begin to appear in this development. The first two terms from Eq. (3.16) are given next and are exactly as presented in the original reference for this element [16]. Reproducing those terms here, the stiffness components of the mechanical shell element are:

$$\mathbf{K}_m^{IJ}=\int(\mathbf{B}_m^I)^T\mathbf{D}_m\mathbf{B}_m^J\,dV^{(e)} \qquad (3.18)$$

$$\mathbf{K}_s^{IJ}=\int(\mathbf{B}_s^I)^T\mathbf{D}_s\mathbf{B}_s^J\,dV^{(e)} \qquad (3.19)$$

Here, the subscripts *m* and *s* indicate the membrane and shear contributions to the stiffness. For these element arrays, a selective reduced integration technique was employed for reducing the shear locking effects in shell elements [22].

In order to present the complete formulation of the mechanical shell element given in the original publication, it is important to briefly mention that a penalty method was used to generate

a third stiffness component for *torsional* restraint. The torsional stiffness was represented as $\mathbf{K}_t$ and its details are not reproduced from [16] here other than simply stating that this torsion term restricts free rotation about the surface normal of the shell element and left-multiplies the displacement DOF similar to how a typical stiffness term would in the prior development up to this point. Thus, the mechanical stiffness term $\mathbf{K}_M$ is as follows:

$$\mathbf{K}_M^{IJ} = \mathbf{K}_m^{IJ} + \mathbf{K}_s^{IJ} + \mathbf{K}_t^{IJ} \tag{3.20}$$

The mechanical stiffness term can then be substituted into the main governing equation:

$$\mathbf{K}_M^{IJ}\,\mathbf{U}^J - \int (\mathbf{B}_m^I)^T \mathbf{D}_m \mathbf{A}\theta\, dV^{(e)} = \mathbf{R}_M^I \tag{3.21}$$

The form of Eq. (3.21) above is clearly incomplete as the element volume integral for the coupling term is still undefined. At this point in the development, it is typical for the thermal strain terms to be converted to right-hand-side forcing terms in which the thermal strain enters the FEA governing equations as a thermal load in the uncoupled thermo-mechanical analysis. However, the fully coupled formulation treats the temperatures as DOF to be solved for simultaneously with displacements.

3.2.4 Coupled shell element

The remainder of the formulation presented here seeks to establish the thermoelastic stiffness components that are produced by the temperature change. As the temperature change only appears in the σ_{11} and σ_{22} stress terms, the thermoelastic coupling term can be isolated from the virtual work equation. In a typical finite element, strains $\boldsymbol{\varepsilon}$ are interpolated from nodal displacements $\mathbf{U}$ according to the strain-displacement matrix $\mathbf{B}$, i.e., $\boldsymbol{\varepsilon} = \mathbf{BU}$. In the original formulation, the strain-displacement matrix was decomposed into membrane $\mathbf{B}_m$ and shear $\mathbf{B}_s$ terms to simplify the notation [16]; this was repurposed in Eq. (3.11) of the current study. Keeping

the original notation, the main coupling term that arises from this development is the integral seen in Eq. (3.21), which is isolated here:

$$-\int (\mathbf{B}_m^I)^T \mathbf{D}_m \mathbf{A}\theta \, dV^{(e)} \tag{3.22}$$

Note that the thermal strain has been included as $\mathbf{A}\theta$ where θ is the temperature difference (a scalar). The negative sign in the front of Eq. (3.22) is to preserve the sign of this term, as it appears on the left-hand-side of Eq. (3.21). The negative sign is a result of the constitutive law, which subtracts thermal strains, as was seen in Eq. (3.6).

Equation (3.22) will be separated into two different contributions. The first one appears on the left-hand side of the element equations (i.e., a coupling stiffness $\mathbf{K}_C$ multiplied by the nodal temperatures $\mathbf{U}_T$). The second contribution is a forcing term on the right-hand side of the system of equations to include the reference temperature, which is expressed as $\mathbf{R}_0$ here. Writing the equation of motion for the mechanical shell element to include the coupling terms that arise from Eq. (3.22) gives the following equation:

$$\mathbf{K}_M^{IJ} \mathbf{U}_M^J + \mathbf{K}_C^{IL} \mathbf{U}_T^L = \mathbf{R}_M^I + \mathbf{R}_0^I \tag{3.23}$$

In Eq. (3.23), $\mathbf{K}_M$ is the elastic stiffness for the shell element, as defined in Eq. (3.20), $\mathbf{U}_M$ is the vector of nodal displacements and rotations, and $\mathbf{R}_M$ is the vector of nodal forces. These terms remain unchanged from the original formulation [16] and are denoted by superscripts *I* and *J*, which each range from nodes 1, 2, …, *n*; thus, the contributions from nodes *I* and *J* make up the (*I, J*) sub-matrix of $\mathbf{K}_M$. The superscript *L* that is appended to the new coupling term ranges from 1, 2, …, *NL* number of layers in the layered thermal shell. Equation (3.23) gives the governing finite-element equations for a layered, deformable elastic body with a consideration for temperature variations through the volume of the solid as provided by the thermoelastic

constitutive law of Eq. (3.6). $\mathbf{R}_0$ is not explicitly given here; it represents the forcing term for the reference temperature T_0 which arises from the temperature difference $\theta = T - T_0$.

By bringing the governing equation of the thermal shell element from Eq. (3.2) together with the mechanical shell equation that includes thermoelastic coupling in Eq. (3.23), the full system of coupled equations for the thermo-mechanical shell element can finally be expressed in the following matrix-vector form for a single element of the fully assembled system:

$$\begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{C}_C & \mathbf{C}_T \end{bmatrix} \begin{bmatrix} \dot{\mathbf{U}}_M \\ \dot{\mathbf{U}}_T \end{bmatrix} + \begin{bmatrix} \mathbf{K}_M & \mathbf{K}_C \\ \mathbf{0} & \mathbf{K}_T \end{bmatrix} \begin{bmatrix} \mathbf{U}_M \\ \mathbf{U}_T \end{bmatrix} = \begin{bmatrix} \mathbf{R}_M + \mathbf{R}_0 \\ \mathbf{R}_T \end{bmatrix} \tag{3.24}$$

This gives the form of the governing equations needed for a coupled analysis. The exact expressions of $\mathbf{K}_C$ and $\mathbf{R}_0$ in Eq. (3.24) arise from the assumptions that are embedded in the thermal and mechanical shell formulations and these terms are unique to the coupled formulation. For completeness, the thermoelastic damping term $\mathbf{C}_C$ is also shown in Eq. (3.24) as the pre-multiplier of velocity. However, in the quasi-static equilibrium problem presented here, the velocity DOF are assumed zero and are thus represented by the zero-vector in the equation (i.e., $\dot{\mathbf{U}}_M = \mathbf{0}$). While not discussed in detail here, the thermoelastic damping term $\mathbf{C}_C$ is defined in the ANSYS documentation [23] as the following: $\mathbf{C}_C = -T_0 (\mathbf{K}_C)^{\mathrm{T}}$ and may be used in problems with evidence of strong coupling.

The coupling stiffness matrix $\mathbf{K}_C$ pre-multiplies temperature DOF on the left-hand-side of the governing system of equations in Eq. (3.24). Returning to the development that will lead to the desired final form given in Eq. (3.24), the typical strain-displacement matrix term $\mathbf{B}_m{}^I$ has been split into its two sub-components using the following relation from Eq. (3.8) to explicitly factor out the ζ component: $\mathbf{B}_m = [\mathbf{B}_{1m} \quad \zeta\mathbf{B}_{3m}]$. Notice the natural coordinate ζ representing the through-thickness direction has entered the integrand to left-multiply the submatrix $(\mathbf{B}_{3m})^I$; this will be important in the through-thickness integration step that follows.

Recall that ζ is a natural coordinate in the through-thickness direction of the shell domain. The other two natural coordinates (ξ and η) in the planar dimensions of the shell element are located at the mid-surface. Additionally, the subscripts 1 and 3 were used in the definition of the $\mathbf{B}_m{}^I$ contributions. These labels come from [16] and it is important to note that they were used as subscripts and do not reflect any particular matrix entries. Also note that there is a $(\mathbf{B}_{2m})^I$ term in Eq. (3.8) which is the zero matrix, as mentioned previously.

Substituting this expression of $\mathbf{B}_m{}^I$ and using the previous definitions for $\mathbf{D}_m$, $\mathbf{A}$, and θ in Eq. (3.24) produces the following integrand for the coupling term:

$$-\int \begin{bmatrix} \left(\mathbf{B}_{1m}^I\right)^T \\ \zeta\left(\mathbf{B}_{3m}^I\right)^T \end{bmatrix} \begin{bmatrix} D & \nu D & 0 \\ \nu D & D & 0 \\ 0 & 0 & G \end{bmatrix} \begin{bmatrix} \alpha \\ \alpha \\ 0 \end{bmatrix} (T - T_0)\, dV^{(e)} \tag{3.25}$$

This updated expression in gives two terms in the full formulation: a left-hand-side stiffness component including the scalar T as nodal/layer DOF (shown next) and a right-hand-side forcing term taking into consideration the reference temperature T_0 (the $\mathbf{R}_0$ component). For the first part of the coupling stiffness definition, the focus will be on representing the temperature DOF as left-hand-side components while the reference temperature term will be saved for a later discussion.

The integrand of Eq. (3.25) may be simplified by multiplying the $\mathbf{D}_m\mathbf{A}$ product first and converting the integral to natural coordinates while introducing the Jacobian matrix $\mathbf{J}$ to provide the mapping of bases as in a typical isoparametric formulation. The result is the following expression, where the constant κ is introduced after the matrix multiplication of the two interior arrays $\mathbf{D}_m\,\mathbf{A}$:

$$\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \begin{bmatrix} \left(\mathbf{B}_{1m}^I\right)^T \\ \zeta\left(\mathbf{B}_{3m}^I\right)^T \end{bmatrix} \begin{bmatrix} \kappa \\ \kappa \\ 0 \end{bmatrix} T\left|\mathbf{J}\right| d\zeta\, d\xi\, d\eta \tag{3.26}$$

Where κ represents the product of $-\alpha D(1+\nu)$. Then, κ can be factored out of the volume integral as follows:

$$\kappa \int_{-1}^{1}\int_{-1}^{1} \left\{ \int_{-1}^{1} \begin{bmatrix} \left(\mathbf{B}_{1m}^{I}\right)^{T} \\ \zeta\left(\mathbf{B}_{3m}^{I}\right)^{T} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} T\,|\mathbf{J}|\,d\zeta \right\} d\xi\, d\eta \tag{3.27}$$

Three main features shall be noted to summarize the resulting expression in Eq. (3.27):

1. The 3×1 array containing the thermoelastic constant κ in Eq. (3.26) effectively sums columns one and two of its pre-multiplier, namely the $\mathbf{B}_m{}^I$ matrix, and scales it by the thermoelastic constant κ, which was subsequently factored out to the front of the equation in the form shown in Eq. (3.27).
2. The fiber integral (i.e., the integral for through-thickness integration) was highlighted using the curly braces above and will be important with regards to how the layers are handled.
3. Temperature DOF will be introduced through the T variable above; as it was stated above, for now T is simply the continuous field variable $T = T(\xi, \eta, \zeta)$ in the shell element domain. The finite-element interpolation has not been introduced.

3.2.5 Through-thickness integration

The expression given in Eq. (3.27) shows the fiber integral explicitly marked with curly braces. From this equation, we can define the coupling stiffness matrix $\mathbf{K}_C$ as seen in Eq. (3.24) by using Simpson's Rule for numerical integration through the thickness. This integration approach requires an odd number of sampling points; in this case, the sampling points are the ζ-coordinate locations through the shell thickness where temperature DOF exist at the nodes. Thus, to use this approach, the thermo-mechanical shell element requires an odd number of layers, which

must be greater than or equal to three. This integration method uses a quadratic approximation of the temperature solution through the thickness and therefore two points or less is not sufficient.

Section 3.2.1 described the thermal shell element and discussed how temperature DOF were used through the thickness. An important relation in that development was the fact that at a particular layer in the shell, the temperature in that layer could be computed by using the element basis functions to interpolate among the nodal temperatures of that layer. The temperature of a point in layer *k* for the *n*-noded shell was computed as follows:

$$T = \sum_{I=1}^{n} N^I T^{I(k)} \tag{3.28}$$

Once again, the basis functions of the element are signified by N^I and superscript (k) has been added to denote layer *k* for clarity.

Using Simpson's Rule for numerical integration through the thickness and the temperature field *T* as described in terms of nodal DOF as in Eq. (3.28), the fiber integral of Eq. (3.27) can be evaluated by addressing the contributions associated with $\mathbf{B}_{1m}$ and $\mathbf{B}_{3m}$ separately. First, notice that the $\mathbf{B}_{1m}$ matrix in Eq. (3.27) is not multiplied by ζ. Since both $\mathbf{B}_{1m}$ and $\mathbf{B}_{3m}$ are independent of ζ, they do not need to remain within the fiber integral. The same thing can be said of the 3×1 array containing $[1\ 1\ 0]^T$ which serves to only add the first two columns of the $\mathbf{B}_m$ matrix that pre-multiplies it. For the $\mathbf{B}_{1m}$ matrix, the components of the fiber integral reduces to the following:

$$\int_{-1}^{1} T \left|\mathbf{J}\right| d\zeta \tag{3.29}$$

Equation (3.28) is substituted into Eq. (3.29) and then Simpson's 1/3rd Rule is applied. Specifically, Simpson's Rule was used to evaluate the integrand at the layer coordinates ζ_k for layers k = 1, 2, …, *NL* during the numerical integration which gives:

$$\int_{-1}^{1} T\,|\mathbf{J}|\,d\zeta = \int_{\zeta_1}^{\zeta_3} T\,|\mathbf{J}|\,d\zeta + \int_{\zeta_3}^{\zeta_5} T\,|\mathbf{J}|\,d\zeta + \cdots + \int_{\zeta_{NL-2}}^{\zeta_{NL}} T\,|\mathbf{J}|\,d\zeta$$

$$\begin{aligned} &\approx \frac{h}{3}\left(\mathbf{N}\,\mathbf{T}^{(1)}\left|\mathbf{J}^{(1)}\right| + 4\,\mathbf{N}\,\mathbf{T}^{(2)}\left|\mathbf{J}^{(2)}\right| + \mathbf{N}\,\mathbf{T}^{(3)}\left|\mathbf{J}^{(3)}\right|\right) \\ &+ \frac{h}{3}\left(\mathbf{N}\,\mathbf{T}^{(3)}\left|\mathbf{J}^{(3)}\right| + 4\,\mathbf{N}\,\mathbf{T}^{(4)}\left|\mathbf{J}^{(4)}\right| + \mathbf{N}\,\mathbf{T}^{(5)}\left|\mathbf{J}^{(5)}\right|\right) \quad + \cdots + \\ &+ \frac{h}{3}\left(\mathbf{N}\,\mathbf{T}^{(NL-2)}\left|\mathbf{J}^{(NL-2)}\right| + 4\,\mathbf{N}\,\mathbf{T}^{(NL-1)}\left|\mathbf{J}^{(NL-1)}\right| + \mathbf{N}\,\mathbf{T}^{(NL)}\left|\mathbf{J}^{(NL)}\right|\right) \end{aligned} \tag{3.30}$$

In Eq. (3.30), the superscript (k) was added to the Jacobian matrix to indicate that it should be evaluated at each layer coordinate ζ_k and the basis-function array **N** was substituted into Eq. (3.30) as well. In the Simpsons 1/3rd Rule used here, the numerical integration weights can be seen as the {1, 4, 1} factors for the three points used per sub-integral. Another factor on the weights is the additional h/3 term where h is the distance between two integration points. For the isoparametric shell element mapped to the bi-unit domain, h is simply defined as 2.0 / (NL – 1) where the 2.0 comes from the length of the interval [-1, 1] of the original integral. The general form of Eq. (3.30) is as follows:

$$\int_{-1}^{1} T\,|\mathbf{J}|\,d\zeta \approx \frac{h}{3}\mathbf{N}\left(\mathbf{T}^{(1)}\left|\mathbf{J}^{(1)}\right| + 4\sum_{i=2,4,\ldots}^{NL-1}\mathbf{T}^{(i)}\left|\mathbf{J}^{(i)}\right| + 2\sum_{j=3,5,\ldots}^{NL-2}\mathbf{T}^{(j)}\left|\mathbf{J}^{(j)}\right| + \mathbf{T}^{(NL)}\left|\mathbf{J}^{(NL)}\right|\right) \tag{3.31}$$

The convenience of this approach can be seen from Eq. (3.31) whereby it is clear that the basis functions **N** have been factored out and now the nodal temperature DOF $\mathbf{T}^{(k)}$ for k = 1, 2, …, NL are visible and consistent with the format of the original thermal shell element. This same approach was used for the $\mathbf{B}_{3m}$ term in Eq. (3.28) as well. The details will be spared because Eq. (3.31) may be used in the same manner but now with an extra ζ term in the integrand that must be included in the numerical integration step. This was accomplished by means of adding a factor of

ζ_k applied to each term in the summation seen in Eq. (3.31) for use with the $\mathbf{B}_{3m}$ partition previous shown in Eq. (3.27).

The details of the integration in the plane of the shell have been left out. The process is typical for a finite element employing Gaussian Quadrature in the plane. One feature to note from this development is the fact that the basis function array **N** does not need to remain within the fiber integrand because it is not a function of ζ and thus does not change through the thickness. By factoring out the temperature DOF from the evaluated fiber integral shown in Eq. (3.31), the coupling stiffness term $\mathbf{K}_C$ remains. The approach for the reference temperature contribution is similar but extremely simplified due to T_0 being a constant that does not vary through the thickness. The final note on the mechanical shell stiffness terms is that selective reduced integration was used for the integration of the mechanical shell element array while the standard 2×2 Gaussian quadrature was used layer-wise to form the element arrays of the thermal shell element.

3.3 Numerical results

A series of verification studies will be presented in the following subsections that characterize the performance and highlight the benefits of the thermo-mechanical shell element presented here. The numerical results show a verification of the element through benchmark testing in preparation for the full fire-structure coupling procedure for particular applications in the intended final use case. Prior to performing tests with the coupled thermo-mechanical shell element, the implementation of the mechanical shell element was verified using traditional benchmark testing as provided, for example, in the Abaqus Benchmarks Manual [20]. Specifically, the sections of *2.3.1 The barrel vault roof problem* and *2.3.2 The pinched cylinder problem* were used to ensure the displacements were calculated accurately with this particular user-element

(UEL) implementation of the original element formulation. Verification of the thermal shell element implementation used here was performed in previous studies [4,5].

For the barrel vault roof, the UEL computed the reference displacement at the desired node with an error of -3.3% using the coarsest mesh of 4×4 elements. The results improved to an error of -0.67% for the finest mesh of 18×18 used in this test. These results were comparable to the closest Abaqus library equivalent for this element: the S4 general shell element. Results for the S4 element were +4.7% and +0.77% for the same meshes, respectively. For the pinched cylinder, using the finest mesh of 20×20 elements, the UEL computed the reference displacement with an error of +4.3% while S4 resulted in -4.0% error. Note that the for the torsion coefficient term mentioned earlier, a value of κ_T = 1.0 was used to produce these results with the UEL implementation (see [16] for details on the torsional stiffness term).

3.3.1 Thermal stress in a cylinder

The first example using the thermo-mechanical shell element is a verification problem for computing the stress in a hollow cylinder subjected to a higher temperature (200 °C) on the inside surface and a lower temperature (100 °C) on the outer surface. These conditions on each of the surfaces were provided by specifying a temperature boundary at the surface of the shell element. This test was defined in the Abaqus Verification Manual (*1.3.17 Thermal stress in a cylindrical shell*) [20]. In Fig. 3-4 below, the full cylindrical structure is shown and the single-element mesh required for this benchmark is highlighted, as only one element was needed for this test. Additionally, a steady state condition was assumed according to the documentation.

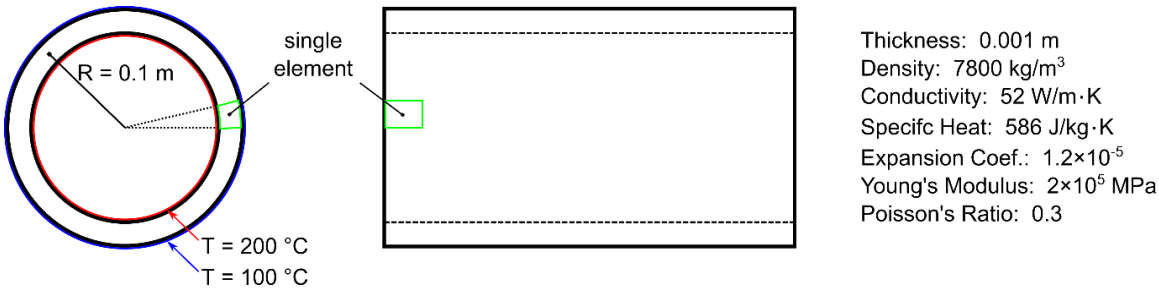


Figure 3-4: Verification problem for analyzing thermal stress in a cylinder

The size of the element used in this test corresponds to one that subtends an angle of 11.25° at the center of the cylindrical shell opening; thus, it is consistent with a scenario in which 32 elements compose the circumferential direction. Proper boundary conditions were specified for the stress analysis such that the rotations about the circumferential direction were constrained but the cylinder was allowed to extend freely due to thermal expansion along its axial dimension, in accordance with the original problem statement. A reference solution was given in the Abaqus Verification Manual [20] for this problem. The theoretical stress at the outer and inner surfaces of the cylindrical structure should be ±171.43 MPa, with the plus/minus sign depending on the surface.

Results computed by Abaqus using the Fortran implementation of the proposed thermo-mechanical shell element matched this reference solution exactly. In fact, using the analytical solution for this problem to compute the stress with more significant digits, the thermo-mechanical shell element demonstrated more than sufficient accuracy, showing an absolute error of $O(10^{-13})$ when comparing computed stresses with the analytical solution. The thermo-mechanical shell element presented here also performed the same as the similar elements provided in the Abaqus library for this problem, such as the S4T coupled temperature-displacement shell.

3.3.2 Simply supported heated plate

A simply supported plate problem for measuring deflection due to non-uniform heating was modified for a thermoelastic shell analysis. Originally, the reference solution for this problem was derived for a plate and presented in a classical mechanics textbook on plate and shell theory [24]. Since the model was specified for a plate problem, modifications were needed for the thermo-mechanical shell version of this verification in order to provide proper in-plane restraint. The original plate problem and its boundary conditions are shown in Fig. 3-5 below; in this scenario, a linear temperature difference was applied through the thickness of the of the plate.

The original reference [24] provides a series solution to the partial differential equation for the out-of-plane displacement (i.e., along the z-axis in this model) for the deflection $w(x, y)$ pertaining to this case. Using this provided solution, not given here for brevity, the deflection at any point could be obtained and compared with a finite-element model employing the thermo-mechanical shell element. However, modifications were needed for using shell elements in this test in order to restrain in-plane, rigid-body motions. The boundary conditions used for the shell-element model of this problem are given in Fig. 3-6, where in-plane translations were restrained on two edges to provide uniform thermal expansion towards the x_{max} and y_{max} sides.

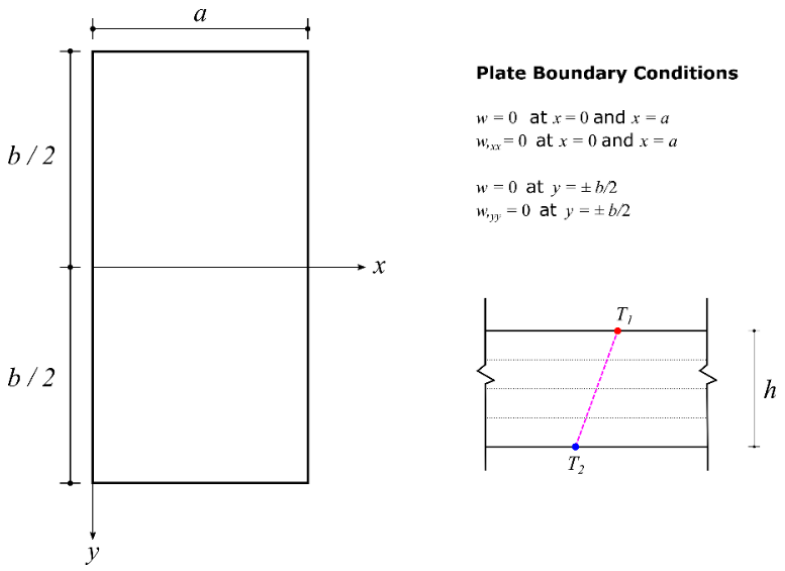


Figure 3-5: Simply supported rectangular plate geometry and boundary conditions; the linear temperature through the thickness *h* is shown as well

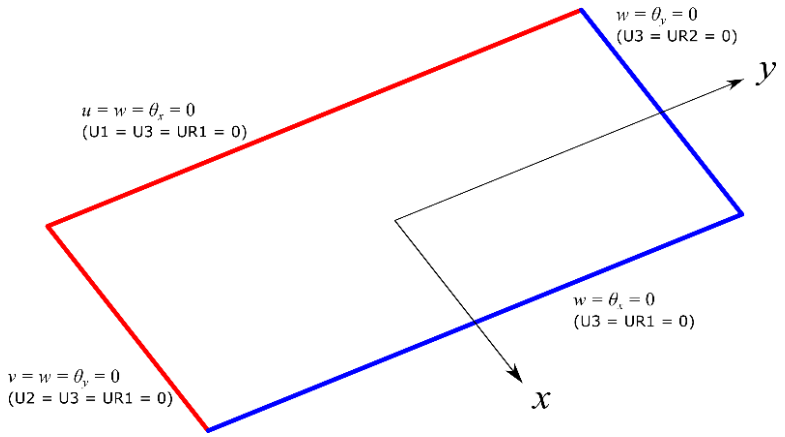


Figure 3-6: Simply supported plate with displacement boundary conditions explicitly defined for the 3D FEA model employing shell elements

The particular material properties and dimensions chosen for this problem were selected objectively, as the provided solution is general for any thermoelastic problem. Thus, the dimension *a* and *b* from the original diagram in Fig. 3-5 were 1.0 m and 2.0 m, respectively, and the thickness of the plate *h* was 0.1 m. The thermoelastic material properties were a Young's modulus of 2.0×10^{10} Pa, Poisson's ratio of 0.20, and coefficient of thermal expansion of 1.0×10^{-5} m/(m·°C).

Finally, the only load on the system was the linear temperature field prescribed through the thickness such that the difference $T_1 - T_2$ was equal to 100 °C.

For this test, two meshes were used for the FEA models: 4×8 and 8×16 elements in the plane of the plate model via the UEL implementation of the thermo-mechanical shell element in Abaqus. Five temperature layers were used through the thickness. In order to define the linear temperature field through the thickness, the exact temperature was specified at each layer. Therefore, in the five-layer case used here, temperatures of 0, 25, 50, 75, and 100°C were directly prescribed as the temperature DOF for each layer as input. This approach isolated the thermo-mechanical coupling features that were targeted for investigation in the model; namely, this allowed for directly assessing the deformation based on a known temperature field.

The results of this verification test are shown in Fig. 3-7 below, where the out-of-plane displacement w was plotted along two lines of the plate. These lines correspond to the $x = a/2$ and $y = 0$ lines, referring to the coordinate system of Fig. 3-5 where these lines represent the centerlines in the x- and y-directions. The five curves in the figure come from the theory-based reference solution according to the original plate example [24], the results for the new thermo-mechanical shell element presented here (UEL), and, for comparison, the coupled temperature-displacement shell provided by Abaqus (S4T).

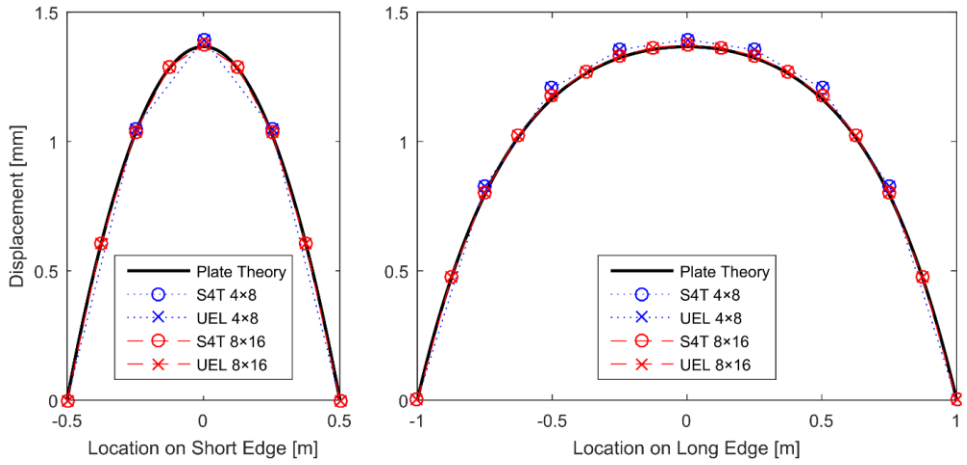


Figure 3-7: Comparison of the results for the out-of-plane displacement of the plate along the two centerlines of the domain using thermo-mechanical shell elements

To quantify the difference between the theoretical solution and the one obtained by the UEL for the *w* displacement along these two lines, the relative error based on the two-norm was computed. The 8×16 mesh was used for this comparison. Namely, the two-norm of the difference in the computed and reference solution, $\|\mathbf{w} - \mathbf{w}_{UEL}\|_2$ was computed and then divided by the two-norm of the solution, $\|\mathbf{w}\|_2$, from the provided reference solution for **w** as an array composed of the scalar *w* evaluated at the nodes along each of the lines. Thus, the two-norm relative error between the shell element and the reference solution was 7.1×10^{-3} for the long edge and 4.9×10^{-3} for the short edge. As another measure of the accuracy, the displacement at the center of the plate, where the maximum deflection occurred, was recorded as 1.373 mm using the UEL shell; the reference solution at this point was 1.366 mm, which is a difference of 0.5% between the two. Additionally, the results for the two FEA solutions, UEL and S4T, matched each other exactly: the two-norm of the difference between displacements computed by the UEL and S4T models was zero, using full double-precision formatting to record the displacement output and compute the differences. In Fig. 3-7 above, the *w* displacement in this case corresponded to the U3 displacement in Abaqus. The two-norms for the differences in displacements for the other components, such as U1-U2 translations and UR1-UR2-UR3 rotations, were also zero when comparing the two FEA models at

all the nodes in the mesh. Thus, these two verification problems served to demonstrate the ability of the new implementation to accurately model three-dimensional deformation of the simply supported plate due to a prescribed temperature field through the thickness.

3.4 Conclusions

In order to improve the state of fire-structure simulation methods, an approach using coupled thermo-mechanical shell elements was presented in the current study. By developing the finite-element equations for a coupled thermo-mechanical shell element and subsequently implementing the formulation as a Fortran subroutine in Abaqus, the presented shell element showed beneficial results from a computational modeling perspective and the verification problems demonstrated the accuracy of the coupled formulation.

One of the practical limitations of this approach is the fact that a larger system of equations is solved for each increment of the coupled analysis: two uncoupled systems, of sizes $m \times m$ and $n \times n$, are solved for the weakly coupled problem whereas an $(m+n) \times (m+n)$ system is solved in this strongly coupled formulation. Here, m is the total number of displacement DOF in the model originating from the mechanical shell element while n is the total number of temperature DOF in the model from the layered thermal shell for conduction heat transfer analysis.

While this approach has shown promising early results, additional consideration and future research will be needed to apply these methods in scenarios with more structurally significant fire exposure. The use of fire-structure coupling methods will be employed in the next stages of the research in order to include boundary conditions from CFD-based fire simulations as input for the FEA model. For such cases, the analyst must consider the effects of temperature-dependent

material properties and the potential for large displacements or buckling: both features deserve additional exploration in the future work.

3.5 Acknowledgements

The authors would like to thank the Office of Naval Research's support on contract number N00014-13- C-0373, which funded a major portion of work completed during this project.

3.6 References

[1] McGrattan K, Hostikka S, McDermott R, Floyd J, Weinschenk C, Overholt K (2015) Fire dynamics simulator user's guide, 6 edn. National Institute of Standards and Technology, NIST Special Publication 1019, Gaithersburg. doi:http://dx.doi.org/10.6028/NIST.SP.1019.

[2] X. Yu, A.E. Jeffers, A comparison of subcycling algorithms for bridging disparities in temporal scale between the fire and solid domains, Fire Saf. J. 59 (2013) 55–61. doi:10.1016/j.firesaf.2013.03.011.

[3] P.A. Beata, A.E. Jeffers, Spatial homogenization algorithm for bridging disparities in scale between the fire and solid domains, Fire Saf. J. 76 (2015) 19–30. doi:10.1016/j.firesaf.2015.05.008.

[4] A.E. Jeffers, Heat transfer element for modeling the thermal response of non-uniformly heated plates, Finite Elem. Anal. Des. 63 (2013) 62–68. doi:10.1016/j.finel.2012.08.009.

[5] A.E. Jeffers, P.A. Beata, Generalized shell heat transfer element for modeling the thermal response of non-uniformly heated structures, Finite Elem. Anal. Des. 83 (2014) 58–67. doi:10.1016/j.finel.2014.01.003.

[6] A.E. Jeffers, Triangular Shell Heat Transfer Element for the Thermal Analysis of Nonuniformly Heated Structures, 142 (2016) 1–9. doi:10.1061/(ASCE)ST.1943-541X.0001335.

[7] K.S. Surana, P. Kalim, Isoparametric axisymmetric shell elements with temperature gradients for heat conduction, Comput. Struct. 23 (1986) 279–289. doi:10.1016/0045-7949(86)90219-1.

[8] K.S. Surana, R.K. Phillips, Three dimensional curved shell finite elements for heat conduction, Comput. Struct. 25 (1987) 775–785. doi:10.1016/0045-7949(87)90169-6.

[9] K.S. Surana, N.J. Orth, Axisymmetric shell elements for heat conduction with p-approximation in the thickness direction, Comput. Struct. 33 (1989) 689–705. doi:10.1016/0045-7949(89)90243-5.

[10] K.S. Surana, G. Abusaleh, Curved shell elements for heat conduction with p-approximation in the shell thickness direction, Comput. Struct. 34 (1990) 861–880. doi:10.1016/0045-7949(90)90357-8.

[11] K.S. Surana, N.J. Orth, Three-dimensional curved shell element based on completely

hierarchical p-approximation for heat conduction in laminated composites, Comput. Struct. 43 (1992) 477–494. doi:10.1016/0045-7949(92)90282-5.
[12] A. Bose, K.S. Surana, Piecewise hierarchical p-version axisymmetric shell element for non-linear heat conduction in laminated composites, Comput. Struct. 47 (1993) 1–18. doi:10.1016/0045-7949(93)90274-H.
[13] A.K. Noor, W.S. Burton, Steady-state heat conduction in multilayered composite plates and shells, Comput. Struct. 39 (1991) 185–193. doi:10.1016/0045-7949(91)90086-2.
[14] N. Mukherjee, P.K. Sinha, A comparative finite element heat conduction analysis of laminated composite plates, Comput. Struct. 52 (1994) 505–510. doi:10.1016/0045-7949(94)90236-4.
[15] J. Noack, R. Rolfes, J. Tessmer, New layerwise theories and finite elements for efficient thermal analysis of hybrid structures, Comput. Struct. 81 (2003) 2525–2538. doi:10.1016/S0045-7949(03)00300-6.
[16] W. Kanok-nukulchai, A simple and efficient finite element for general shell analysis, Int. J. Numer. Methods Eng. 14 (1979) 179–200. doi:10.1002/nme.1620140204.
[17] W.Y. Jung, S.C. Han, An 8-node shell element for nonlinear analysis of shells using the refined combination of membrane and shear interpolation functions, Math. Probl. Eng. 2013 (2013). doi:10.1155/2013/276304.
[18] F.M. Adam, A.E. Mohamed, A.E. Hassaballa, Degenerated Four Nodes Shell Element with Drilling Degree of Freedom, IOSR J. Eng. 3 (2013) 10–20.
[19] R.. Cook, D.. Malkus, M.. Plesha, R.. Witt, Concepts and Applications of Finite Element Analysis, 2002.
[20] Abaqus 6.12 Documentation, (2012). Simulia.
[21] C. Zhang, J.G. Silva, C. Weinschenk, D. Kamikawa, Y. Hasemi, Simulation Methodology for Coupled Fire-Structure Analysis: Modeling Localized Fire Tests on a Steel Column, Fire Technol. 52 (2016) 239–262. doi:10.1007/s10694-015-0495-9.
[22] T.J.R. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Prentice-Hall, Inc. (1987) 825. doi:620/.001/51535.
[23] ANSYS User Manual, (2012).
[24] S.P. Timoshenko, S. Woinowsky-Krieger, Theory of Plates and Shells, McGraw-Hill, 1959.

Chapter 4
Applications of the Thermo-mechanical Shell Element in Coupled Fire-structure Analyses

In the previous chapter, the formulation of the thermo-mechanical shell element was presented along with two verification problems. While the shell element was designed for the coupled fire-structure problem, applications of this new development using boundary conditions from a computational fluid dynamics (CFD) based fire simulation were not previously provided. The main goal of the current investigation was to demonstrate the capability of the thermo-mechanical shell element implemented as a user-defined subroutine (UEL) in Abaqus. The fire-structure coupling methods discussed in Chapter 2 were implemented in the coupled shell presented in Chapter 3. Two applications, one involving a rectangular plate and the other a structural I-beam, were chosen to show how the element can be used with spatially and temporally varying boundary conditions representing the CFD-based fire. Additionally, various choices of the mesh size and time step in coupled fire-structure simulations were selected to analyze the performance using different combinations of temporal and spatial discretization using a square plate model exposed to different thermal loads.

4.1 Introduction

The concept of coupled fire-structure interaction problems has been discussed in Chapters 2 and 3 thus far. The three main components of the coupled problem are the CFD-based fire simulation, the coupling of the thermal boundary conditions from the CFD model to the FEA model, and finally the thermo-mechanical analysis of the structure. Figure 3-1 in the previous

chapter presents a high-level view of the coupled fire-structure interaction problem using data from a CFD-based fire simulation. In addition to providing a means for modeling the thermo-mechanical response of structures through the use of shell elements (image in the right pane of Fig. 3-1), the coupling methods used to bridge the two domains together are also an important step in the fire-structure interaction problem (image in middle pane of Fig. 3-1). In Chapter 2, the methods for sequentially coupling the thermal boundary conditions from the fire to the solid domain were presented. In Chapter 3, the formulation of the coupled thermo-mechanical shell element was provided. In the current chapter, applications employing the shell element with the coupling methods implemented at the element level are provided to show the use of this methodology in the fire-structure coupling problem.

Structural response under a wide range of potential fire hazards has been of increasing interest in recent structural fire engineering research and modeling the thermal boundary conditions using the CFD-based software of Fire Dynamics Simulator (FDS) is a popular approach for capturing the physics of a localized fire. For example, Zhang et al. [1] used adiabatic surface temperatures (AST) as the transfer data from FDS to an FEA model in ANSYS [2] for thermo-mechanical analysis. Luo, Xie and DesJardin [3] provided a method for performing 2D fluid-structure coupling based on a level-set function approach, specifically for use in the fire-structure problem. Chen, Luo, and Lua [4] developed a two-way coupling approach with a customized version of FDS and linked it with Abaqus, which formed the basis of the Abaqus Fire Interface Simulator Toolkit (AFIST). Tondini, Vassart, and Franssen [5] employed a partitioned solution approach to the fire-structure analysis in their development of an interface between CFD fire simulations and FEA heat transfer models for applications to compartment fire models. Finally,

Silva, Landesmann, and Ribiero [6] presented the Fire-Thermo-Mechanical Interface tool used for processing fire simulation results and transferring AST measurements to an FEA model.

The fact that researchers, including the authors of this current study, need to perform subsequent heat transfer and structural simulations is at least in part because FDS, which is the fire simulation software most typically relied upon, is not suited for its conduction analysis in the solid domain. This results in various users applying a wide range of assumptions regarding the representation of the boundary conditions (e.g., AST approaches vs. applied heat flux vs. simplified traveling fires, etc.), and the method for resolving the discrepancies in the mesh (e.g., sampling heat fluxes vs. averaging heat fluxes) and the time step. Presently, the ramifications of these varied assumptions are unknown.

Yu and Jeffers [7] showed that sampling boundary conditions in time could have a significant impact on accuracy of the computed temperatures in the solid due to heavy oscillations in heat fluxes that come from a CFD model. Beata and Jeffers [8] similarly showed that sampling data in space could have consequences if the solid and fluid models have significantly different meshes. However, few “best practices” have been identified in the literature for performing a fire-structure simulation using a coupled CFD-to-FEA simulation approach. Moreover, the verification and validation of existing models is inconsistent, making reproducibility of results a major concern for this application.

In general, the transfer of any CFD-based output representing the thermal boundary conditions as the input of the FEA model for heat transfer is a key aspect for providing an accurate characterization of the non-uniform thermal boundary observed in localized fire scenarios. At its most fundamental level, the fire-structure coupling analysis is a fluid-structure interaction (FSI) problem that is assumed a one-way coupling such that the structural deformation and the

temperature field of the structure do not influence the fire simulation, which is appropriate for many applications. Partitioned systems for multiphysics problems, such as FSI simulations, are a practical approach to connect two field solutions; thus, a partitioned approach was employed in the current study as well.

Matthies and Steindorf [9] in part discuss one of the simplest and best-known types of partitioned approaches for FSI analysis problems, specifically the staggered approach, which represents a weak or loose coupling between the two domains of fluid and solid. Bathe and Zhang [10] focus on compatibility for general fluid-structure coupling in two-way systems; in particular, displacement and traction compatibility were presented as the essential criteria for interaction at an interface of the two domains. These two concepts also have parallels in the fire-structure interaction problem: (1) one-way coupling providing a thermal boundary condition based on CFD-computed data and (2) defining a compatible finite-element representation of the thermal boundary condition at the interfacing surfaces. Additionally, partitioned approaches are attractive because they use two different solvers and source code to produce the coupled solution, which takes advantage of the expertise used to develop each code separately by the experts of the individual sciences [11].

This chapter will mainly use the contributions from previous developments in the dissertation. Specifically, modeling the thermo-mechanical response of thin-walled structures to fires using the newly developed shell elements from Chapter 3 and implementing the fire-structure coupling methods based on partitioned solutions from Chapter 2. The relevant information from each chapter will be referred to in describing how the thermal boundary conditions from the fire simulation were used in the coupled fire-structure simulations.

4.2 Fire-structure Coupling using Shell Elements

To handle the transfer of temporal data from the CFD-based fire simulation to the FEA model comprised of thermo-mechanical shell elements, a subcycling technique was used to allow multiple steps of the fluid-domain solution to be time-averaged and passed into the solid domain [7]. Specifically, the net heat flux along a rectilinear grid in the fire simulation was passed from the fire domain to the FEA model containing the structure. The time-averaging approach is based on computing the average of recent heat-flux measurements in the fire simulation within a small window (i.e., the temporal subcycling time step) and using that value as the input for the thermo-mechanical model. The equation for computing the surface flux $\bar{q}''$ at a particular time t_n in the FEA simulation is provided in a previous study [7]:

$$\bar{q}''(t_n) = \frac{\sum_{i=1}^{m} q''(t_i)\,\Delta t_i}{\sum_{i=1}^{m} \Delta t_i} \tag{4.1}$$

The range of the summations in Eq. (4.1) is from one to *m*, where *m* is the number of time increments that fall within the given subcycling time step (window) and is based on the relative time increments used in the CFD and FEA simulations. In the case of the CFD model, this increment is related to the frequency at which data is written to the output file in FDS since the actual time step used to solve the CFD equations is possibly much smaller on some increments. The Δt_i term in Eq. (4.1) is the time step of the output frequency in FDS. When a uniform time step is used, the calculation in Eq. (4.1) reduces to a simple average of the measured heat fluxes in the subcycling step. If the CFD simulation provides heat fluxes every 1.0 seconds and the subcycling step is 10 seconds, then the average is computed for the 10 measured heat fluxes that fall into this moving window throughout the simulation.

Handling spatially non-uniform boundary conditions (such as a net flux) over the surfaces of solid models can be accomplished within the FEA solution framework once the computed surface fluxes are transferred from the fire simulation and mapped to the FEA mesh. Numerical integration is the basis for computing finite-element arrays, both left-hand-side and right-hand-side terms, in any FEA simulation. Traditionally, Gaussian quadrature has been used to compute element arrays by integrating through finite-element volumes and surfaces. Gaussian quadrature is very accurate for the integration of smooth functions and especially polynomials, using a small number of sampling points over an element surface and/or through the volume.

However, for the case of non-uniform heating conditions, as in the case of fire simulations, other methods exist which may be more appropriate for handling discretely measured data points as opposed to smooth functions. For example, integration of the forcing-term arrays at the element interfaces with non-uniform heating may be accomplished with the trapezoid rule for numerical integration by using as many available discrete data points as available at the element boundary [8]. The trapezoidal rule approach uses additional integration points in the element surface integrals to compute equivalent nodal fluxes from the non-uniform fluxes available at the surface. This approach was employed for the fire-structure simulations presented next together with the temporally non-uniform heating methods mentioned earlier. Details of these methods were provided in Chapter 2, where the methods coupling 2D and 3D non-uniform surface fluxes were presented. In the following subsections, applications of the thermo-mechanical shell element are provided to demonstrate the use of this new shell with the fire-structure coupling methods included as an additional subroutine in the UEL.

4.3 Plate Exposed to Local Fire

With the successful verification of the thermo-mechanical shell element in the previous chapter, the implementation of this element was upgraded to handle non-uniform thermal boundary conditions. The approach was consistent with previous work [7,8] but now in the setting of coupled temperature-displacement analyses in Abaqus. For this application, a flat plate was exposed to a localized fire simulated using FDS; specifically, FDS was used to compute a temporally and spatially non-uniform surface flux field which was sequentially applied as a net flux boundary condition in the FEA model on the bottom surface of the plate. This form of one-way coupling allows the researcher or analyst to perform a single fire simulation (at a high computational expense) and use the data in various FEA models employing different mesh configurations, time steps, and other boundary conditions (at a significantly lower computational expense in comparison to the CFD-based fire simulation).

On the opposite surface (i.e., on the top of the plate) a uniform convection boundary condition with a heat transfer coefficient of 25 $W/m^2 \cdot K$ and a sink temperature of 20°C was defined for this particular set of tests. The edge faces of the plate were thermally insulated. Displacement boundary conditions represent the same simply supported restraints as shown in Fig. 3-6 previously for the flat plate verification problem in the last chapter. The remaining material properties and model details for the current plate problem are shown in Fig. 4-2 below as well.

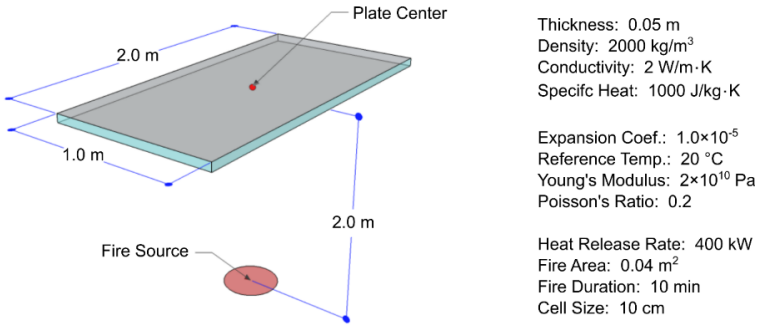


Figure 4-1: Model details and material properties for the flat plate exposed to a local fire

Employing the newly presented element as a UEL in Abaqus, a series of coupled temperature-displacement analyses were completed for this single fire scenario. Note that a subcycling time step of 10 seconds was chosen for the data transfer process from FDS to the Abaqus model based on the temporal subcycling methods presented in an earlier study [7]. As in the two verification problems, five layers were used through the thickness of the shell elements.

The first demonstration of the results in this problem is the dependency of out-of-plane deflection on the FEA mesh chosen for the plate model (i.e., translation in the *z*-direction, such as *w* in the previous verification). Four different meshes were used in the plane of the plate model: 2×4, 4×8, 8×16, and 16×32 elements. Each element had four nodes in accordance with the bilinear shell formulation presented earlier, where each node contained three translation DOF and three rotation DOF at the mid-surface of the shell and five temperature DOF corresponding to the five layers through the thickness. In Fig. 4-2 below, the temperature and corresponding displacement results throughout the duration of the fire event are shown for the node at the center of each mesh: at the location ($x = 0$, $y = 0$) with respect to the coordinate system in Fig. 3-6, specifically. The computed temperatures were provided for the bottom (non-uniform heating), mid-surface (interior), and top (convection boundary) layers in the model, however, the displacements were computed at the mid-surface only.

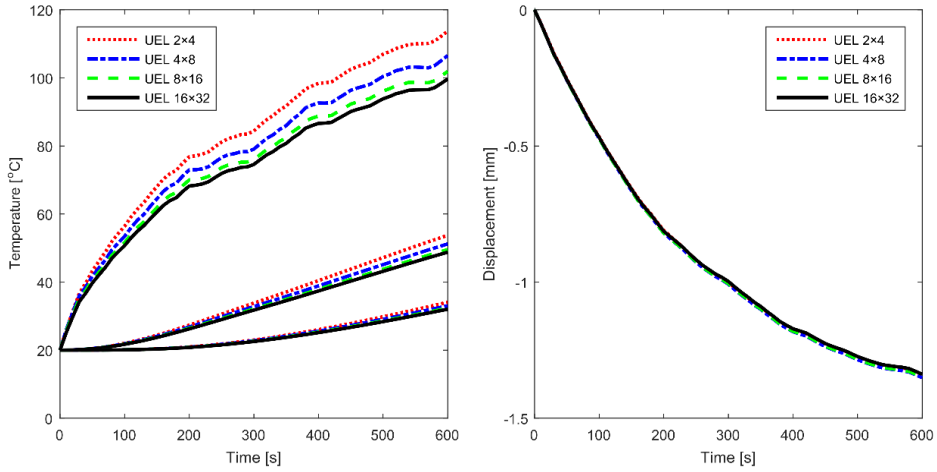


Figure 4-2: Temperature and displacement results at the center of the plate; temperatures provided at the top, middle, and bottom (highest temperatures) of the plate at this central point

The results for the computed temperatures and displacements along two lines in the plate domain are presented next. Specifically, with respect to the coordinate system on the plate reference surface shown in Fig. 3-6, the results are provided along the line of $y = 0$, for $x = [-0.5, 0.5]$, and the line $x = 0$, for $y = [-1.0, 1.0]$. The temperatures in Fig. 4-3 were the nodal temperatures at the non-uniformly heated (bottom) surface after three minutes and ten minutes, respectively. As in the previous demonstrations, the out-of-plane displacements are shown in Fig. 4-4 for at both three and ten minutes as well.

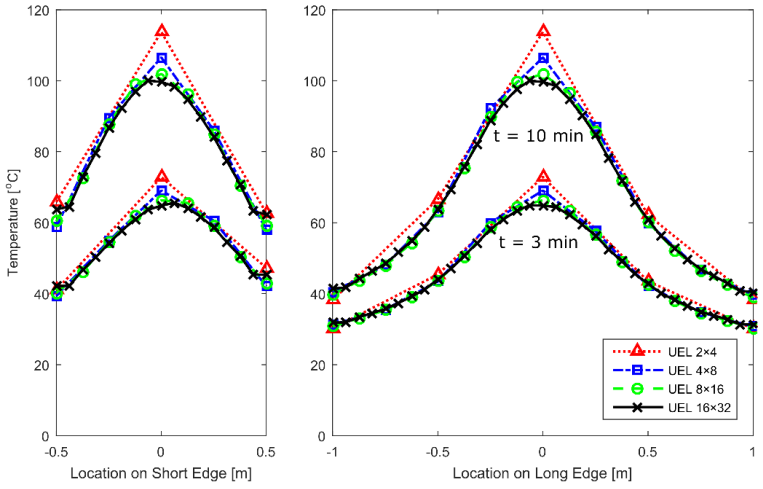


Figure 4-3: Temperatures on the heated surface along the two centerlines of the plate at 3 min and 10 min using four different meshes

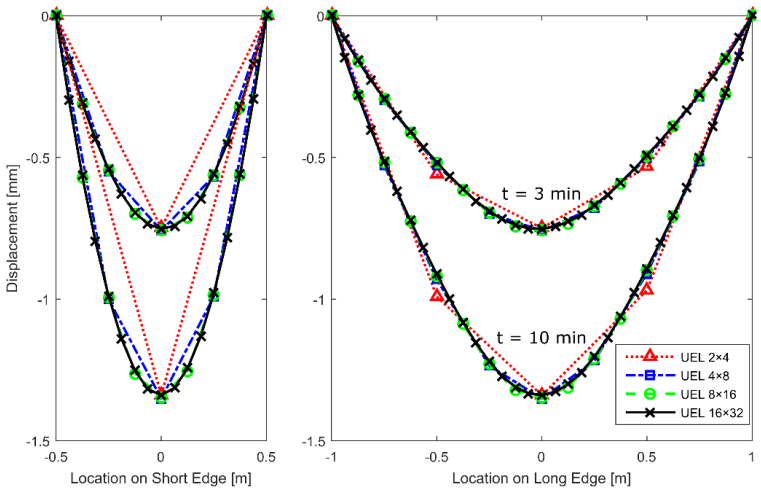


Figure 4-4: Displacement along the two centerlines of the plate at 3 min and 10 min

The following contour plots demonstrate the computed results in the plane of the plate at the final time step in order to show a representation of the final temperature and displacement fields. In Fig. 4-5, the mesh for each test is shown followed by the final temperature state at the heated (bottom) surface and the final out-of-plane displacement state. Qualitatively, these results show that by increasing the number of elements in model, thereby using more nodes and DOF, the computed profile of the temperature and displacement fields converge.

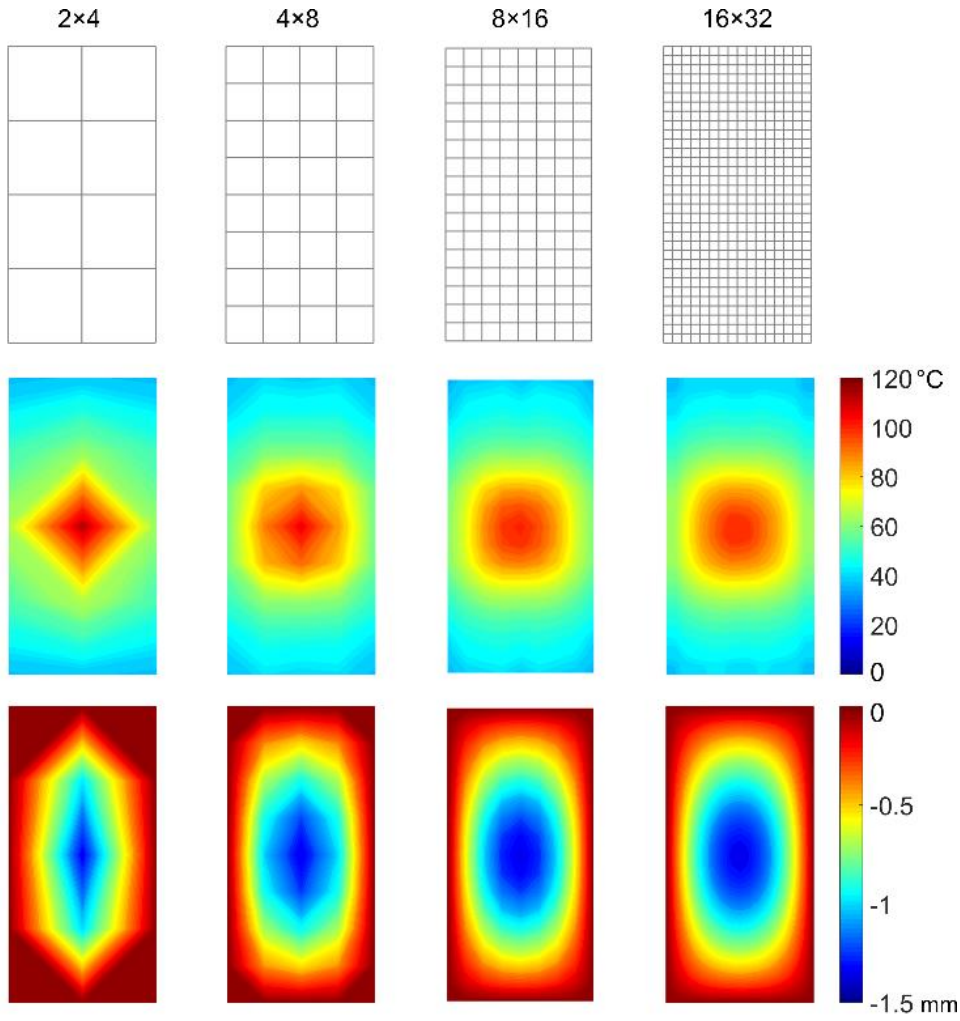


Figure 4-5: The meshes used for each test (top), the resulting temperature field for the heated surface at the final time step (middle), and the corresponding out-of-plane displacement at this time (bottom)

Simulation times for these models were recorded in two parts: first, preprocessing time for mapping the FDS boundary conditions to the FEA model, as described in Chapter 2, and second, the actual thermo-mechanical simulation of the structure using the shell element from Chapter 3. The preprocessing step, much like the fire simulation, was only needed one time; this is one benefit of the coupling procedure used which allows various FEA meshes to map the FDS output data individually. Preprocessing performed by MATLAB scripts required 1.1 sec for the 10-min worth of flux data used in this test. The time required for each transient temperature-displacement simulation in Abaqus was shown in Table 4-1 below, in addition to the final temperature and

displacement calculated by each model. The total time in the final column of Table 4-1 represents the Abaqus-measured CPU time found in the output DAT file plus the preprocessing time of 1.1 sec mentioned earlier.

Table 4-1: Properties of the FEA models and the required simulation times; note that the temperatures and displacements are the results at the final time step

Mesh	DOF	Surface Temp. at Center [°C]	Disp. at Center [mm]	Total Time [s]
2×4	165	113.8	-1.340	3.1
4×8	495	106.6	-1.351	5.9
8×16	1683	101.9	-1.349	16.5
16×32	6171	99.7	-1.339	59.7

Convergence of the results for the nodal temperatures at the exposed surface as well as the transverse displacements was analyzed using the two-norm relative error. For the reference solution in this application, the finest mesh of 16×32 elements was used for comparison with the results of the other meshes. Figure 4-6 demonstrates the convergence behavior for the FEA meshes used in this study with respect to the total DOF in the system.

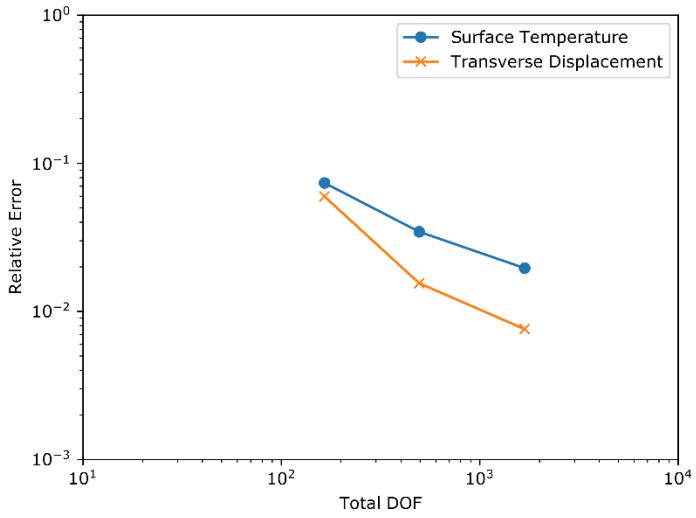


Figure 4-6: Convergence plot for temperature and displacement; computed using the two-norm relative error between each mesh with the finest mesh (16×32)

4.4 Beam Exposed to Local Fire

For the final numerical example, the fire-structure coupling method was extended to handle multiple interfaces, as in the case of the I-beam scenario shown in Fig. 4-7 below. This problem presents the need to map fluxes measured in the FDS fire simulation to multiple surfaces of the beam flanges and web. The particular shape selected for this test was a W460×52 wide-flange steel section. Dimensions and properties were defined in Fig. 4-7 as well and the fire source was modeled using FDS, just as in the plate fire problem previously, with the fire source offset from the beam center, as depicted in the figure. The beam was simply supported with a pinned connection at one end and a roller at the other, with torsion sufficiently restrained for stability of the beam member. This scenario allowed for bending as in a traditional beam problem, with the boundary conditions for the pin and roller defined at the neutral axis on each end of the beam.

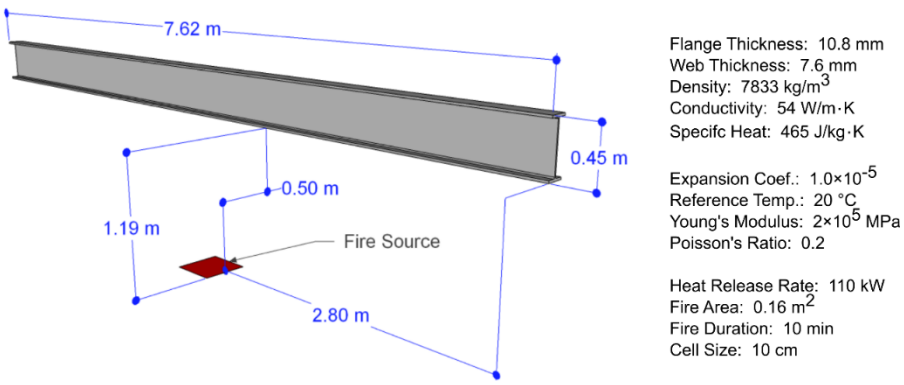


Figure 4-7: Beam model exposed to a local fire with dimensions and material properties provided; the fire is off center by 0.5 m in the CFD model

The purpose of this test was to demonstrate the ability of the thermo-mechanical shell element in conjunction with the fire-structure coupling techniques to model a realistic structural component exposed to a localized fire simulated by FDS. For example, Fig. 4-8 shows the final computed temperature distribution along the axial direction of the beam with the localized heating included in the analysis. Additionally, the displacement field is shown below it in the same image.

The results provided here used a modification of the coupling method described earlier which now uses a traditional 2×2 Gaussian quadrature approach for computing equivalent nodal fluxes from the non-uniform net heat flux data. This was appropriate in this case based on the relative sizes of the FEA elements and CFD cells; the appropriate trapezoidal rule application would also employ a 2×2 integration scheme for this test. Temperature results are shown at the mid-surface layer of each component of the beam model; in this test, only three total temperature layers were used in the through-thickness direction of the web and flanges. Additionally, the image in the right-most frame of Fig. 4-8 shows the beam deflection from its centerline corresponding to displacement in the *x*-axis for this particular model.

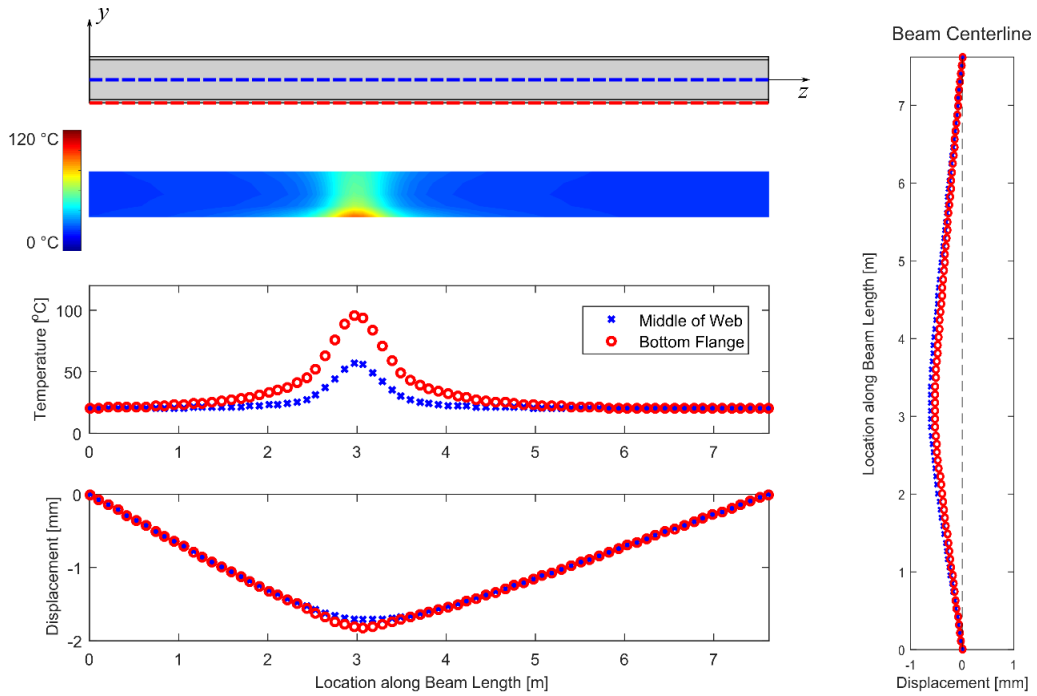


Figure 4-8: Temperature and displacement results at the final time step along the neutral axis (middle of the web) and along the bottom flange centerline; the right image shows the displacement from the centerline (i.e., displacement U1 along the x-axis in the Abaqus model)

One of the benefits of this thermo-mechanical shell analysis combined with the fire-structure coupling methods is the ability to perform the heat transfer and structural analysis using the same mesh and in a single analysis step (i.e., a multiple-time-increment coupled temperature-

displacement analysis step). In the case of a beam application, the analyst may wish to refine the mesh in order to investigate the convergence of their FEA model or change some other parameter of the solid domain. It is important to note that the FDS fire simulation need only be completed one time for a given CFD mesh and the data can subsequently be mapped to various finite element models of the same structure.

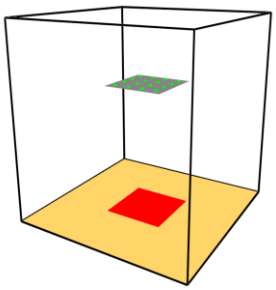
For this particular mesh, 576 elements using three temperature layers totaled 5,913 thermal and mechanical DOF; the total computation time was less than two minutes for this mesh. The preprocessing step, much like the fire simulation, was only needed one time just as in the previous test with the flat plate. With three main sections in this I-beam model, namely the web and two flanges, the preprocessing step performed by the MATLAB scripts for data sorting and mapping required 8.8 seconds for this particular geometry. Note the increase in this computing time from the previous test, which only had one surface (the bottom of the plate) and required 1.1 seconds, while the I-beam model had six surfaces (top and bottom of the web and flanges). The actual simulation required was 98.3 seconds, as recorded in the Abaqus output file in terms of CPU time. Combined with the preprocessing step, the total time required was 107.1 seconds using Abaqus 6.14 and MATLAB R2015b on a standard personal computing workstation with a 2.53 GHz processor in this case.

4.5 Analysis of the Time Step and Mesh Size

A test matrix was developed in order to strategically vary several analysis parameters related to the time step and mesh size used in the coupled fire-structure simulation. First, three boundary conditions were selected: a uniformly applied flux field, a bilinear flux field, and, finally, the net flux from a true fire simulation from Fire Dynamics Simulator (FDS). Second, three grid

sizes were chosen to represent these various boundaries as discrete data points: for example, using a coarse, medium, and fine grid to represent uniform, bilinear, and CFD-based heat flux data. These parameters were related to the boundary condition data only and change the representation of the thermal data as a thermal load on the structural system. The third variable was the subcycling time step used for representing temporally varying data from the fire simulation. Five subcycling steps were used to average the temporal data which was passed from the fire model to the structural model. Finally, the fourth variable was the size of the finite elements used in the FEA model containing the thermo-mechanical UEL elements. Five finite-element mesh sizes were chosen for this study, where each element size was halved in subsequent models.

To test the impact of selecting particular time steps and mesh sizes, a square plate model was used as the basis for the suite of tests. The square plate measured 1.0 $\times$ 1.0 m in plane and 5 cm thick. On one surface, an applied flux was used as the thermal boundary condition; on the opposite surface, a convection boundary was provided with a heat transfer coefficient of 25 W/m^2·K and atmospheric temperature of 20°C. The model for this test is provided in Fig. 4-9 below which shows a particular case of the FDS fire boundary. Figure 4-9 gives the basic test setup and fixed parameters while the different simulation variables used in the comparison study and are discussed further. Note that only the results of the thermal shell were analyzed here.



Density:	2000 km/m^3
Conductivity:	2 W/(m·K)
Specific Heat:	1000 J/(kg·K)
Expansion Coef.:	1.0 $\times$ 10^{-5} / °C
Young's Modulus:	2.0 $\times$ 10^4 MPa
Poisson's Ratio:	0.2
Reference Temp.:	20 °C
Plate Thickness:	5 cm
Exposure Duration:	10 min

Figure 4-9: FDS model for a particular configuration used in the mesh size and time step study

In total, there were three heating cases, three boundary grid sizes, five subcycling time steps, and five FEA mesh sizes, as mentioned previously. A summary of the actual values used for

each variable is given in Table 4-2 below. This series of testing resulted in 225 FEA simulations employing various sets of boundary data. In order to use the non-uniform thermal boundary condition coupling methods, uniform and bilinear boundary data was transformed in a standard way to match the same format as the output of a fire simulation conducted via FDS. Specifically, a constant value of 10.0 kW/m^2 in the uniform case was represented as a set of discrete data points in a .csv file, as if it were "recorded" at discrete locations using standard heat flux devices in FDS. Each boundary condition was applied for 10 minutes in all of the cases presented here.

Table 4-2: Overview of the variables used in the testing matrix

3	Boundary Conditions: uniform, bilinear, FDS fire ▪ Uniform: Q(x, y, t) = 10.0 kW / m^2 ▪ Bilinear: Q(x, y, t) = 10.0(x + y) ▪ FDS Fire: Q(x, y, t) = (*data from fire simulation*)
3	Boundary Grid Size: coarse, medium, fine ▪ For uniform and bilinear case: [1, 1/4, 1/16] m ▪ For FDS fire case only: [1, 1/4, 1/10] m
5	Subcycling Time Step: [60, 30, 15, 10, 5] s
5	Finite Element Edge Size: [1, 1/2, 1/4, 1/8, 1/16] m
225	**Total Number of Tests**

In order to observe the results of choosing particular FEA mesh sizes relative to the boundary condition mesh size, the testing matrix presented in Table 4-2 provided several different combinations of boundary grid size and FEA mesh size. Note that for the case in which actual FDS fire simulation data was used, the discretization of the boundary mesh needed to change slightly to accommodate the device spacing in the FDS software in a convenient manner. As a result, the finest grid size used to represent boundary data was changed from 1/16-m to 1/10-m for the FDS-generated fire data only, as shown in Table 4-2.

The various cases produced by the relative sizes of each mesh are shown in Fig. 4-10 below.

The top row of labels gives the size of the FEA mesh, the far-left column of labels gives the size of the boundary grid size, and finally the bold number floating above each mesh is the "relative size" of the FEA mesh compared to the boundary data grid. For example, a bold value of **4** indicates the FEA mesh was "4 times the size" of the boundary grid mesh. In Fig. 4-10, the visible mesh (i.e., the structured cells) represents the FEA mesh sizes while the small circles represent the boundary data grid as individual FDS devices.

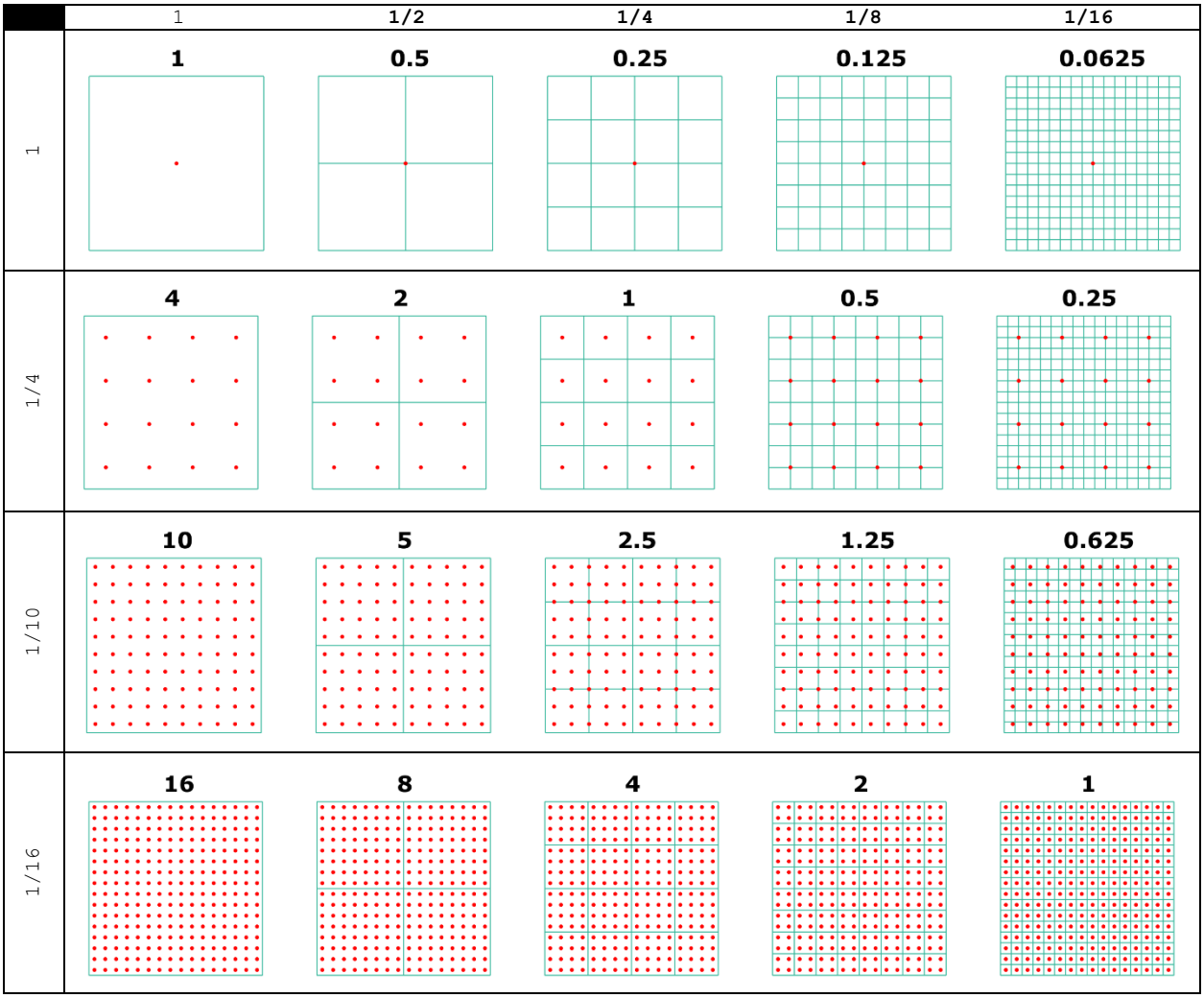


Figure 4-10: Comparison of the various mesh sizes used in the study and the corresponding boundary data grids; the number above each mesh is the ratio of the FEA element size (green outline of each element) to the CFD cell size (red dot for the center of each cell)

In order to provide the raw results of all 225 tests, temperature-time plots were created for particular points in the plate domain; Fig. 4-11 shows the locations of the two particular points

used in the following plots. The planar view of the plate from above is shown with the points A and B located at the center of the plate area and the top-right corner, respectively.

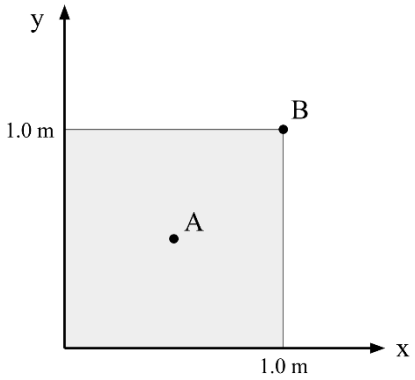


Figure 4-11: Diagram showing the locations of points on the plate selected for the temperature-time plots

Results for the temperature at Point A for the uniform boundary condition are shown in Fig. 4-12, where the left, center, and right images correspond to the exposed surface (i.e., heated face), middle surface, and top surface of the plate, respectively. Similar results are shown in Fig. 4-13 but at Point B for the bilinear boundary condition. Finally, Fig. 4-14 shows the results at Point A once again but for the FDS-computed surface fluxes. Note that every case is shown here resulting in 75 curves per case (i.e., per boundary condition). The reader should note that it is best to view these results in color, as this is simply an overview of the full suite of unfiltered results: more attention and detail will follow as particular cases were selected for discussion. However, for a very high-level view of the data, the finest boundary grid data is represented by the blue curves, the medium grid size by the black curves, and finally the coarsest grid was the red curves.

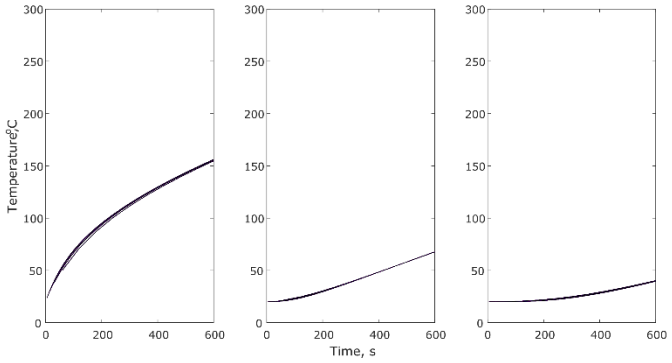


Figure 4-12: Temperature-time curve at Point A for the uniform boundary condition

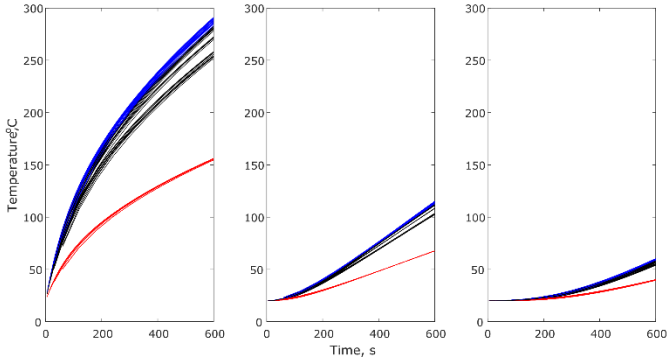


Figure 4-13: Temperature-time curve at Point B for the bilinear boundary condition

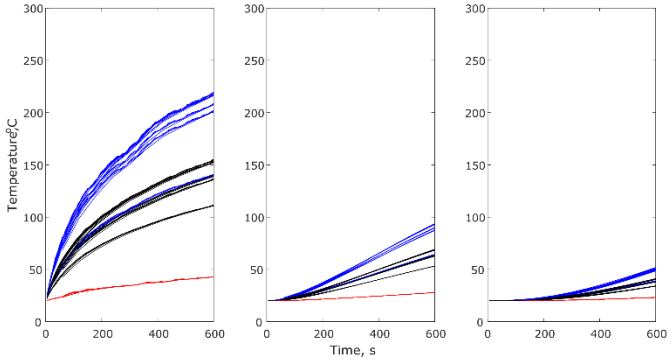


Figure 4-14: Temperature-time curve at Point A for the FDS-based boundary conditions

Upon initial inspection of the data, it was clear that for the bilinear case and the FDS case, there was no need to use the results from the single-data point representation of the boundary conditions (i.e., the red curves). This was not related to the performance of the shell but rather due to the fact that a single data point cannot accurately represent the boundary data of the bilinear

case or the FDS case in a meaningful way. Thus, the test results from the first row of Fig. 4-10 were filtered out of the processing that follows. Proof of this claim can be seen in Figs. 4-13 and 4-14 in which the lowest-magnitude groupings of test results in each case were those produced by the coarsest boundary grid; if viewing this report in color, notice that these were the red curves.

In order to focus on the particular impact of the mesh sizes, a few cases have been selected for discussion next. For all three boundary cases, the coarse representation of boundary data was removed as mentioned in the previous segment. The finest data transfer time step (i.e., subcycling time step) of 5 seconds was chosen to filter the results even more. Finally, only temperatures at the exposed surface and the middle surface were considered. The temperature at either Point A or B at the final time of the simulations (i.e., at the 10-min mark). Thus, for each of the three cases (uniform, bilinear, and FDS data) and at each of the two layers (exposed and middle) there were 10 tests: five using the medium data grid (1/4 m) and five using the fine data grid (1/10 m or 1/16 m for the bilinear and FDS cases, respectively). Those five tests cover the range of FEA mesh sizes used in Table 4-2. Results from this subset of data are shown in Fig. 4-15 next. Figure 4-15 (a-c) show the results for the uniform, bilinear, and true FDS fire tests, respectively.

(a)

(b)

(c)

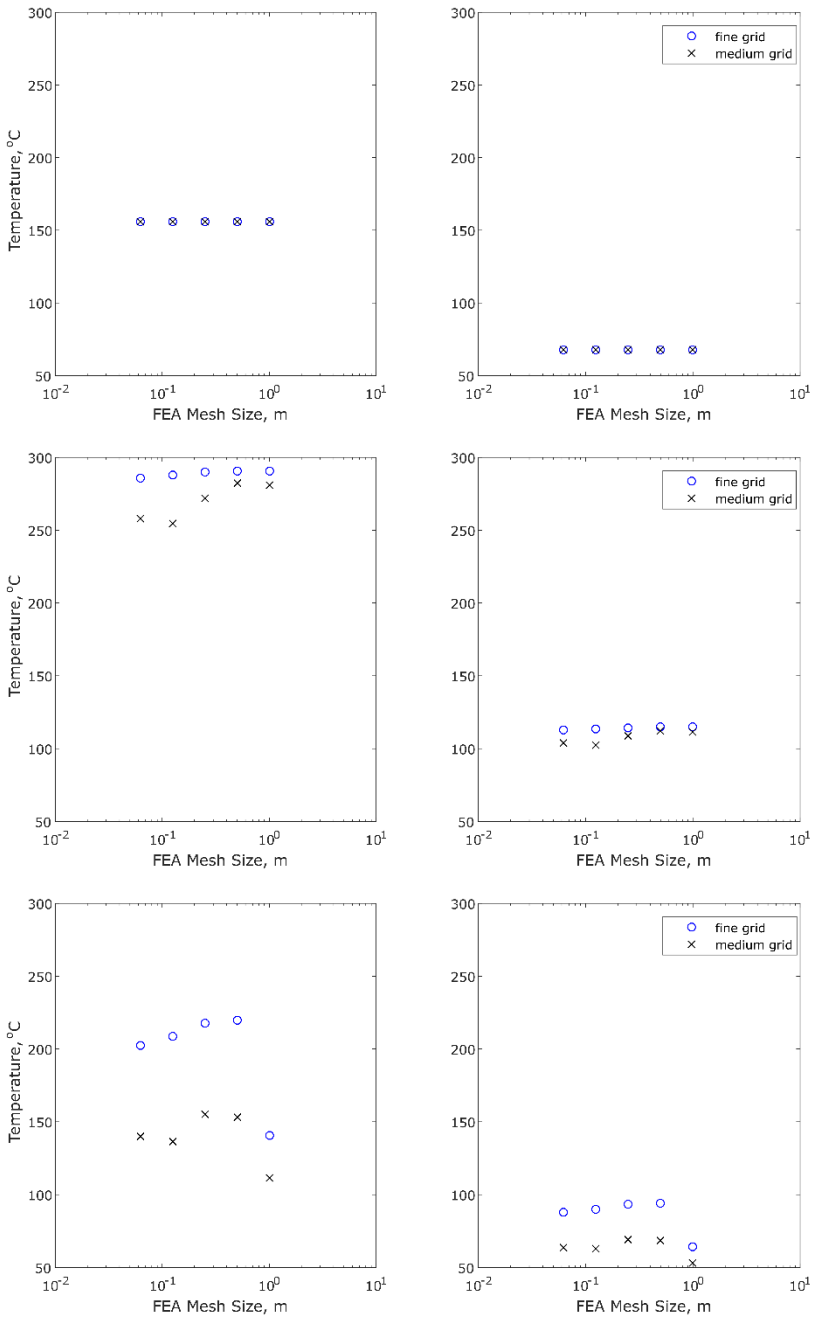


Figure 4-15: Temperature results at the final step for either point A or B (as in previous figure) with respect to the FEA mesh size for the three cases of (a) uniform, (b) bilinear, and (c) FDS boundary conditions

A similar filtering to show the dependence on time step choice is shown next. Here, the finest FEA mesh size of 1/16 m was chosen and the five results related to the time steps are shown in each of the images of Fig. 4-16; once again, all three boundary cases were considered and the coarse boundary grid was filtered out of these results.

(a)

(b)

(c)

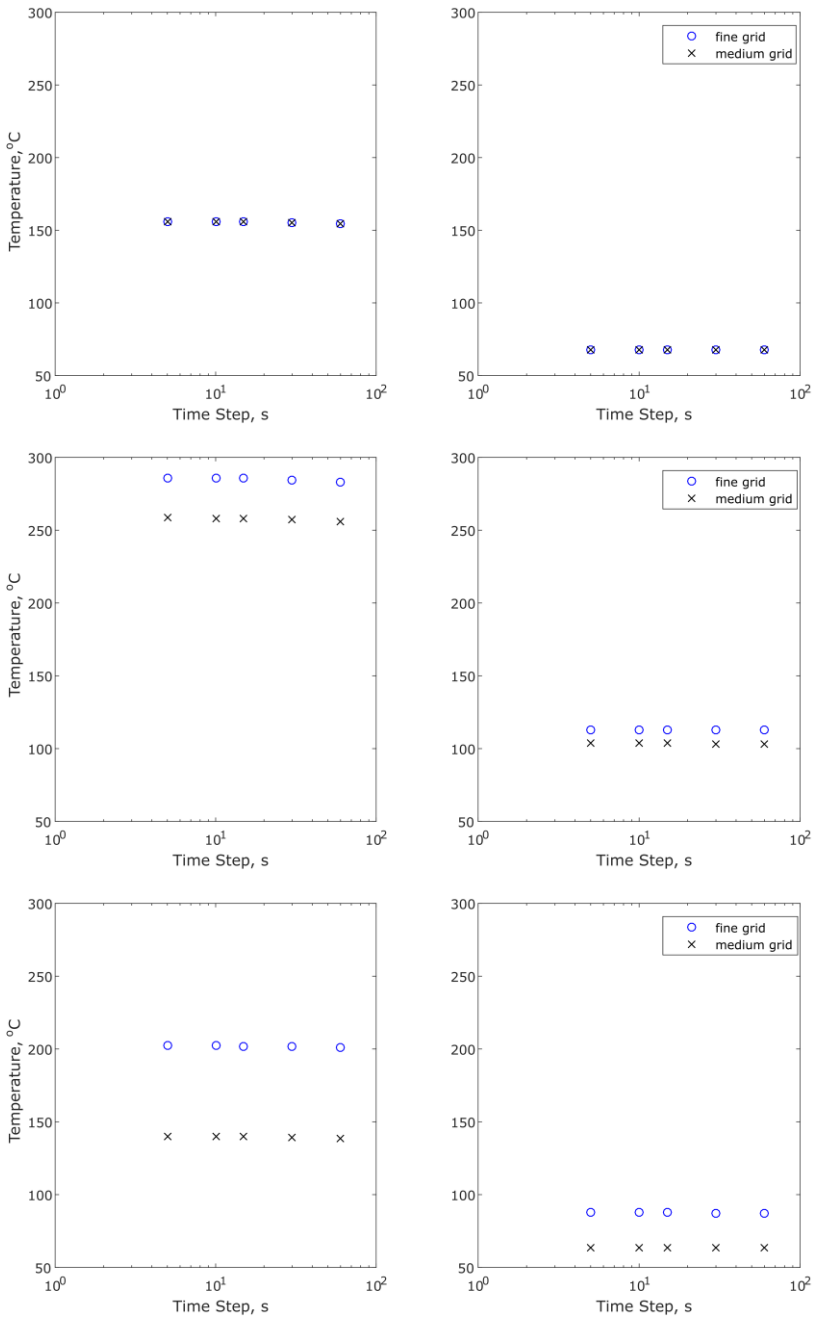


Figure 4-16: Temperature results at the final step for either point A or B (as in previous figure) with respect to the subcycling time step for the three cases of (a) uniform, (b) bilinear, and (c) FDS boundary conditions

4.6 Discussion of Results

The use of the thermo-mechanical shell element was presented in the first two cases for localized heating. The first case was a rectangular plate while the second case was an I-beam; the

main input to the system was the localized fire sources present in each model represented as thermal boundary conditions from the CFD-based fire simulation. Fire-structure coupling methods from previous studies and the methods from Chapters 2 and 3 were used to demonstrate the capability of using coupled shells in this application for localized heating. The analysis of the time step and mesh size on the results of the thermal analysis are discussed in more detail next.

In the case of the uniform boundary conditions (Fig. 4-15a), there was no change in the results from using a finer mesh in representing the boundary data or in defining the elements of the FEA model. The bilinear case showed a slight dependence on the grid size of the boundary data that was more pronounced at the exposed surface (Fig. 4-15b). Finally, in Fig. 4-15c, there was a very significant dependence of the results on the grid size used in FDS (in this case, on the order of 50°C). This was an expected result as it is important to use the proper cell sizes for the FDS fire simulation before conducting any subsequent FEA tests; it was clear that the medium mesh used in the FDS simulation did represent a converged solution for that model. These trends for the boundary grid size continued for Fig. 4-16 when the subcycling time step was varied and the mesh size was held constant. However, there were nearly no changes in the final computed temperature results when the subcycling step was changed for a given boundary grid.

The plate exposed to the local fire showed how the shell could be efficiently linked with fire simulation data; and finally the beam problem provided a practical application of the methods for a typical structural component. In terms of defining a policy for future use, it should be noted that starting with a correct representation of the fire physics in the FDS simulation is the first goal in making sure that the results are reliable. Without a sufficient resolution in the fire model, subsequent results will be tainted in this multi-model solution. Beyond this claim, referring back to the results in Fig. 4-15 for the finest grid, a clear convergence can be seen beyond the single-

element FEA mesh condition for both the bilinear case and the case with FDS data (the single-element result being the value positioned at a mesh size of 1.0 m). This behavior of consistent convergence for halving the element sizes is critical for obtaining accurate results by decreasing the sizes of the element mesh in the FEA model for a given loading condition.

4.7 References

[1] C. Zhang, J.G. Silva, C. Weinschenk, D. Kamikawa, Y. Hasemi, Simulation Methodology for Coupled Fire-Structure Analysis: Modeling Localized Fire Tests on a Steel Column, Fire Technol. 52 (2016) 239–262. doi:10.1007/s10694-015-0495-9.

[2] S. A. S. Ansys, Reference Manual - version 12, Swanson Anal. Syst. Inc (2009).

[3] C. Luo, W. Xie, P.E. DesJardin, Fluid-Structure Simulations of Composite Material Response for Fire Environments, Fire Technol. 47 (2011) 887–912. doi:10.1007/s10694-009-0126-4.

[4] L. Chen, C. Luo, J. Lua, FDS and Abaqus Coupling Toolkit for Fire Simulation and Thermal and Mass Flow Prediction, Fire Saf. Sci. 10 (2011) 1465–1477. doi:10.3801/IAFSS.FSS.10-1465.

[5] N. Tondini, O. Vassart, J.-M. Franssen, Development of an Interface Between CFD and FE Software, in: 7th Int. Conf. Struct. Fire, 2012: pp. 459–468.

[6] J.C.G. Silva, A. Landesmann, F.L.B. Ribeiro, Fire-thermomechanical interface model for performance-based analysis of structures exposed to fire, Fire Saf. J. 83 (2016) 66–78. doi:10.1016/j.firesaf.2016.04.007.

[7] X. Yu, A.E. Jeffers, A comparison of subcycling algorithms for bridging disparities in temporal scale between the fire and solid domains, Fire Saf. J. 59 (2013) 55–61. doi:10.1016/j.firesaf.2013.03.011.

[8] P.A. Beata, A.E. Jeffers, Spatial homogenization algorithm for bridging disparities in scale between the fire and solid domains, Fire Saf. J. 76 (2015) 19–30. doi:10.1016/j.firesaf.2015.05.008.

[9] H.G. Matthies, J. Steindorf, Partitioned strong coupling algorithms for fluid–structure interaction, Comput. Struct. 81 (2003) 805–812. doi:10.1016/S0045-7949(02)00409-1.

[10] K.-J. Bathe, H. Zhang, Finite element developments for general fluid flows with structural interactions, Int. J. Numer. Methods Eng. 60 (2004) 213–232. doi:10.1002/nme.959.

[11] J. Degroote, P. Bruggeman, R. Haelterman, J. Vierendeels, Stability of a coupling technique for partitioned solvers in FSI applications, Comput. Struct. 86 (2008) 2224–2234. doi:10.1016/j.compstruc.2008.05.005.

Chapter 5
Real-Time Fire Monitoring for the Post-Ignition Fire State in a Building

During a fire event, environmental threats to building occupants and first responders include extreme temperatures, toxic gases, disorientation due to poor visibility coupled with unfamiliar surroundings, and a changing indoor environment. In addition to these hazards, firefighters often lack critical information for making decisions on the ground. The lack of information coupled with the dynamics of natural fire events leads to a number of near-misses, injuries, and deaths each year. Additionally, these challenges slow the rescue time of building occupants and prolong the progression of fire. Integrating real-time measurements from sensors into the fire intervention strategy may provide an opportunity for a new technological advancement to improve the practice of firefighting. Specifically, the integrated use of measurement, computation, and visualization could provide a unique new tool for aiding firefighters in their approach to extinguishing a fire and saving building occupants. In this study, a computational framework was developed for connecting real-time fire data to an event detection sub-model to demonstrate the possibility of real-time computing can be used for fire-monitoring and sensor-assisted firefighting. A post-processed example showing this monitoring tool in conjunction with a Building Information Model (BIM) using schedule simulation is provided. The work serves as a step towards an intelligent firefighting system based on computing in an effort to provide real-time tools for effective decision making during the fire event.

5.1 Introduction

This work is based on the anticipation of ubiquitous sensing technology of the future and the subsequent use of measured fire signatures (from these sensors) to help identify fire spread,

burn threats, smoke toxicity, and flashover in a way that would inform firefighters of impending dangers within the burning structure. This research aims to provide a flexible and extendible software infrastructure for a real-time fire monitoring system to be used with a wireless sensor network (WSN) of the future. It will provide firefighters the ability to make data-informed and computation-based decisions during a structural fire event. The novelty of the research lies in the use of multiple fire signatures to identify fire events in real-time and to graphically represent the information in BIM software to facilitate rapid decision-making during firefighting operations. While there are many challenges in terms of hardware, implementation, cost, feasibility, and adaptability, this chapter seeks to provide one part of the solution: a solid computational framework for integrating real-time fire data with computation and visualization for sensor-assisted firefighting.

One such challenge in this line of research and development, which is also unique when compared with other applications such as indoor air quality monitoring, is clear: too much data and information without careful assimilation and presentation will be useless as an aid to firefighting. Cowlard et al. [1] has provided, in addition to the benefits, several cautionary warnings for using sensors, real-time data, modeling, and visualization as intentional tools for helping firefighters. Additionally, Silvani et al. [2] discusses some of the limitations of using a WSN in the fire-monitoring field from the hardware perspective. For example, issues related to time delays and loss of information are important factors to consider in the development of a practical WSN technology for fire monitoring, from a hardware and implementation standpoint specifically. Information overload [1] and hardware limitations [2] in conjunction with restrictions on time for strategizing during the fire are acknowledged as a challenge to this real-world implementation of a fire monitoring system.

The goals of this research were to provide strong foundational contributions to the real-time fire monitoring problem from a computing standpoint. Although the current system is not immediately field-ready, the following work shows recent advances towards a future where real-time fire monitoring may indeed become a beneficial tool for strategic firefighting. Figure 5-1 shows the proposed inclusion of a data-informed fire-monitoring system for providing feedback to the incident commander about the status of the fire. In the traditional firefighting approach, an improvised response must be taken at the scene of the fire event where very few details about the fire are available. In the present study, a real-time fire monitoring system was designed to use sensor data at the scene to provide more information about the fire status inside the building with the goal of augmenting the improvised response with data, computation, and visualization.

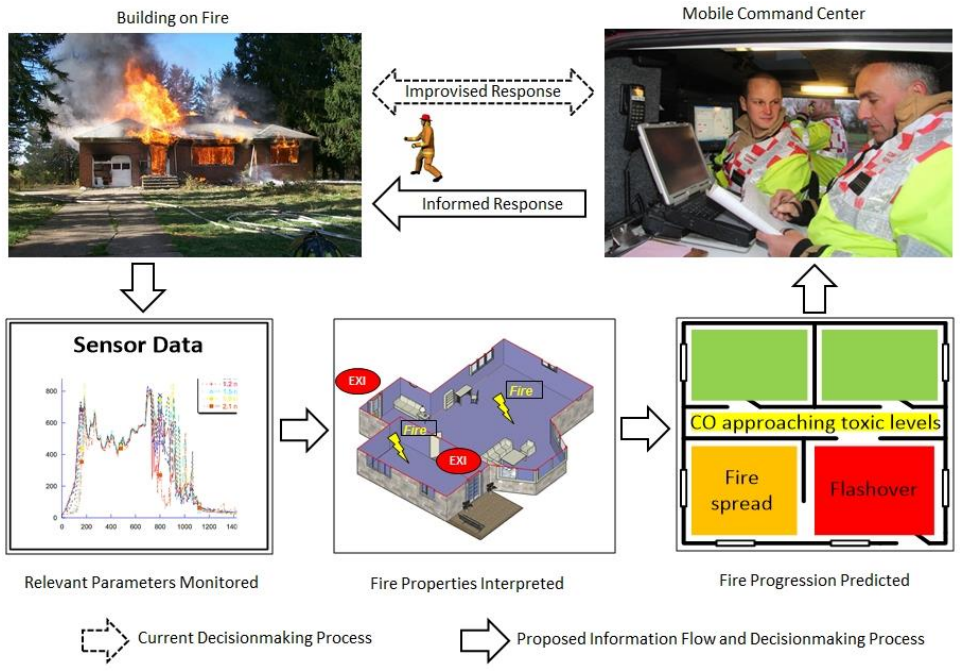


Figure 5-1: The use of data measured at the scene of the fire event could provide an informed response using real-time computation augmented with the traditional improvised response

In the natural sciences and engineering fields, there has been a considerable research interest in monitoring the status of dynamic environments through the use of data and real-time computation. Advances in real-time remote monitoring technologies have been seen in several

non-fire-monitoring scenarios over the last decade, such as in water-quality and environmental monitoring applications [3–8], building energy and building hazard monitoring [9, 10], and indoor air quality monitoring [11]. In some cases, these sample applications are closely related to the problem of monitoring fires in the built environment. For example, Tilak et al. [7] discussed some of the challenges of real-time monitoring in the natural environment where conditions may be extreme and weathering becomes an issue for long-term monitoring. This problem exists in the fire monitoring arena as well, where extreme temperatures, lack of durable wireless technology, and toxic conditions make data collection difficult as the normal ambient conditions deteriorate during a fire.

Real-time monitoring of wildfires is a combination of the environmental application with the primary hazard of interest in current study, i.e., building fires. Several researchers have used WSN for the wildfire application, such as the system presented by Doolin and Sitar [12], which collected temperature, relative humidity, and barometric pressure at the scene of the wildfire event to track its progression over time. Additionally, work by Zervas et al. [13] brings multi-sensor data fusion to the wildfire monitoring field as well, also relying on a WSN but now including vision sensors. Yu, Wang, and Meng [14] applied a neural network approach with in-network data processing for real-time forest fire detection and forecasting. The method relies on using a larger number of cheap sensors densely distributed throughout a forest region. However, the use of real-time monitoring techniques during firefighting within buildings has been difficult to implement and adopt on a large scale and has unique challenges compared to wildfire monitoring, including standards of firefighting practice.

Specifically, for the case of fires in buildings, Jiang et al. [15] developed a context-aware computing system, which was designed to integrate real-time fire data from both static and

dynamic environmental sensors carried by individual firefighters into the building for exchanging information between firefighters. Their platform, called Siren, used a peer-to-peer distributed architecture designed for mission-critical search-and-rescue procedures. The system provided context-aware warnings to individual firefighters at the scene of the fire event. Recent work by Takahashi et al. [16] aimed to provide firefighters with wearable technology for recording species concentrations and measurements of the particulates in the atmosphere during the overhaul phase (i.e., post-fire). Sha, Shi, and Watkins [17] demonstrated an example of this in the area of fire rescue. Their system, named *FireNet*, was designed as a WSN in which each individual firefighter was part of the network and broadcasted their local information to a main computer located with the incident commander. Lim et al. [18] proposed a framework which can detect fires quickly and support rescue activities using a WSN. They implemented a testing version and evaluated the performance via experiments with the intent of applying this technology to situations involving an outdoor public space, such as the case of a fire at a large public bus station.

Fire monitoring and fire detection are closely related topics because they involve the use of sensors to gather data and make decisions regarding the environmental conditions. The idea for using multiple fire parameters for fire detection is not a novel concept. However, using similar measurements beyond the detection of a fire was the basis for the current study. Multi-sensor/multi-criteria methods often include measurements of smoke, temperature, species concentrations, and possibly other signatures to determine the presence of a fire. Many researchers have focused on developing the logic used to detect the critical event of fire ignition [19–26]. For example, Chen et al. [26] developed a fire alarm algorithm which used rate-of-increase values to define the rate thresholds for detecting a fire based on measuring smoke, carbon monoxide, and carbon dioxide. All of these approaches use some combination of thresholds, rates-of-increase,

and additional logic, and a few employed neural networks for the fire-detection problem. New research using video monitoring has expanded the use of camera technology to aid in the detection problem as well [27–29].

For the current study, the research aimed to provide a building information modeling (BIM) –based visualization tool by integrating real-time fire monitoring data with three-dimensional models. Hajian and Becerik-Gerber [30] provide several inspiring points for the current work. In particular, the idea of updating the BIM with real-time data extends the application of BIM from construction stages into the occupancy stages of the building lifecycle in terms of continuity of use. Integrating real-time field data into BIM offers an exciting new opportunity for the Architecture, Engineering, and Construction (AEC) industry to use BIM beyond the construction phase [30].

Chen et al. [31] used real-time sensor data in BIM, where once again it was noted that current models in BIM are useful during construction but then are left static beyond the construction phase. Bringing BIM into use for dynamic evolution of the building status to present real-time building information could be beneficial to building managers during the lifecycle of the building, not just during construction, and then potentially for use by the responding fire department in the event of a fire. This particular case study by Chen et al. [31] focused on a de-icing system for a bridge deck; the real-time data was used to assess the state of the bridge deck under various climate conditions as the environment changed over time. Additionally, Chen et al. [32] discusses the concept of "bridging BIM and building" (BBB) and highlights a growing interest in the idea of bringing BIM into the building management phase as well. Furthermore, Rueppel and Stuebbe [33] presented a BIM-based navigation platform for emergencies in large buildings. In the extensive review by Volk, Stengel, and Schultmann [34], the use of BIM in the context of

building assessment, management, and even monitoring are mentioned. For example, several applications in the setting of monitoring and performance measurement through the use of sensors are presented. These examples serve as a motivation for the potential to bring fire monitoring into this application of BIM as well, specifically in the post-ignition fire state of the building.

The primary objective of the current study was to create a real-time fire monitoring system to integrate measured sensor data, simulation sub-models, and visualization tools into a comprehensive package for delivering actionable information to the responding fire department during a live fire event. This included the development of the foundation for a real-time fire monitoring software and its sub-components using a distributed architecture (in this case, two workstations in the research lab). In Fig. 5-2, the high-level architecture shows the three main components of the system: (1) data collection at the scene of the fire event, (2) data coordination and subsequent computation at the central computing workstation, and finally (3) visualization through the use of BIM and live graphs. The goal was to use multiple fire signatures which could be collected at the scene of the fire event. For testing the proposed system, sensor simulators for pushing live data to the fire-monitoring system were used. Thus, in the lab environment, the sensor simulator and computing workstation were hosted on one machine (Ubuntu 14.04) and the visualization components were hosted on a second machine (Windows 7) to provide this distributed system.

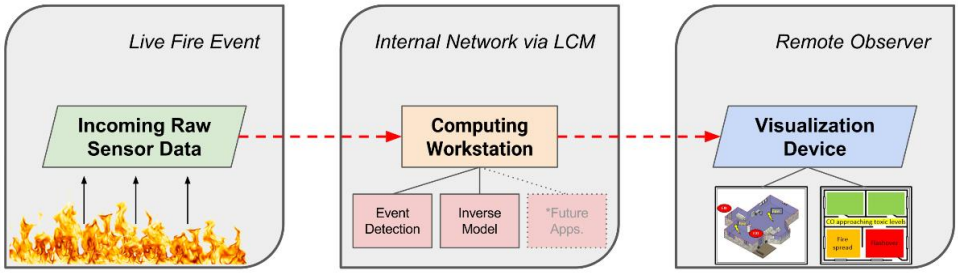


Figure 5-2: A three-component view of the major features used for real-time fire monitoring; data is collected within the building during the fire event and used for computation and visualization in the proposed system

The system was designed for the integration of sub-models into the system, such as a new event detection model presented here, and with the flexibility to include additional features such as an inverse fire model [35] in the future. This required the development of the event detection model for determining updates to the fire status based on measured data from sensors and computed data from the sub-models. The integration of these sub-models was performed such that the process can be replicated for the inclusion of future models, thus lending flexibility and extensibility. Communication between the simulated sensors, sub-models, and main computing controller was provided through the use of Lightweight Communication and Marshalling (LCM) [36], which handled data transfer between models through the use of message-passing. This work included the design of methods for visualization of measured and computed data that are easy to interpret and informative in showing the progression of the fire in a building in real-time. BIM was used to showcase the potential for using data visualization in the fire monitoring application, but only as a playback feature for the current study.

The distributed architecture was chosen in anticipation of the desired use-case: field data collected at the fire scene (sensors) followed by data coordination, simulation, and other computation performed at a stationary location (computing workstation), and finally remote visualization received by firefighter or incident commander (laptop or some other mobile device). In various studies, multiple fire signatures have been used for determining the existence of a fire in a building. Extending this concept to the post-ignition state and measuring features such as temperature and oxygen concentration during the fire event can provide us with predictions of the heat release rate and several other important fire parameters. The newly developed event detection model uses the measured fire signatures and computes useful information for the visualization device in real time. The visualization tool for showing the progression of the fire in the building

would allow the responding fire department to receive time-sensitive information about the fire as an easily readable graphical representation of the status of the fire.

5.2 LCM-Based Computing Infrastructure

The framework for wireless sensing in a fire rescue scenario presented by Lim et al. [18] details some of the most important requirements of a robust fire-monitoring system; specifically, they have noted fault-tolerant communication, low latency, remote management, and reuse of collected information as the main priorities for their system. These requirements are applicable to the present work as well. The integration of measurements from sensors with computing tools for real-time simulation and detection, and a scalable and efficient computing platform are also necessary. To this end, a distributed cyber-infrastructure customized for firefighter decision-making was developed to provide a computing framework employing the message-passing capabilities of the LCM library [36]; an overview the proposed system is illustrated in Fig. 5-3, which includes the first two nodes of Fig. 5-2 and excludes the visualization component.

Originally developed for robotics applications, LCM allows for low-latency message passing between applications, which, in the context of fire monitoring and visualization, would include connecting live data streams with various sub-models. In the current study, using simulated sensor data, each virtual sensor was connected to the main monitoring system through the use of individual data channels. LCM was used to provide this connection and to simulate the transmission of field-measured sensor data from the fire to an off-site server performing the computational work. On the right side of Fig. 5-3, the event detection model is shown as receiving data from the main fire monitoring program. Additionally, extra models can be connected in a

similar manner; the main program in the computing server controls the flow of data by directing new measurements from the sensors to the appropriate sub-models for further specific calculations.

At the core of the proposed system is the LCM message-passing network, which is a set of libraries that facilitate the interaction of various simulation modules through the use of consistent data structures. In LCM, messages are communicated between applications using these consistent data structures and communication is performed by transmission over pre-determined channels. For example, in Fig. 5-3, the various ribbons connecting each sensor to the central computing program are colored differently to visually represent these unique channels. Sensor *i* will only send messages on the `SENSOR<i>` channel and the main program will only expect to receive data from sensor *i* on this same `SENSOR<i>` channel by subscribing to the channel of each sensor and receiving new data whenever it is published by the sensors. The coordination of channel assignments and the publish/subscribe processes were all handled automatically in the system. This was provided by carefully establishing channels for the data to be transmitted between components of the system and determining in advance which data must be sent to each sub-model. This same approach of publish/subscribe is used to communicate between the main fire monitoring program and the sub-models as well.

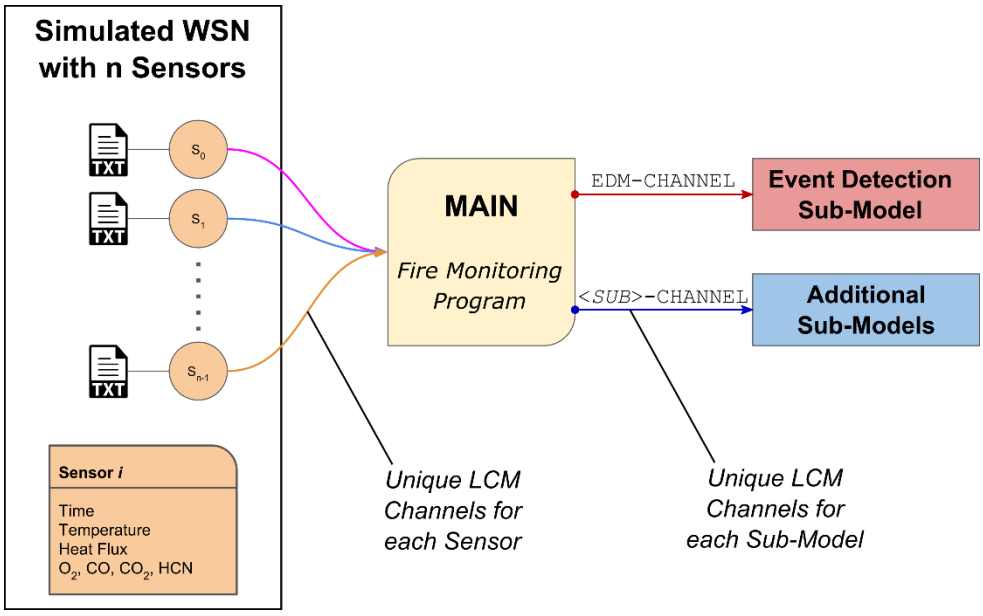


Figure 5-3: Overview of the proposed LCM-based monitoring system for real-time fire monitoring; the various colors of the links between different system components represent unique channels of communication in the LCM framework

One key feature of this system is the ability to use LCM with applications developed in various programming languages. For example, the main program for coordinating the computational work was implemented using Python and the event detection model is built from C++ source code. LCM provides the flexibility to use common data structures among these various components with ease: to employ a particular language-specific data structure, the necessary programming languages are supplied at compile time using unique compiler flags for each communicating component (a standard feature of the LCM library suite). Then, LCM compiles the agreed-upon-data structure for each language-specific application's use during the simulation. Since Python is not a compiled language, this step consists of LCM automatically preparing the corresponding class containing the necessary member attributes.

Thus, only one LCM file describing the contents of the data package in a C-style structure is needed for defining a common data type between two models built from dissimilar languages. For example, to send a scalar variable such as `temperature` from the main fire monitoring program

to the event detection model for a particular sensor, one LCM file template must be created declaring `temperature` as a double precision member variable of the structure (a C-style `struct`) named, for example, `send_to_edm` for sending data from the main program to the event detection model. Both the main program (Python) and the event detection model (C++) are able to access the contents of a message sent from one to the other which is of this data type (`send_to_edm`). This is accomplished using both the Python and C++ compiler flags to compile the C-style LCM file prior to the simulation.

In Fig. 5-3, the `Sensor` class is shown to contain the private variables associated with one particular sensor in the system. Simulation of the sensor data is described in the following section, but the information in Fig. 5-3 serves to show which physical quantities are related to each sensor. A corresponding LCM data file is necessary for transmitting the variables from the sensors to the main computing program and then on to the sub-models, as mentioned previously. The C-style `struct` used for this purpose is shown in Fig. 5-4, where this derived data type is used as the common interface between two different components (i.e., between the sensor simulator and the main program as well as between the main program and the event detection model later).

```
package sim_sensor;
struct sensor_data
{
  int16_t    sensorNum;
  double     sendTime;
  double     temperature;
  double     O2conc;
  double     COconc;
  double     CO2conc;
  double     HCNconc;
  double     heatFlux;
}
```

Figure 5-4: The LCM data structure used as a common data type between the main monitoring program and the event detection sub-model (file `sim_sensor.lcm`)

```
import lcm
import select
from sim_sensor import sensor_data
#============================================================
# FUNCTION TO HANDLE NEW INCOMING DATA
  def edm_handler(channel, data):
    msg = sensor_data.decode(data)
    newSensorData.sensorNum = msg.sensorNum
    newSensorData.heatFlux = msg.heatFlux
    # ... continue for all variables in the sim_sensor.lcm
#============================================================
# MAIN PROGRAM
  lc = lcm.LCM()
  newSensorData = sensor_data()

  # subscribe to all sensor channels
  for i in range(0, NUM_SENSORS):
    lc.subscribe("Sensor" + str(i), edm_handler)

  # main time loop
  while ("on"):
    # check if new msg was sent via LCM using "select" function
    rfds,wfds,efds = select.select([lc.fileno()],[],[],timeout)

    # if new msg received, then decode data and send to EDM:
    if rfds:
      lc.handle()
      lc.publish("EDM_CHANNEL", newSensorData.encode())
      # check if sensors are still "on"
```

Figure 5-5: Abbreviated version of the main fire monitoring program `main_rtfm.py`

Figures 5-5 and 5-6 serve to represent the basic components of the main fire monitoring program and the event detection model. The important component of these two abbreviated code samples is the `import` and `#include` statements in Fig. 5-5 and Fig. 5-6, respectively, which provides access to the LCM library and the compiled LCM data structure shown in Fig. 5-4. With this data type available to both applications, data between the two dissimilar programs (Python and C++) can be easily exchanged using LCM to publish and subscribe messages on specific channels. Additionally, both programs shown here rely on the `select` function to continuously monitor for incoming messages passed via LCM. This allows for nearly immediate processing of incoming data (for either the main program or the event detection model) as opposed to a polling approach in which each component requests new data at a pre-defined frequency. Data measured

from the sensors is stored in the private variables of the Sensor class objects in the event detection model such that when new data is available for sensor *i*, the variables are updated immediately upon receipt in this sub-model. The member functions for assessing the individual fire hazards are discussed in Section 5.4 and more details about the development full fire-monitoring system from a software engineering perspective can be found in Appendix A of the dissertation.

```
#include <sys/select.h>
#include <lcm/lcm-cpp.hpp>
#include "sim_sensor/sensor_data.hpp"
//=========================================================
#include "DataHandler.h"    // custom class to handle LCM data
#include "Sensor.h"         // custom class storing sensor data
//=========================================================
// (the following is inside the main EDM program)

// initialize Sensor class objects with ID number
Sensor sensorArray[NUM_SENSORS];
for (int i = 0; i < NUM_SENSORS; i++)
  sensorArray[i].setID(i);

// main time loop (exit criterion not shown)
while (1)
{
  // use "select" func. to check if new data arrived (not shown)
  if ("new_data")
  {
    lcm.handle();
    // update data for each sensor in Sensor class (not shown)

    // check each fire hazard at current sensor:
    i = currentData.getSensorNum();
    smokeToxicity[i] = sensorArray[i].checkSmokeTox();
    burnThreat[i] = sensorArray[i].checkBurnThreat();
    fireStatus[i] = sensorArray[i].checkFireStatus();

    // write results to csv output file (not shown)
  }
}
```

Figure 5-6: Abbreviated version of the main event detection model program `main_edm.cpp`

5.3 Data Collection for Fire Monitoring

There are several household devices which individually collect data on the status of the indoor environment: smoke detectors, carbon monoxide detectors, smart thermostats, and indoor

air quality sensors, to name a few. While not every home has all of these individual features and while there may not be an all-encompassing technology for collecting all these potential fire signatures and indicators in one device, yet the ability to monitor multiple features of the indoor climate in ambient conditions has increased within the last decade. Mainstream adoption of indoor environment-monitoring technology, such as the Nest Labs smart thermostat (www.nest.com) and the use of home security systems, appears to be a positive trend towards a future of increased monitoring capabilities. These features of the indoor environment related to comfort (ambient temperature), air quality (oxygen concentration, carbon dioxide, and other measurements of airborne species), and hazard (carbon monoxide, extreme temperature, smoke) may also be useful measurements for analyzing the evolving state of a fire during the post-ignition conditions in a structure. Many of these parameters could function as the input for a multiple-parameter fire-monitoring system intended for tracking the fire status as it progresses and potentially spreads among several rooms.

The specific sensor measurements used in the computational framework presented in this chapter are the gas temperature and radiative heat flux at the ceiling, as well as the species concentrations of oxygen (O_2), carbon monoxide (CO), carbon dioxide (CO_2), and hydrogen cyanide (HCN); these are the six fire signatures used in the following sections. The basis of the real-time fire monitoring system presented in the current study is the assumption of such a future sensing technology which is capable of monitoring several conditions of the indoor environment and treating the data as fire signatures, for example, these six mentioned here.

The challenges of sensor measurement in extreme conditions such as fire are a known reality and an expected obstacle to real-time fire monitoring. The accuracy of instruments may be affected in high-temperature scenarios. This difficulty is faced in the experimental setting with

thermocouples and flux gauges and some of the problems have been discussed in other works as well. For example, Silvani et al. [2] discusses the issues with data losses and time delays using WSN in a forest fire monitoring application. In their study, which employed 2010 mote technology in the field, data transmission problems (losses) for sensors near the flames were potentially caused by the microwaves at the fire source. However, a key feature of a robust WSN is the resilience property [2], which is the ability to reroute information to avoid damaged sensors. Such a feature would also be necessary to help mitigate these potential hardware problems in the building setting.

In many experimental tests involving small-scale or full-scale fires, researchers have the ability to use multiple devices for data collection including several thermocouple trees, heat flux sensors, infrared cameras, and a variety of other useful tools for measuring the key features of the fire experiments. For example, in the tests conducted by Cowlard et al. [1], over 400 sensors were used to instrument a compartment. While not every experimental test has this level of instrumentation, the reality is that experiments allow for the use of multiple sensors and data acquisition systems. For the research setting, this is necessary and expected; however, even with the growing use of smart technology in the home, the ability to monitor in a residential setting is currently limited to approximately one device per room. For example, a traditional rule of thumb is to have one smoke detector for each room of the house and some families may be able to afford an additional CO-detector in the living room or dining room.

Smoke alarms and some CO-detectors can be used to determine whether a fire has started while reducing the possibility for false alarms. For those devices, and others with similar functionality, the goal is to decide if a fire exists and then provide an alarm if necessary. The hazard timeline is an important consideration in the post-ignition fire state in a building. Initially, everything in the building is in a normal ambient condition and there should only be acceptable,

trace levels of the other species concentrations in the air. An alarm would sound and the fire department would be notified in the event of a fire being detected by existing sensors, which may be indicated by smoke or temperature or some other measurement.

The period of interest for the current study is the post-ignition state after the first fire alarm has already sounded. Figure 5-7 highlights some of the main features of the fire progression and the focus on the post-ignition fire state in this chapter. The work presented was not designed to detect the presence of a new fire, but rather to assess the rooms of the building during the fire for use by the firefighters in making decisions about where to act, when to evacuate, and how to avoid dangers and prioritize aid, in general.

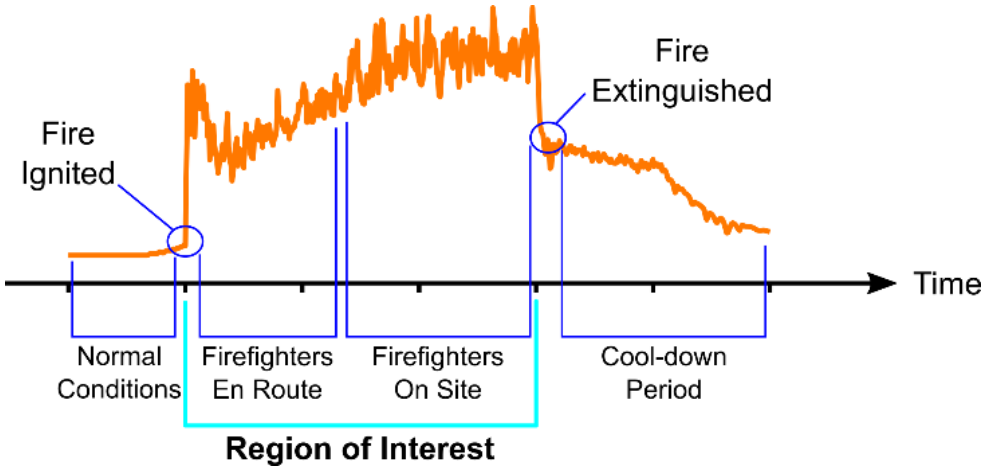


Figure 5-7: The timeline of progression of a fire in a building; the aim for the current work is to provide a monitoring tool for the post-ignition state

For the purposes of the current study, it was necessary to simulate the behavior of multiple sensors measuring data from their location in the test building to the main program. The *n*-sensor model was based on the assumption of the use of one sensor per room in all monitoring simulations, thus providing for the case of an *n*-room monitoring scenario. To send data at regular, but not perfectly timed, intervals, a simple sensor program was developed in C++ to map *n* sensors to *n* threads in a parallel manner. Using the OpenMP library for C++ [37] allows the simulation to

employ as many sensors as are needed for the multi-room monitoring case. The parallel directives of OpenMP were used to split the program into *n* similar threads, each responsible for reading from their designated data file. These data files were pre-populated with time-series data for simulating newly measured fire signatures. For the current monitoring purposes, a typical data file was created as `file<i>.csv`, where the `i` corresponds to the sensor/room number such that the thread with a process ID value of *i* should open, read, and send the data in the i^{th} data file. In order to use, for example, four rooms in a test of the monitoring system, it was necessary to prepare the four corresponding data files with six columns corresponding to each of the six fire signatures mentioned earlier. From a hardware standpoint, this simulation of data was strictly related to the physical values (temperature, heat flux, etc.) as opposed to sensor signals (voltage, etc.); therefore, there was no provision of data conversion, signal processing, or calibration needed for these idealized sensors.

In addition to providing a unique data set to each sensor, every thread runs independently to simulate asynchronous message passing to the main monitoring program; this main system using the LCM-based infrastructure was described in detail in the previous section. The sensor simulation was accomplished using "wait" function called `usleep` between every pushed message. All the sensors in the model have a specified nominal frequency, f_{nom}. However, if the nominal frequency was 1.0 Hz, instead of pushing data packages every 1.0 seconds, the wait function delayed the message by a random but bounded amount. In the following equations, f_{nom} and z were specified as input for the group of sensors. The resulting nominal period, P_{nom}, for sending messages was simply the reciprocal of the nominal frequency:

$$P_{nom} = \frac{1}{f_{nom}} \tag{5.1}$$

The delay bound, *z*, was specified as a percentage of the nominal period in order to compute a maximum and minimum possible delay in sending:

$$P_{\min} = (1.0 - z) \cdot P_{nom} \tag{5.2a}$$

$$P_{\max} = (1.0 + z) \cdot P_{nom} \tag{5.2b}$$

The waiting time, or the time between publishing two messages from the same sensor, was computed by taking a random number in the interval (P_{min}, P_{max}) from the uniform distribution. Specifying *z* as, say, 20% would lead to message-send waiting times within the interval of (0.80, 1.20) seconds for the nominal frequency of 1.0 Hz mentioned earlier. The random number is drawn from this interval for every time step of the simulation and is unique to each sensor (i.e., to each thread). This waiting scheme was used to intentionally slow down the sensor message-sending, since the program is able to read from the data files much faster than 1.0 Hz, for example.

The concept of asynchronous message sending is shown in Fig. 5-8, in which the nominal times and actual message times are illustrated. As part of the control in the main monitoring program, the user is able to launch the sensor-array from a simple menu before starting the simulation. Messages containing new data from the sensors may be sent either before or after the nominal time and in any order; they are all received and handled by the main program. Data from the simulated sensors was packaged and sent to the main program using LCM for message passing within the same machine: this is the lab-environment equivalent of measuring fire signatures and transmitting the data from a WSN in a real-world application. More information about the communication between sub-models is described in the following sections, as the details of how these asynchronous data packages are transferred from the sensor program to the main program is left out for now. The message passing from individual sensors to the main computing program represents a non-blocking send, meaning the sensor simulator does not wait for confirmation from

the recipient: data is pushed from the sensor immediately after its random delay described previously.

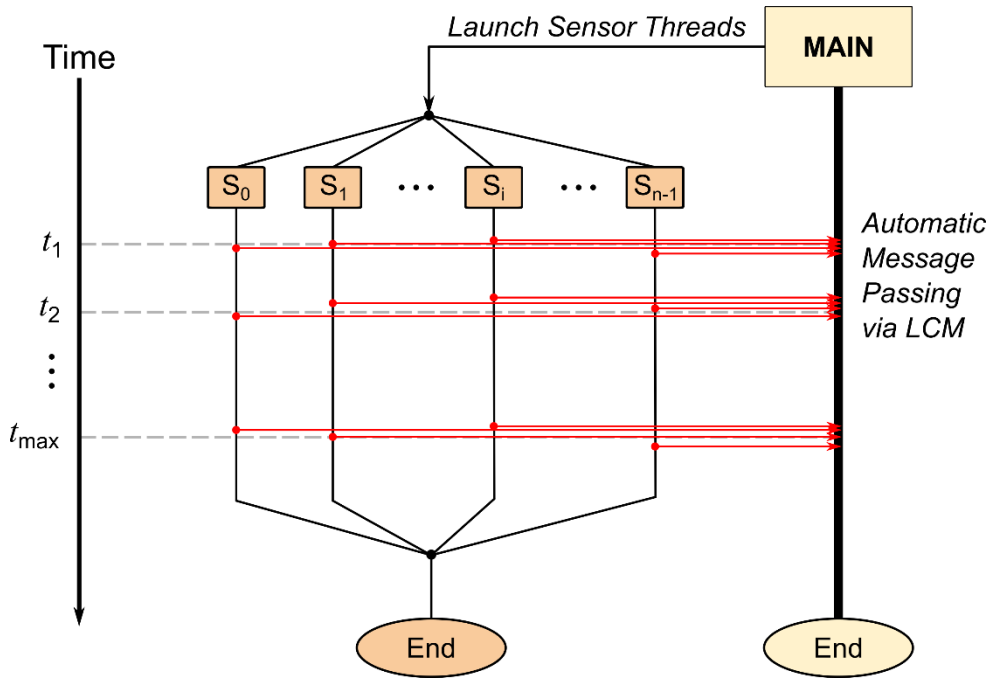


Figure 5-8: The multi-threaded model for simulating multiple sensors with nominal time steps listed on the timeline to the left; each sensor is mapped to its own thread which accesses its own unique data file to read and push new data to the main monitoring program using an LCM data structure

5.4 Event Detection Model

This component of the research aimed to provide methods for determining key fire events from measured data collected by sensors. Thus, one of the primary objectives was to use detection methods that are efficient and reliable such that key information may be extracted in real time. In the post-ignition monitoring setting, sensitivity to nuisance sources is less of a concern than determining real threats efficiently and early enough to provide sufficient lead time in decision making. To achieve this objective, deterministic event detection methods were developed based on existing data and practices used in fire safety to maximize the available knowledge about the

fire progression without hindering the necessary work of firefighters. Methods for event detection were developed and implemented to specifically detect the following three hazards:

1. Smoke toxicity based on the composition of gases
2. Burn threats to building occupants and firefighters
3. Fire status, namely, potential fire spread and impending flashover conditions

Using thresholds (i.e., set points and rates of rise) to detect fire events is a deterministic approach that is commonly used in fire detection. While there is a possibility for false alarms, thresholds are well-suited for instances in which a large enough dataset does not exist yet to allow neural networks to be formed (i.e., in the work by Jones [25]). Given the limited amount of data available during the developmental phases of the proposed monitoring system for training such a neural network algorithm, the present research focused on establishing set points and/or rates-of-rise for temperatures and species concentrations that indicate smoke toxicity, burn threats, and the measure the status of the fire. Additionally, methods from fire safety engineering were used in real-time computation to assess the progression of certain threats (smoke toxicity and burn threats, specifically). In particular, the Fraction Effective Dose (FED) was used for instantaneous assessment of smoke toxicity and burn threats in the monitoring setting [38]. Note that these methods serve as a form of live data evaluation for remote monitoring purposes as opposed to using the measurements for true fire forecasting.

The thresholds were established based on relevant literature and existing experimental data. The event detection methods used here were tested in simple verification cases as well as realistic simulated fires using Fire Dynamics Simulator (FDS) [39]. In this section, one verification case is presented, which uses sample data from the Society of Fire Protection Engineers (SFPE) Handbook [40]. The final example provided at the end of this chapter demonstrates the use of the

event detection model in a real time simulation and in conjunction with the other components of the system described throughout the chapter.

With previous technology developed for detecting the presence of fires through various fire alarm algorithms presented in the literature, there are existing common thresholds for the present monitoring needs highlighted by this study. Recall that the proposed monitoring system has been designed to measure the fire signatures of temperature [°C], heat flux [kW/m^2], and the concentrations of four gases: oxygen (O_2) [%], carbon monoxide (CO) [ppm], carbon dioxide (CO_2) [%], and hydrogen cyanide (HCN) [ppm]. The guiding principle for the development of an event detection model can be stated as follows: based on raw sensor data measuring these particular parameters, determine whether a threshold has been reached for either a raw data point (i.e., temperature at a particular time) or a computed value (i.e., the cumulative FED for smoke inhalation at a particular time) and assign a warning level for each hazard. From these numerical evaluations, we can establish logic for issuing warnings and providing feedback from the monitoring system to a visualization component as well.

These methods were intended to track the status of the fire as it progresses in the building. While many approaches have been used in the detection of new fire events, the methods used here aimed to extend detection into the post-ignition state such that the hazards mentioned be assessed as the fire evolves. The details of the event detection model follow for each of the three hazards. Additionally, the connection between the output of the event detection model and the visualization component will be described in Section 5.5 on visualization using BIM.

These methods were implemented as a C++ program in which each unique sensor was represented by its own object from the `Sensor` class. Each sensor (assuming one per room, as described in the previous section) is responsible for performing its own calculations to determine

if a particular hazard threshold has been reached. For example, at every instance that data is received in the event detection model, the `Sensor` object will automatically check its own private variables, say, to compute smoke toxicity, using a particular member function for each hazard and then return the warning level as an integer which is stored for output to the visualization device: `smokeWarning[i] = sensorArray[i].checkSmokeTox();` (for the sensor `i` in the simulation, located in room `i`). The other two member functions are `checkBurnThreats()` and `checkFireStatus()`; each hazard may return an integer warning from the set {0, 1, 2}, as described next, where each level corresponds to a different severity of effect.

5.4.1 Hazard 1: Smoke Toxicity

Smoke toxicity has been seen as the cause of death for building occupants on many occasions. One way that it is understood is in terms of the concentration and exposure time which leads to either incapacitation or death. Alaire [41] discusses, for example, oxygen depletion in terms of what was observed in blood samples from autopsies of fire victims. This concept of blood concentration is useful in quantifying the effects of exposure to harmful gases. For example, in the case of CO, four minutes of exposure results in about 40% carboxyhemoglobin (COHb) in the bloodstream leading to incapacitation due to the CO binding with the hemoglobin in the blood [40]. But for the purpose of real-time monitoring of smoke toxicity levels, it is necessary to translate this information into engineering parameters for tracking human exposure *during* the fire event: something that has been established through the use of FED in the past.

For the current study, toxicity levels were monitored using measurements of CO, CO_2, HCN, and O_2 (depletion of O_2) based on limits that are linked to smoke inhalation incapacitation and death. According to the study by Alarie [41], if oxygen levels in the room of origin reach less

than 7%, then this will become a primary cause of local incapacitation and death. However, such low oxygen levels usually accompany very hot smoke which would cause skin surface and systemic hyperthermia to become major factors as well. Low oxygen levels plus extreme heat are fast-acting and principal factors leading to incapacitation and death [41]. Since burn threats due to such high-intensity heating represents another hazard altogether, they were handled in the following subsection.

Due to the nature of the hazard of smoke inhalation, checking thresholds instantaneously does not sufficiently capture the time of exposure to such harmful gases. Thus, the Fractional Effective Dose (FED) methods were implemented to assess each room in the building for smoke hazards [38], but in real time for this application. FED has been used in other applications as well, including smoke visualization [42], path safety evaluation [43], fire simulation software such as FDS [39], evacuation planning based on GIS [44], and in research related to testing materials for smoke toxicity [45]. In recent years, Xu et al. [42] used FED in their definition of smoke hazard as part of a virtual reality tool using smoke visualization and evacuation pathfinding. FED has been used in various applications of fire safety and even visualization, but is not normally found in the real-time monitoring space.

While these expressions for FED that follow were designed for unprotected human exposure based on testing with rodents in the laboratory setting [38], there is still value in using these calculations for warning firefighters as well, even though personal protective equipment in a typical firefighting scenario can be expected. Using these FED indicators in real-time, though, can help provide assistance in assessing the smoke toxicity dangers with early warning, which could also improve decision-making for rescue applications of trapped building occupants or simply for avoiding severely toxic regions altogether if full evacuation was assumed.

The FED is typically used to determine the increasing threat due to exposure to toxic gases over time during a fire event. It is based on the concept of taking a summation of several consecutive, short, transient "exposure-doses" of harmful gases relative to known threatening levels of these gases. The basic equation for computing the FED is shown in Eq. (5.3) below:

$$FED = \sum_{i=1}^{n} \frac{C_i \cdot \Delta t_t}{(Ct)_{\max}} \tag{5.3}$$

In the equation, the summation includes the instantaneous concentration C_i and the short time interval Δt_i over which that concentration was measured. The denominator holds the limiting value of the total exposure-dose $(Ct)_{max}$ required to cause a reaction in 50% of the occupants. When the summation reaches a value of 1.0 at some point during the summation, the threshold has been crossed and the particular reaction is expected in 50% of the human population (where the reaction thresholds are typically incapacitation and death).

The SFPE Handbook [40] as well as the FDS User Guide [39] provide us with the details of the empirical formulas that are used to assess smoke toxicity: specifically for irritants using the Fractional Lethal Dose (FLD) and asphyxiants using the FED. The primary equation for assessing smoke toxicity using FED was presented in the SFPE Handbook as follows:

$$FED_{tot}(t_i) = (FED_{CO} + FED_{CN} + FED_{NO_x} + FLD_{irr}) \times V_{CO_2} + FED_{O_2} \tag{5.4}$$

The instantaneous FED at time t_i can include multiple gases, as indicated in Eq. (5.4) above. In particular, carbon monoxide (CO) was one of the gases considered:

$$FED_{CO} = (3.317 \times 10^{-5}) \cdot [CO]^{1.036} \cdot \frac{V}{D} \cdot \Delta t \tag{5.5}$$

The concentration of carbon monoxide is measured in units of ppm and is given by [CO]. The V term represents a breathing volume rate measured in units of L/min and has the following associated values: 8.5 L/min for resting or sleeping, 25 L/min for light work (which is typically

used and equivalent to walking to a fire exit), and 50 L/min which represents slow running or walking up a staircase [40]. The limitation is found in *D*, which represents the exposure-dose of COHb required for incapacitation (30% is default). Finally, the time step is defined as $\Delta t = t_i - t_{i-1}$ (in minutes), which is the difference between the last two time steps received by the event detection model in this case.

Additionally, the effects due to hydrogen cyanide (HCN) were also considered:

$$FED_{CN} = \frac{[CN]^{2.36}}{1.2 \times 10^6} \cdot \Delta t \tag{5.6}$$

This formula is used in the SFPE Handbook and differs slightly from the FDS version in the calculation of FED. In this equation, the direct concentration of hydrogen cyanide is not used. Instead, the concentrations of nitric oxide (NO) and nitrogen dioxide (NO_2) must be subtracted first: [CN] = [HCN] – [NO_2] – [NO] (all in units of ppm for this equation). Irritants such as NO_x gases, hydrochloric acid (HCl), hydrobromic acid (HBr), hydrogen flouride (HF), formaldehyde, and acrolein may also exist within buildings during real fire events and they are handled in the SFPE Handbook. However, in the current study, none of the NO_x gases nor any other irritants were measured or used in the simulations. As a result, [NO_2] = [NO] = 0 and consequently the value of FLD_{irr} in Eq. (5.4) is also zero.

Another important contribution in Eq. (5.4) is the hyperventilation factor, which serves to increase the occupants' potential consumption of toxic gases during extreme conditions experienced during the fire:

$$V_{CO_2} = \frac{\exp(0.1903 \cdot [CO_2] + 2.0004)}{7.1} \tag{5.7}$$

Here, carbon dioxide [CO_2] is passed into Eq. (5.7) in units of volume percentage. Similarly, oxygen depletion is handled in the final term of the FED equation given in Eq. (5.4):

$$FED_{O_2} = \frac{1.0}{\exp(8.13 - 0.54(20.9 - [O_2]))} \cdot \Delta t \tag{5.8}$$

Equations (5.4-5.8) were used in the event detection model to compute the FED of the smoke over time in the simulation.

Relating computed values to the warning levels for fire monitoring in this component of the model, FED > 1.0 corresponds to warning Level 1. If the oxygen percentage decreases to 7%, the warning is incremented by one as well (i.e., to Level 2). This limit of 7% is not checked in the FED (explicitly), but it is associated with rapid deterioration and death in previous fire events [41] and is thus included as another criteria for assessing the hazard level. Therefore the possible outcomes are warning levels in range of {0, 1, 2} as implemented here in the event detection model for smoke toxicity.

5.4.2 Hazard 2: Burn Threats

Burn threats are another concern for firefighters, even when shielded by personal protective equipment [46]. Burn threats were detected by monitoring critical temperatures and heat fluxes to bare skin. Additionally, extremely low oxygen levels usually accompany very hot smoke which would cause skin surface and systemic hyperthermia to become major factors as well [41]. Lawson [46] gives skin temperature values for human tolerance to burning, such as second degree burns occurring when the skin reaches 55°C. Also, skin is uncomfortable at 44°C and first-degree burns start at 48°C for reference. Naturally, there is a time component to this exposure as well, just as in the case of exposure to toxic gases. A 30-second exposure to 4.5 kW/m^2 will cause second-degree burns to human skin; thus, a person standing within 6 m of a 600-kW fire for 30 seconds would likely receive a second-degree burn [46]. Firefighters can potentially avoid burns by limiting their time in high thermal radiation environments.

Knowing that firefighters in the vicinity of hot flames can be affected, even if not present in the current room where the fire is located, there is a way to measure the "exposure-dose" to heat effects as well. Using the FED concept, burn threats may be assessed in real time using measured temperatures and heat fluxes [40, 42]. There exist empirical formulas for assessing convection and radiation effects based on the exposure-dose concept used for smoke in the preceding section, now applied to heating and burn effects for the current hazard assessment.

In the most recent SFPE Handbook [40], the equations for time of tolerance and time to fatal injury are provided for convection heating as follows.

$$t_{conv}^{pain} = (2\times10^{31})\cdot T^{-16.963} + (4\times10^{8})\cdot T^{-3.7561} \tag{5.9a}$$

$$t_{conv}^{fatal} = (2\times10^{18})\cdot T^{-9.0403} + (1\times10^{8})\cdot T^{-3.10898} \tag{5.9b}$$

From either of these equations (representing different levels of severity), we can obtain the time to pain or time to fatality, t_{conv}, due to heating by convection at a particular instance during the fire event. From this, we can compute the instantaneous FED_{conv} as $\Delta t / t_{conv}$ and add it to the sum, seen later in an equation to follow. For radiation effects, when the radiative heat flux is greater than 2.5 kW/m^2, the equation is as follows:

$$t_{rad}^{pain} = \frac{1.33}{q^{1.33}} \tag{5.10a}$$

$$t_{rad}^{fatal} = \frac{16.7}{q^{1.33}} \tag{5.10b}$$

If the heat flux is less than 2.5 kW/m^2, then the expected time to pain or fatality for radiative heating, t_{rad}, is greater than 30 minutes. In the general form of Eq. (5.10), the numerator is represented by a constant value: r. There are various values of the r constant for different levels of severity [40]. The values of 1.33 and 16.7 seen in the numerators of the Eq. (5.10) represent two limits of pain tolerance (1.33) and third-degree burn (16.7) thresholds. These two values were

selected as the lower and upper limits, respectively, of the monitoring system for issuing visual warnings in accordance with the empirical equations.

The combined effects of convective and radiative heating are seen in the FED equation for heat exposure:

$$FED_{heat} = \sum_{i=1}^{n} \left(\frac{1}{t_{conv}} + \frac{1}{t_{rad}} \right) \cdot \Delta t_i \tag{5.11}$$

The calculation in Eq. (5.11) was performed for each of the two levels, namely pain and fatality, from Eqs. (5.9-5.10) earlier. To translate these computed values into the desired warnings at an instant in time, if the pain tolerance version of the cumulative FED value in Eq. (5.11) reaches 1.0, then the warning is incremented by one. If the next threshold for fatality is reached, the warning is incremented by one again. In this approach, the lower bound pain tolerance threshold will be crossed first and then the more severe fatality threshold may be reached later as the conditions worsen for each individual room. The possible outcomes for warning levels are thus {0, 1, 2} for the burn threat model at any given time of the monitoring scenario.

5.4.3 Hazard 3: Fire Status

The goal for defining this hazard was to determine the fire status based on measurements of temperature and heat flux from the sensors and to assess whether flashover is impending. Naturally, fire spread and flashover are two main concerns in firefighting [47–50]. Flashover has been characterized from temperature and heat flux data (as shown in previous research [47, 51]) obtained from several experimental studies.

Recent work by Jones [25] presented a study using neural networks for determining the presence of fires as the main focus, but it also provides information on some traditional thresholds used for detection as well. In particular, typical values of set points and rate-of-rise criteria for

determining the presence of a fire (using single-point measurements) were provided. While many multi-criteria algorithms exist for detecting the presence of a fire in a building, it may not be appropriate to use measurements of species concentrations, for example carbon monoxide, as a fire-status indicator during the post-ignition state due to the potential for increased amounts of toxic combustion gases in the building atmosphere during this period. Thus, in order to assess the fire status in each room of the simulation, sensor measurements of temperature and radiative heat flux were isolated for this purpose.

The first warning level for the fire status hazard corresponds to threshold values of 57°C for the temperature and 0.6 kW/m^2 for the heat flux measured at the ceiling. The temperature limit was adapted from Jones' [25] comparisons with traditional threshold values for fire detection. The heat flux limiting value was mentioned by Guillaume [52], where this value was the minimum magnitude that could be recorded by the sensors used in the experiments of that study. The reason for using a heat flux value in determining the potential for fire spread is that checking the temperature alone could lead to a detection method that is too sensitive to hot upper layer gases accumulating in rooms away from the fire origin. The heat flux measurement combined with the temperature check is intended to help better identify locations where a fire may be present, potentially away from the room of origin, instead of just finding hot gases (which are handled by the burn threats hazard computations). These two threshold values for temperature and heat flux represent the conditions for a Level 1 warning in the fire status hazard.

Traditional practical criteria for determining flashover in an experimental test usually includes a radiative heat flux of 20 kW/m^2 measured at floor level and a temperature just below the ceiling of 600°C [51]. Flames coming out of open windows and doors is another experimental observation in flashover [53]. In studies on flashover, experimental test results from several

independent researchers have been collected to arrive at acceptable thresholds for engineering purposes; specifically, the heat flux values exceeding 20 kW/m^2 at the floor of the compartment and upper gas layer temperatures greater than 600°C near the ceiling are common values used to define the onset of flashover [47, 51, 53, 54].

The literature review provided by Liang, Chow, and Liu [53] also gives a good overview of additional studies which have found flashover conditions occurring with some wide ranges of temperatures and heat fluxes as well. While both of the related studies [47, 51] mention the considerable amount of scatter in the test results as well, these values for the heat flux and gas temperature thresholds were considered representative of the majority of the experimental studies reviewed. While some research has cautioned the use of these criteria in the fire monitoring application [1], the value offered by using threshold checks of these two flashover indicators (temperature and heat flux) is still reasonable in conjunction with the additional information obtained from the two other hazard models regarding the progression of the fire over time.

In order to provide potentially earlier warnings of impending flashover, these traditional threshold values of 20 kW/m^2 at floor level and temperature just below the ceiling of 600°C were reduced to take into consideration the flashover transition period [55]: the transition period may be considered in the range of 500-600°C and 15-20 kW/m^2. The range of 500-600 kW/m^2 has been used in other recent reports defining flashover as well [56]. As a result, the thresholds used in the event detection model for the second warning level were reduced to the lower bounds of 500°C and 15 kW/m^2 measured at the location of each sensor. These criteria provided the definition for a Level 2 warning in the fire status hazard. In the tests presented in this current study, the sensor measurements were located on the ceiling (as opposed to the floor for the heat flux measurement).

The ceiling location was chosen because this is the more likely location to find an existing detector, such as a smoke alarm.

5.4.4 Summary and Testing

The information in the above subsections on the individual hazards has been collected into one table for convenience. Each of the hazards and the corresponding warning levels is presented in Table 5-1. The conditions in Table 5-1 were implemented in the event detection model as individual member functions for each sensor (one per room) in the simulation. For each hazard, the warning levels of {0, 1, 2} was tracked throughout the complete time history of the monitoring scenario and written to an output file for post-processing and analysis. Additionally, the FED values were also printed to the output files.

Table 5-1: Thresholds and conditions used to define the event detection model

Hazard	Level 0	Level 1	Level 2
1. Smoke Toxicity	Normal Air	Incapacitation $FED_{smoke} > 1.0$	No Oxygen (Fatal) $O_2 < 7\%$
2. Burns to Skin	No Threat	Severe Pain $FED_{pain} > 1.0$	Third-Degree (Fatal) $FED_{fatal} > 1.0$
3. Fire Status	Near Ambient	Potential Fire 57°C and 0.6 kW/m^2	Flashover (Fatal) 500°C and 15 kW/m^2

In order to test the implementation, data from the example in the SFPE Handbook [40] for computing FED was used to compare with the real-time FED results in the current study, as well as to demonstrate the calculation of warning levels for a simple example problem. Results are shown here for a single-room application in which the data provided by the example [40] was used as input for the event-detection model. Table 5-2 shows the input (temperature, flux, and species) and then Table 5-3 follows with the computed output (FED and warning levels).

Table 5-2: Temperature, heat flux, and species concentration input for the event detection model test based on the SFPE Handbook example [40]

Time [sec]	Temp. [°C]	O_2 [%]	CO [ppm]	CO_2 [%]	HCN [ppm]	Flux [kW/m^2]
60	20	20.9%	0	0.0%	0	0.0
120	65	20.9%	0	0.0%	0	0.1
180	125	19.0%	500	1.5%	50	0.4
240	220	17.5%	2000	3.5%	150	1.0
300	405	15.0%	3500	6.0%	250	2.5
360	405	12.0%	6000	8.0%	300	2.5

Table 5-3: Results computed from the event detection model for the example in the SFPE Handbook [40]

Time	FED_{smoke}	$FED_{heat(pain)}$	$FED_{heat(fatal)}$	Smoke Toxicity	Burn Threats	Fire Status
60	0.00	0.00	0.00	0	0	0
120	0.00	0.02	0.00	0	0	0
180	0.04	0.20	0.04	0	0	0
240	0.41	1.77	0.23	0	1	1
300	2.11	19.87	1.71	1	2	1
360	6.17	37.97	3.19	1	2	1

For comparison with the SFPE Handbook results for the FED calculations, specifically Table 63.22 in that document, one might notice the slight difference in the FED for smoke toxicity presented in Table 5-3 of the current study. The difference arises from not including irritant gases (HCl, acrolein, and formaldehyde) in the current event detection model. All other differences are due to the small round-off errors encountered when performing these calculations by hand, as demonstrated in the SFPE Handbook example case, versus allowing the computer to store the FED as a floating-point number in the current real-time application of these methods.

5.5 Visualization using BIM

The visualization component was developed to present the results of the sub-models in a manner that could be useful for firefighters in monitoring the status of the fire in real time. The system was desired to include a component within a Bentley BIM product that will allow the user

to visualize live information from the fire event (specifically, in real time). It was envisioned to transmit the live information to an incident commander who is using the visualization interface at the scene of the fire event and it should transmit this information wirelessly. The system was designed to use a 3D representation of the building environment with transmitted information included in this BIM model.

The reality of the implementation was an approach that resulted in the ability to play back discrete events from the fire monitoring simulation in a BIM environment *a posteriori* to demonstrate the ability of the sub-models to deliver time-sensitive information as visual representations of the evolving fire scene. This project used Bentley AECOsim Building Designer (ABD; www.bentley.com) as the primary visualization software for this task. Due to the some of the constraints in the ABD software, the BIM visualization had to be handled as a post-processing feature, as opposed to a real-time component of the monitoring system. The use of BIM in a real-time manner is an extension of the original purpose of this software and thus such a limitation is simply a reflection of the current intended use of such tools: it is not intended for transient, real-time data applications. A clear distinction between the automated real-time components and the post-processing demonstration will follow.

The full monitoring system was specifically designed for distributed deployment. In particular, the computing resources to be used for simulation, calculation, and data management were hosted on one machine dedicated to this purpose alone (Ubuntu 14.04). This single machine was responsible for running the main monitoring program, which received new data from the simulated sensors over time and coordinated data transfer to the various sub-models. The visualization features presented here were hosted on a second machine (Windows 7). Information needed to be transmitted from the remote computing workstation to the local visualization device.

The system used the newly developed event-detection model to produce visualization-specific output intended for helping to identify fire hazards. This visualization-specific output was generated automatically in the monitoring system contained in the computing workstation and will be described next. Communication between the two machines is also discussed further.

Real-time data visualization techniques must be adapted to allow users to rapidly interpret critical information and evaluate alternative courses of action. Identifying fire hazards through the use of visual warnings displayed in the BIM environment could provide early warning for impending dangers as the severity of the conditions in the building are reflected in the codified warnings programmed into the visualization device. Thus, from the work completed in the development of the event detection model, it was necessary to link the computed results to visual ques relating numerical data to a graphical representation for a real-time setting. To connect computed values from the fire monitoring system to the visualization component, a post-processing step was added.

The LCM-based monitoring system described in the previous section included a function in the event detection model to write time history data for each sensor to its own output file in a standard way. The output from the event detection model included the measured temperatures, fluxes, and species concentrations received from each individual sensor in the event detection model (possibly at a different frequency than the sensor data generation) as well as the computed values of FED and the corresponding warning levels. For the purpose of visualization, the warning levels {0, 1, 2} for each of the three hazards discussed in the section on event detection were used to generate particular visual indicators in the BIM environment. In general, for each hazard, Level 0 is associated with no danger or concern, Level 1 indicates a potentially non-lethal threat, and

Level 2, the most severe, indicating immediate danger and potential fatality. These levels were defined in Table 5-1 previously.

Each of the warning levels {0, 1, 2} has standalone significance in assessing the danger in a particular compartment. However, to avoid a problem of information overload, additional logic was provided to combine the resulting warning levels into an aggregated *threat level*. With three hazards and three warning levels, a potential of 27 threat combinations arises: [0-1-1], [1-2-1], [1-1-0], etc. The logic proposed here serves to reduce these combinations to four main scenarios and three particular threat levels: warning (yellow), danger (orange), and severe (red). The particular rules used to decipher computed results follow.

First, if any *one* hazard (smoke, burns, or fire) reaches Level 1, for example the cases [1-0-0] or [0-1-0] or [0-0-1], then the threat is considered a "warning" (yellow). Second, if any *two* hazards reach Level 1, for example the cases [1-1-0] or [1-0-1] or [0-1-1], then the threat increases to "danger" (orange). Third, if all *three* hazards reach Level 1, specifically the case [1-1-1], then the threat increases to "severe" (red). Fourth, and finally, if any *one* hazard reaches Level 2, then the threat is escalated to "severe" (red) immediately for that room. Since the hazards at Level 2 are associated with fatality and flashover, as shown in Table 5-1, this hazard condition is treated with immediate progression to the severe threat level.

A Python script was developed for this post-processing step. It was employed after the simulation of the monitoring scenario in order to develop an example of using BIM with monitored data. It was used to translate the event detection model output, namely the hazard levels, into three new threat levels for every time step in the history of the fire monitoring scenario. Next, the Python script converted this integer-based data for the threat levels {0, 1, 2, 3} (corresponding to "initial/ambient" for 0, "warning", "danger", and "severe" threats, respectively) into a compatible

format for use in the BIM setting. Specifically, schedules of tasks were automatically generated from the threat-level data in the form of XML files that could be imported into Bentley ABD directly for visualization as schedule simulation. An overview of this process is shown in Fig. 5-9 for the relevant components of the system.

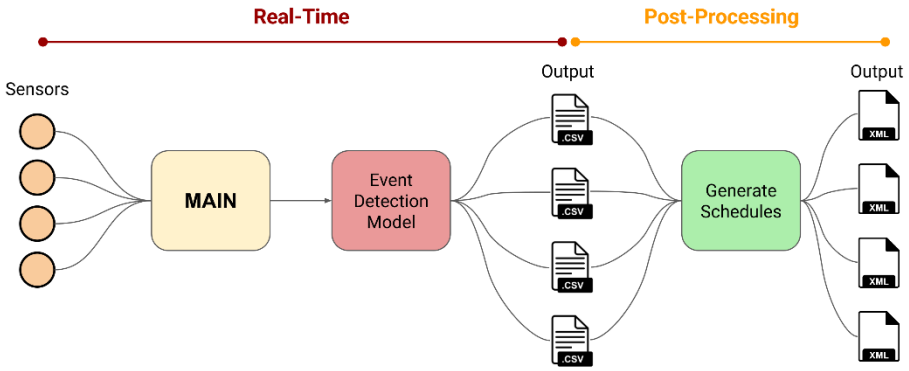


Figure 5-9: The post-processing step was included for use after fire monitoring in order to create XML-based schedules for animation in ABD; the tasks in the schedule were automatically created using start and finish times of each threat level based on the event detection model output

The playback for visualization used the Bentley animation producer for schedule simulation. Each of the threat levels were linked to elements in the BIM which would change color based on the current threat level. In particular, the floor area of each room was used as the dynamic material for visualizing changes in the fire-monitoring scenario (Fig. 5-10). The threat levels were translated to tasks in the XML schedules; this was accomplished by determining the start and finish times of particular threats (warning, danger, and severe) and then assigning the appropriate corresponding color value (yellow, orange, and red, respectively); start of a higher threat level indicated the end of the lower level. The main benefit of this method is the ability to automatically generate the XML-based schedules for each room in the simulation. Included in the automation is the determination of the start and finish times of the threats, the assignment of the proper material colors for those events, and the appropriate formatting for XML compatibility accomplished through the use of the Python package `lxml` to create event trees.

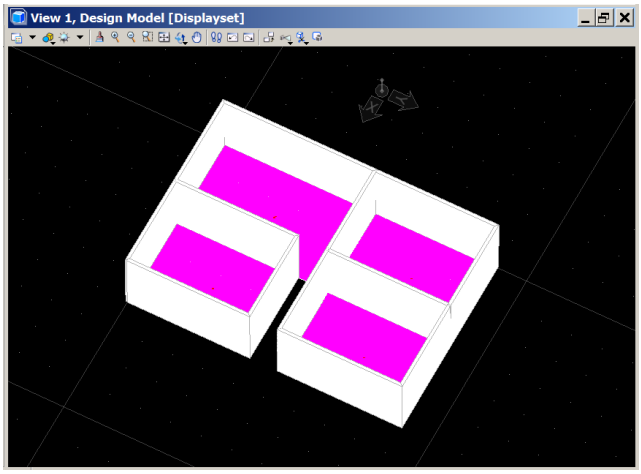


Figure 5-10: Screenshot of the 3D BIM exterior walls and colored floor plates, highlighted here in magenta, used for visualizing threat levels; tasks in the XML schedules were tied to the magenta-colored floor elements

For example, the four rooms in Fig. 5-10 required four unique XML schedules containing the task information. It was then possible to import the files directly into the Bentley ABD animation producer. In Fig. 5-11, the imported schedules can be seen as the list of tasks that appear in the *Schedule* tree. While the XML tasks can be generated automatically by the process described above and shown in Fig. 5-9, one step of manual intervention was needed to enable this visualization in ABD. Specifically, the schedules in ABD can only contain information about the tasks themselves: start time, finish time, start color, finish color, unique identification numbers, task names, etc. However, no information about elements that exist in the BIM can be included in the XML-based schedule, i.e., the schedule is not aware of any of the physical components in the model. Thus, once all four schedules were imported, it was necessary to link each task (initial, warning, danger, and severe) to the proper room element, namely the floor plates shown previously in Fig. 5-10. While this manual intervention is not attractive from an automation standpoint, it at least allows the exploration into the use of real-time computed results in BIM environments.

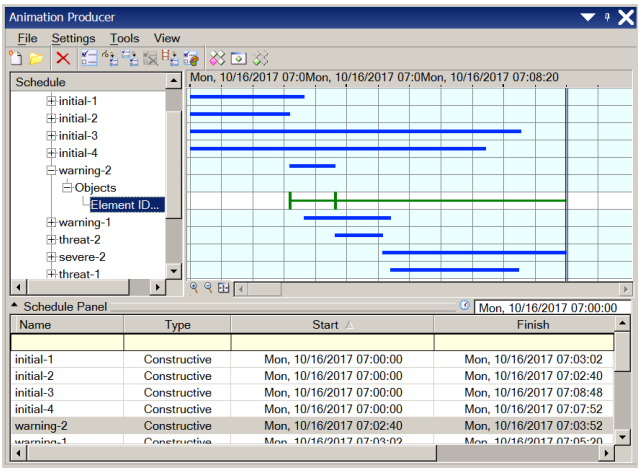


Figure 5-11: Screenshot of the Animation Producer in Bentley ABD with the imported XML tasks present in the *Schedule* tree

An example of some of the details for a "warning" task are shown in Fig. 5-12, where the start and finish times are provided as well as the color codes corresponding to each time. The four possible tasks are represented as different color floor plates in that figure as well. The additional task "initial" simply serves to bring the color-changing floor plates into existence for use with the actual threat warnings that follow.

The incident commander and active firefighters were the intended recipients of the information via the BIM visualization module. The system aimed to present visualized data in order to facilitate rapid decision making during a real firefighting operation, as mentioned in the goals of this study. This implies the need to select only important information for such a decision-making process as this system cannot make decisions for the users but rather indicate areas of importance based on the measured data and sub-model calculations presented in the previous sections. This system and others like it should present decision-making information in a way that potentially helps the incident commander in strategizing firefighting operations while not complicating operations with an abundance of data and distractions.

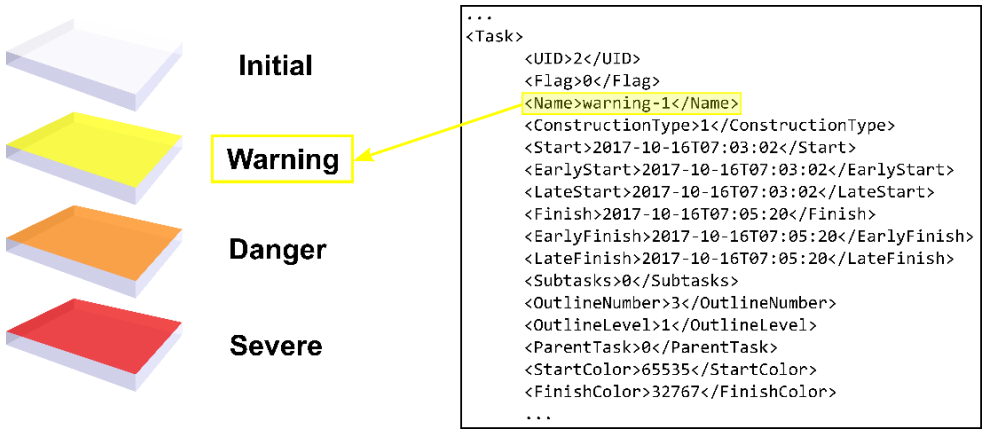


Figure 5-12: The four possible tasks are shown on the left with their corresponding colors; the actual representation of a "warning" task is shown to the right for Room 1 in XML format, where the start/finish times and colors were automatically generated from output generated by the event detection model

In addition to the playback showing a visualization of the event detection, a truly real-time feature was added that provides room-by-room graphs of the changing fire parameters. In particular, the changing temperature, heat flux, carbon monoxide, and oxygen levels were transmitted from the monitoring system to the visualization device using a local network. The live graph tool is launched at the beginning of the fire event on the visualization device. A connection with the computing device was established using a local network provided by the interoperable open-source software called http-server (https://github.com/indexzero/http-server). The http-server program was used to create a local server for each computer and the IP address and port of each temporary server was used to provide a link between the two machines using different operating systems. This provided a live connection between the Ubuntu-based computing workstation and the Windows-based visualization device.

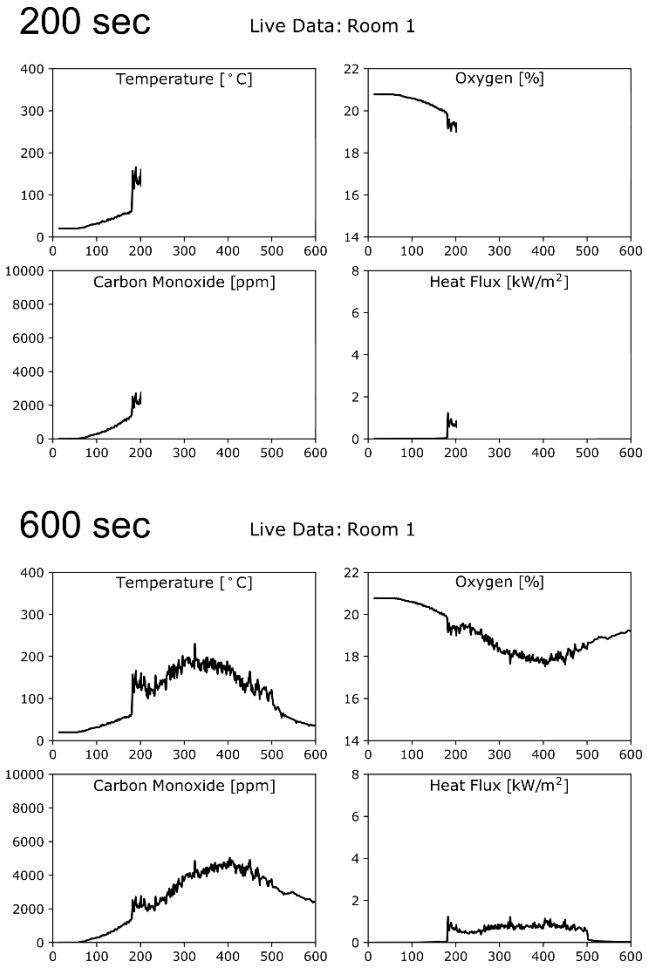


Figure 5-13: Sample live data captured and plotted in real-time automatically in the monitoring simulation; these plots were generated on the visualization device (Windows) while the incoming data was being processed on the computing workstation (Ubuntu)

Using this connection, the visualization device automatically checked the main computing workstation for the most recent data updates and then the downloaded information was appended to a text file on the visualization end. The live plotting tool continuously checks this file for changes and plots the updated time series data for each room in the model in real time. An example of the live plots of results is shown in Fig. 5-13 to demonstrate a sample of this tool's ability to plot updated information without user intervention in real time (for one room as a demonstration).

5.6 System Testing with Simulated Fire

To demonstrate the real-time monitoring system with a natural fire case, an FDS simulation was used for the four-room model seen in Fig. 5-10 in the BIM setting. The model represents a multi-room apartment with four main rooms and one hallway connecting them. The floor plan is shown in Fig. 5-14, which also identifies the fire locations and magnitudes as well as the sensor locations (one sensor per room). The total duration of the FDS fire model was 600 seconds (10.0 minutes). Room 1 was 3×5 m and Rooms 2-4 were 3×4 m in the plan dimensions; the ceiling height was 2.5 m above the ground in the model

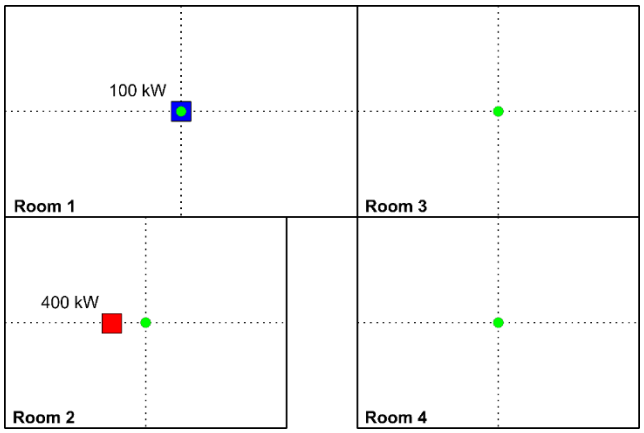


Figure 5-14: The floor plan of the fire simulation is shown with the locations of the two fires (blue square in Room 1 and red square in Room 2); the sensor measurements used in this test were recorded at the centers of each room, specifically at the ceiling level, and are marked with green circles in the image

In Room 1, the fire was designed as a 100-kW propane fire in FDS and placed directly in the center of the room. The fire area was 40×40 cm and the ignition of the fire was intentionally delayed and then ramped from a magnitude of 0.0 to 1.0 over one second, starting at 180 seconds (3.0 minutes) into the simulation. Using the fire ramp in FDS, this magnitude was held constant at 1.0 until 500 seconds (8.3 minutes) where it was reduced to 0.0 again until the end. In Room 2, the fire was slightly off center (50 cm from center), as seen in Fig. 5-14 as well. This fire had a peak heat release rate of 400 kW, used propane as the fuel, and also was 40×40 cm in size. The fire in Room 2 followed a *t*-squared fire curve with a slow growth factor, reaching a maximum magnitude

at 262 seconds (4.4 minutes) and held at that level until allowed to burn out at 400 seconds (6.7 minutes). Additionally, the carbon monoxide yield was set to 0.50 and the soot yield to 0.01 for both fires.

The output recorded at the four sensors (one for each room) was captured using point devices in FDS. Specifically, the gas temperature, radiative heat flux, and four species concentrations (O_2, CO, CO_2, and HCN) were measured at the ceiling at each of the device locations, totaling six fire signatures per room. Since the fuel source was propane and did not contain some combination of nitrogen and hydrogen, the resulting combustion products did not include HCN. As a result, the HCN measurements in the following output were zero for all times in the simulation. Data was written to the device output file at 1.0-second intervals for the duration of the fire simulation (10.0 minutes); this was only noted as a reminder that the output of the FDS fire simulation was used directly as input for the sensor simulation to read and push to the main computing system. Additionally, this fire simulation was conducted *a priori* in order to produce the time-series data that could be fed into the fire-monitoring system next in real-time, thus eliminating the delay caused by running the CFD analysis. The measurements for the six fire signatures are shown in Figs. 5-15 through 5-17, representing the output of the FDS simulation that was used as input to the fire-monitoring simulation. Note that the concentration [HCN] was zero for all measurements because the fuel source did not contain Nitrogen.

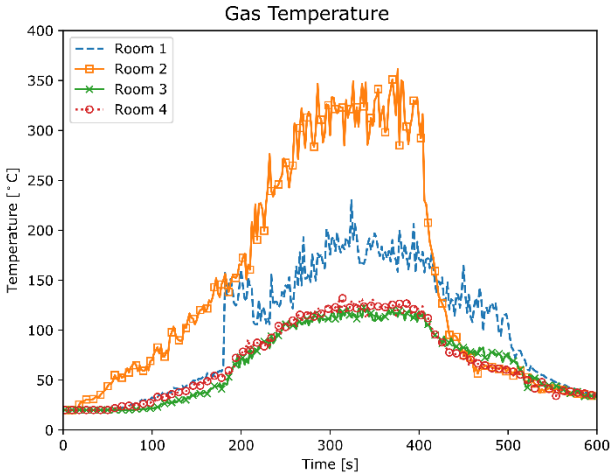


Figure 5-15: Time history of the gas temperature output from the FDS fire simulation for the four-room model; subsequently used as input for the fire monitoring system

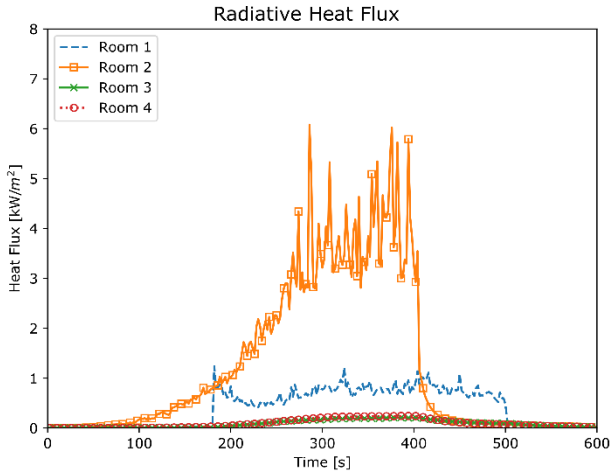


Figure 5-16: Time history of the radiative heat flux output from the FDS fire simulation for the four-room model; subsequently used as input for the fire monitoring system

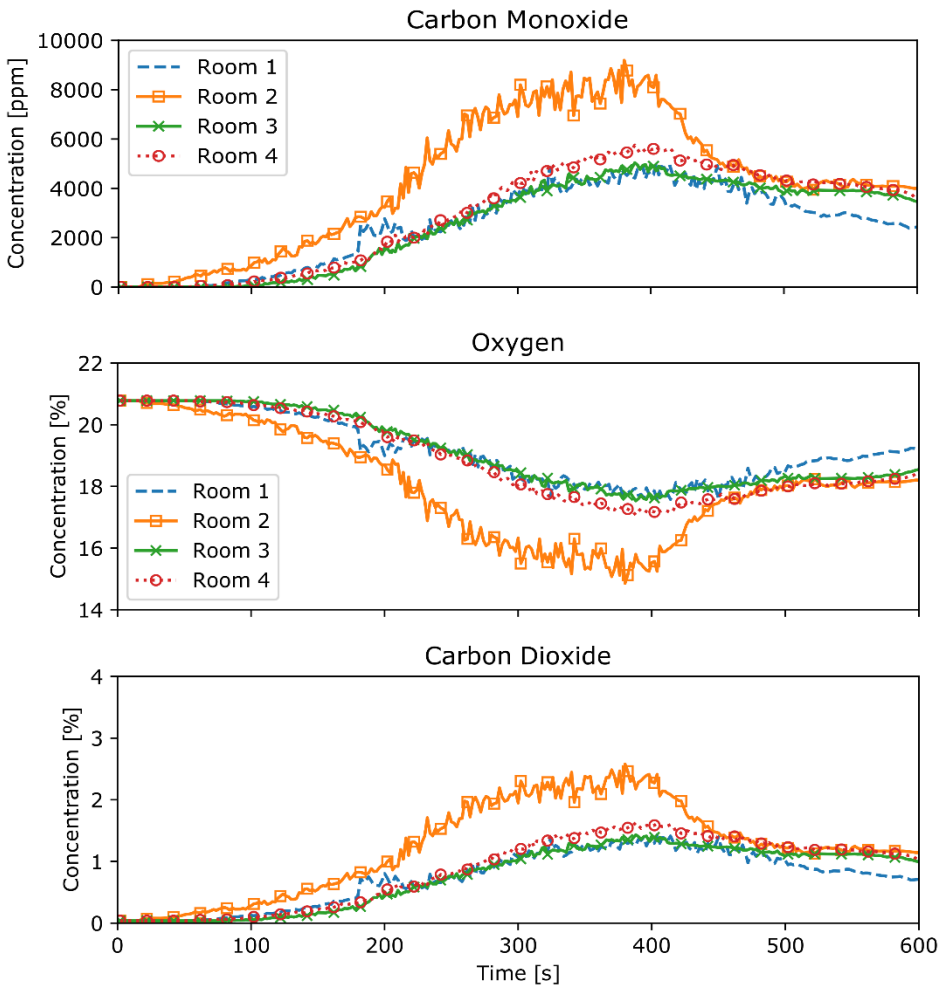


Figure 5-17: Time history of the concentrations of carbon monoxide, oxygen, and carbon dioxide output from the FDS fire simulation for the four-room model; subsequently used as input for the fire monitoring system

The data presented in Figs. 5-15 through 5-17 was used as input for the fire monitoring system. Specifically, a Python code was used to interpret the FDS device output file (`fire_devc.csv`) and automatically create from that output four separate data files for the sensors to use. In the section of this chapter on data collection and sensor simulation, it was shown in Figs. 5-3 and 5-8 that each sensor needs its own unique data file for its own unique thread to read and push packets of data to the main computing system. For the test of the system, the unique data files were created directly from the FDS output. The sensor simulation used a nominal frequency of 1.0 Hz with a potential noise range of ±10%; thus, each sensor would send its new data packet every

0.9 to 1.1 seconds to the main program. The nominal frequency chosen for the main program to push new data to the event detection model was 0.5 Hz, meaning new data was sent to the event detection model for processing every 2.0 seconds in real time.

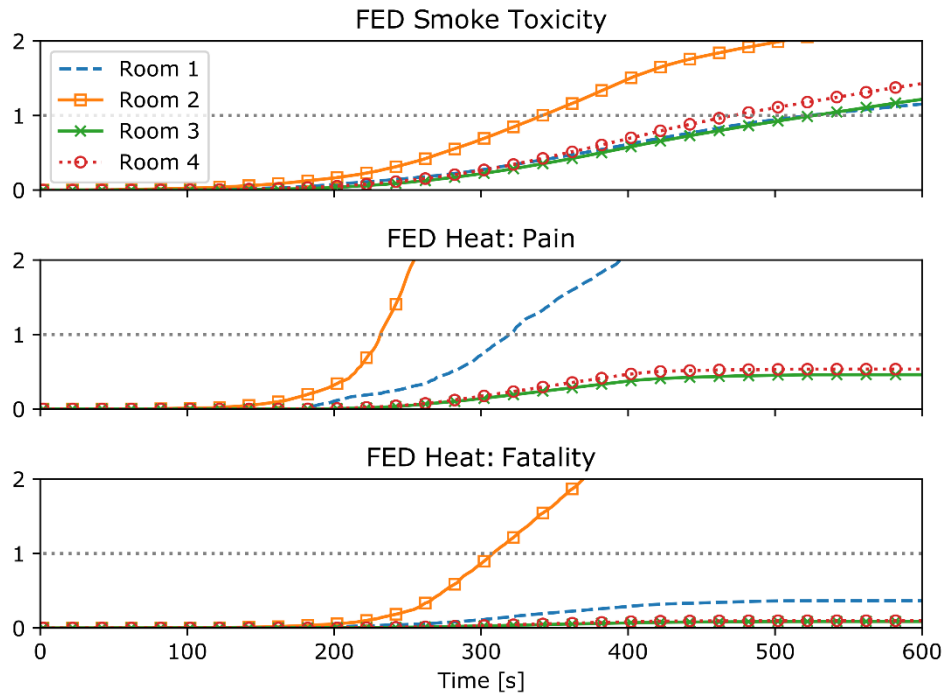


Figure 5-18: Results of the real-time FED computation for smoke toxicity and burn threats due to heat; results were computed using measurements at one sensor location per room every 2.0 seconds as received by the event detection model for real-time calculation

Data was passed from the sensors to the main program and then to the event detection model, as shown in Fig. 5-3 previously. The event detection model was responsible for computing the warning levels {0, 1, 2} described previously for each of the three hazards. For presentation purposes, the time-history of FED throughout the monitoring scenario is given here. Specifically, results are shown in Fig. 5-18 for the FED calculations for both smoke and heat (pain and fatality thresholds) for the duration of the test and for all four rooms. The threshold line for FED = 1.0 is also included in Fig. 5-18 for reference. It can be seen that the warning flag will be triggered at the time when the FED crosses this 1.0 limit.

A view of the FDS model at 201 seconds into the simulation is shown in Fig. 5-19 to show the progression of the fire at a particular instant. At this time in the monitoring scenario, the warning level for the fire status check has just reached Level 1 for Rooms 1 and 2. The FED was only part of the real-time calculation: warning levels were computed as well. Results for the warnings computed for each of the three hazards are shown in Fig. 5-20 for each room in the model. Recall that these warnings for the hazards of smoke toxicity, burn threats, and fire status were based on FED values and the thresholds summarized in Table 5-1 earlier and can only be whole integer values {0, 1, 2} at any given time in the simulation.

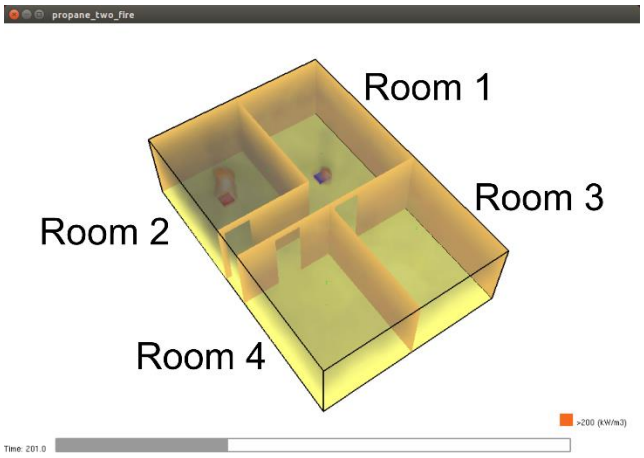


Figure 5-19: The FDS simulation is shown at 201 seconds, just after the warning for fire status in Rooms 1 and 2 had increased to Level 1

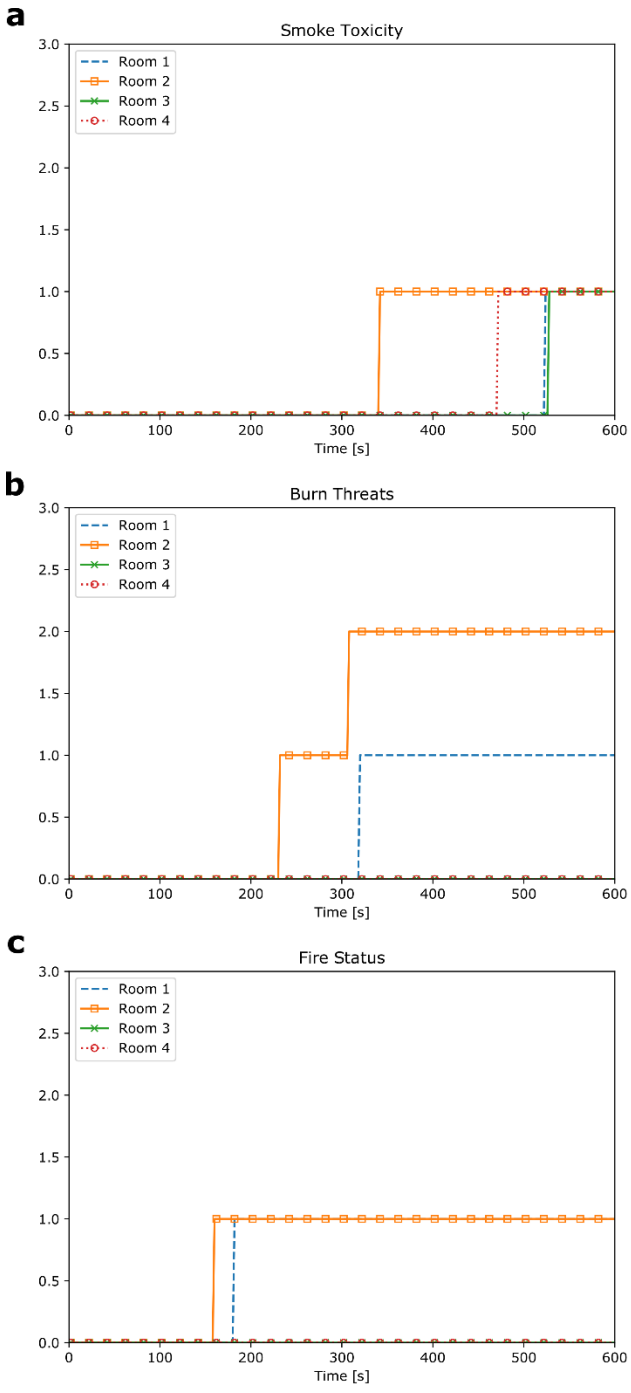


Figure 5-20: Results from the event detection model for each of the three hazards: a) smoke toxicity, b) burn threats, and c) fire status; the warning levels of {0, 1, 2} are whole integers at every increment in the monitoring simulation using a 2.0-second time step

The data used for the plots in Figs. 5-18 and 5-20 was also used in post-processing to automatically generate XML-based schedules of tasks for the Bentley ABD animation producer. One schedule was created for each of the four rooms and then imported into the BIM environment.

After performing the manual linking step to connect tasks to elements in the model, a playback of the monitoring results was available using the schedule simulator. The XML schedule generator determined the start and finish times for each threat level experienced by each room; the tabular data for each threat level is shown in Table 5-4 for reference. For visualization purposes, a universal starting time was arbitrarily chosen as 07:00:00 and the schedule generator added the threat level start times in Table 5-4 to the universal starting point to get the values seen for start and finish times briefly seen in Fig. 5-11. For example, the warning (yellow) for Room 1 has a start time of 182 seconds; thus, in the XML file generator, this was translated to 00:03:02 and added to the universal start time to get 07:03:02 as can be seen in Fig. 5-11 previously.

Table 5-4: Start times, in seconds, for each of the three threat levels based on event detection model output; the XML schedule generator determined these start times automatically and created the corresponding tasks for schedule simulation

Location	Warning – 1	Danger – 2	Severe – 3
Room 1	182	320	524
Room 2	160	232	308
Room 3	528	N/A	N/A
Room 4	472	N/A	N/A

The schedule simulator animation is presented as a video in the actual Bentley ABD environment. For the purpose of presentation in the current chapter, frames of the animation were selected to demonstrate the progression of the animation from initial, ambient conditions to the severe threat level. In the following set of images, the progression in Room 2 will be the focus, but the remaining rooms will be mentioned where appropriate. The first frame shown in Fig. 5-21 is the time of the first threat level of "warning" (yellow) reached by Room 2 at 160 seconds. By inspecting the temperature-time curve and the corresponding fluxes in Figs. 5-15 and 5-16, it can be seen that this first escalation in the threat level was due to triggering the fire status hazard (temperature and radiative heat fluxes greater than 57°C and 0.6 kW/m^2, respectively).

In the second frame, at a time of 232 seconds, the second threat level was reached as another hazard was detected and the threat level was raised to “danger” (orange). The second hazard was due to the FED for pain due to heating reaching the 1.0-threshold. By this time, Room 1 has already reached its first threat level as well at 182 seconds. Similarly, in the next frame at 308 seconds, the second threshold for the heat-related hazard (fatality from third-degree burns) exceeded 1.0, bringing the burn hazard to Level 2. Once any hazard reaches Level 2, the threat for that room immediately jumps to the “severe” (red) level. Finally, in the last frame of Fig. 5-21, the final frame of the visualization is given. Both Rooms 1 and 2 where the fires had started originally finished the simulation in the severe threat range and thus both appear red by the end. In Fig. 5-21, screenshots from the visualization correspond to stages in the FDS simulation shown with HRRPUV and soot visible in the model.

One interesting result from this example was the fact that the first threat (warning) for Room 2 occurred at 160 seconds while the fire was actually ignited immediately in that room when the FDS simulation began. Recall from the descriptions of the two fires used in this test that the 400-kW fire in Room 2 was designed with a *t*-squared fire curve having a slow growth factor. Referring back to Figs. 5-15 and 5-16, which shows the temperatures and radiative heat fluxes measured at the ceiling, the slow growth of the fire is evident in the slopes of these measurements leading up to the peak of the fire at 262 seconds. Connecting this example back to a realistic fire experience, the smoke alarm or carbon monoxide alarm would have been triggered much earlier than the 160-second warning shown here in Fig. 5-21 for Room 2.

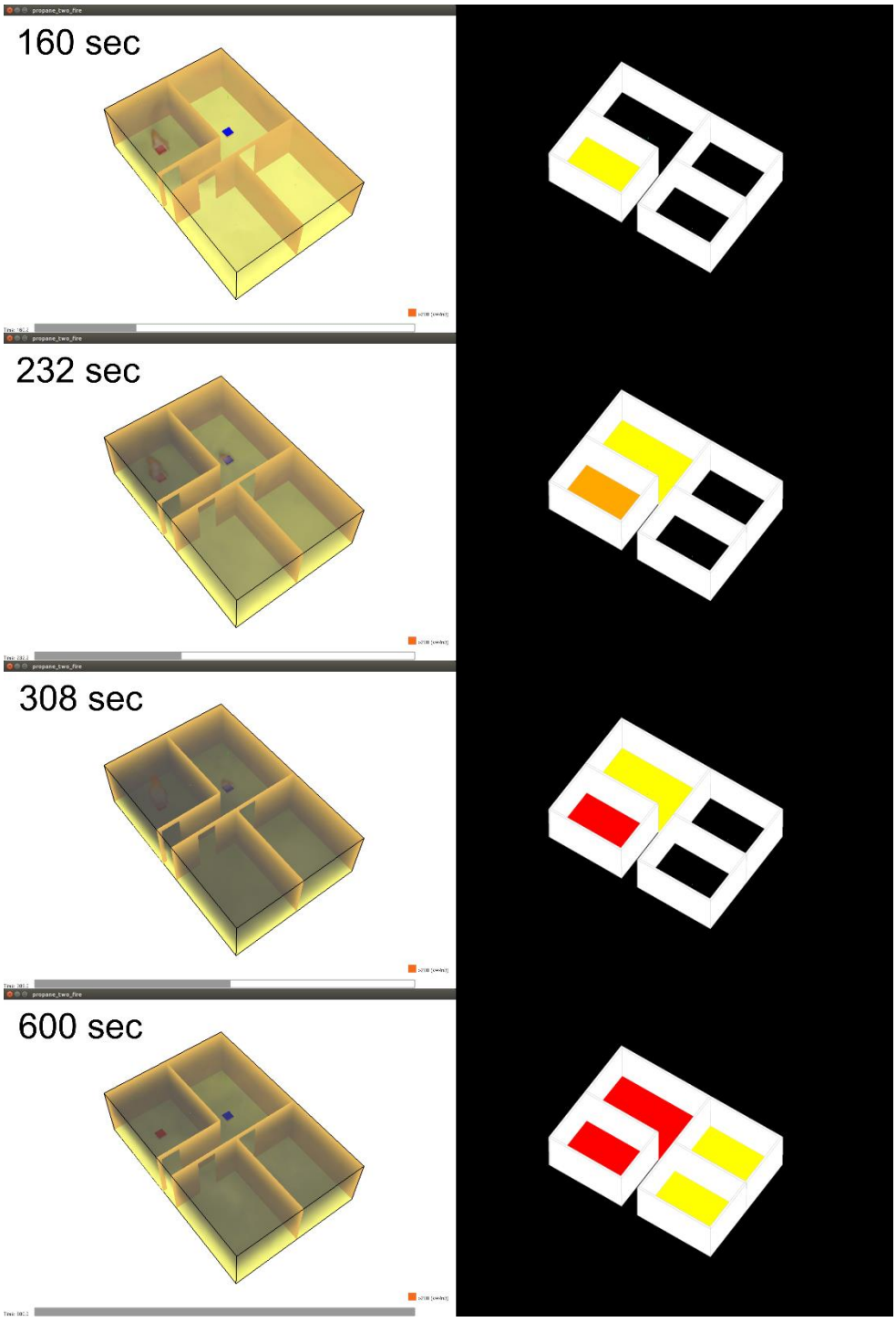


Figure 5-21: Sample results using visualization in BIM; each stage of the threat progression in Room 2 is provided, where the FDS model is shown on the left and the Bentley ABD BIM with imported schedules appears on the right

Investigating some of the data measurements at ceiling, a solely temperature-based fire detection method would have sounded the alarm around 60 seconds into the fire (using approximately 50°C as threshold). If carbon monoxide was used to detect the presence of a fire, where recommendations for this species as a detection signature range between approximately 10-50 ppm according to some studies [25, 49], then the 77 ppm of carbon monoxide received by the event detection model at 18 seconds could have sounded a fire alarm even earlier as well. This exercise serves as a reminder that the purpose of the proposed fire monitoring system is to provide information about the progression of a fire in the post-ignition state of the building, assuming that traditional detection systems would already be in place for detecting ignition.

Note that the presentation of the evolving status of the fire was done in post-processing. Preparation of the XML files for schedule simulation was performed automatically using the output of the event detection model. The demonstration of this visualization concept is meant to serve as a proof-of-concept for preparing easily identifiable information in a streamlined manner using a simple color scheme and attempting to avoid information overload [1] for the actual firefighting scenario.

5.7 Performance Testing for the Real-time Requirement

In the previous example, only four rooms were used to demonstrate the features of the real-time fire monitoring system. To this point, the near real-time computing performance for the case with a single sub-model (event-detection model) has not been discussed. The theoretical performance is expected to be sufficient when analyzing the computations involved in the event-detection model for a single sensor. The computational demand is very low because there is no solver for any matrix equations in the event detection model: all the computation is comprised of

the base floating point operations (addition, subtraction, multiplication, and division) on scalar values with various compound logic loops (while, if, else). This results in a theoretically lightweight solution because the individual calculations associated with a single sensor are limited to the FED calculations and threshold checks resulting in approximately $O(100)$ operations per sensor for each time increment. Thus, scaling from n sensors to $2n$ sensors is not a major concern from a floating-point operations standpoint.

In terms of the message-passing problem, scaling could become an issue for meeting the real-time requirement. For the four-room example shown, data collection was performed using a nominal frequency of 1.0 Hz and thus each of the four sensors would send their new data to the monitoring system for event detection approximately every 1.0 seconds. If the number of sensors grows, the number of messages sent through LCM increases for every time step. For example, in a 100-sensor model at 1.0 Hz, the system would need to handle 100 messages per second and then perform the subsequent event-detection calculations while trying to maintain the real-time feature. The computational demand is need to perform more calculations in the same small time window.

The scaling of the system was tested by employing more sensors and timing the data collection and the completion of the event-detection calculations. This test was designed to measure the computational cost of the real-time fire monitoring system and provide insight on the performance of the system. Recall from Fig. 5-3 that the data is transferred through the system using LCM. First, the simulated sensor measurements are made in the initial component: time series data is read from the file and published to main program using a waiting function in the sensor simulator, as discussed previously. This first time point is t_{sensor}, which is any time that the sensor records a new measurement.

Second, the sensor encodes the newly measured data and sends it to the main program in Fig. 5-3 using LCM: the time it is received is t_{main}. Since LCM is an efficient and reliable software library, the transfer of data from sensors to the main computing program is essentially immediate from a computational cost perspective. However, in the real application of this system for an actual fire scenario, this first data transfer represents the delay between a sensor measurement in the field and the arrival of the data for computation at a stable, remote location. A natural time delay would be expected for data transmission. In the computational case presented here, the delay is less than a fraction of a millisecond and thus t_{sensor} is very nearly equal to t_{main} in these tests. Furthermore, comparing t_{sensor} and t_{main} is essentially a measurement for how efficient LCM can encode and decode messages passed between two applications, which is something that the LCM developers have already provided.

The third and final stage of the data is in the event detection model after it has been forwarded from the main program. A data packet from the main program is encoded with the new measurements and sent to the event detection model using LCM. The message arrives in the event detection model, it is decoded, and is immediately used in the calculations discussed previously. The new event detection model results are written to an output file one line at a time for each calculation. After the last of the three hazard checks is made in the event detection model and the output is written to a file, another time sample is taken and given the label of t_{end} for the time at the end of the event detection calculation for the current data set. Since the sensor simulator and the event detection model were both developed using C++, it is possible to measure these timings using the same tool. In particular, the `gettimeofday` function in conjunction with `localtime` was used to record the times t_{sensor} and t_{end} for every data message sent by every sensor throughout the duration of the simulation. Using these two time stamps gives a measure of the computational cost

required to perform the event-detection calculations on every packet of new data. The format and precision of this time measurement was chosen to give the hour, minute, and second at the current time. However, custom code was used to represent the seconds position to three digits after the decimal. Thus, a sample timing measurement during the monitoring simulation would be in the form of t_{sensor} = 16:55:08.234, where the last position contains the seconds to three decimal places. In this format, the smallest measurable time increment was 1.0 ms. Then at some later point in time (when the event detection model is done performing its calculation on the current data), the time measurement would be taken again within that program: t_{end} = 16:55:08.237 (for example).

The computing cost associated with each message and for every sensor was calculated as the difference between these two timing measurements: $t_{cost} = t_{end} - t_{sensor}$ (performed for every message in the system). For example, in the four-room example of the previous section, fire data was measured at a frequency of 1.0 Hz for 10 minutes resulting in 601 measurements (including the initial time of t = 0 seconds) per room for the duration of the simulation. Four rooms, each with 601 measurements, results in 2404 data messages sent to the event detection model for computational work to be performed and thus t_{cost} was computed 2404 times for this case.

Since the goal of this analysis was to measure the performance of the system handling data in real time, the actual numerical values for the temperatures and species concentrations were not considered important. The data files used in the four-room example (i.e., four .csv files containing the time series data for each room) were copied several times to provide additional data files for the new cases of 8, 16, 32, 64, and 128 sensors. The same 1.0 Hz frequency was used with a random noise of 25% in the sensor delay times. These new cases provided a significantly higher message-passing volume for the system and the simulation time was cut back from 10 minutes down to five minutes for simplicity. Five minutes corresponded to 301 data messages per room

which resulted in 2408, 4816, 9632, 19264, and 38528 total messages and subsequent rounds of event detection model calculations in the simulation for the new cases of 8, 16, 32, 64, and 128 sensors, respectively.

Aggregate statistics for the various cases were computed to analyze the performance. The difference t_{cost} was computed for every message in the system for each of the cases (for example, it was computed 4816 times for the 16-room test). The smallest measurable difference for this calculation was 1.0 ms. For the real-time requirement, a target computing cost of 0.1 seconds per measurement for the 1.0 Hz case would be reasonable and was selected as the target deadline. In other words, if it takes less than 0.1 seconds (100 ms) to perform the event-detection calculations, then the system should perform sufficiently for the real-time scenario. The concern with such a high message volume in the cases of several rooms is that the event detection model must process new data serially and thus bottlenecks could theoretically occur.

Table 5-5. Quantification of the system performance with respect to increasing number of sensors; the average computational cost, maximum observed single cost, and the standard deviation are all provided as well as the percentage of measurements that required ≤ 1.0 ms

Sensors	Messages	Avg Cost [ms]	Max Cost [ms]	Std Dev [ms]	Cost ≤ 1 ms
4	2400	0.33	3.0	0.48	99.7%
8	2408	0.29	3.0	0.46	99.8%
16	4816	0.30	3.0	0.47	99.7%
32	9632	0.28	4.0	0.46	99.7%
64	19264	0.27	6.0	0.47	99.4%
128	38528	0.24	10.0	0.45	99.6%

Thus, the aggregate statistics measured in this test serve to ensure that the system is performing efficiently for the purposes of real-time use. Table 5-5 provides the average computing cost for all the data measurements in each of the cases, as well as the maximum single cost observed in the test. The standard deviation and other characterizing features were included for completeness. For example, the percentage of messages requiring less than 1.0 ms (the smallest difference that could be measured) or equal to 1.0 ms using this timing function. Table 5-5 shows

that, for 32 sensors, 99.7% messages required less than or equal to 1.0 ms of computational time in the event detection model: 9608 of the 9632 messages. This is an acceptable performance for meeting the real-time requirement of the system. Note that for the four-room case, the full 10 minutes from the previous system-testing example was used (600 seconds of data at 1.0 Hz frequency); the remaining cases used only the first five minutes (301 seconds at 1.0 Hz). The results show that for the number of sensors considered, the average computational cost of the event detection model was in the range of 0.24 ms to 0.33 ms with 99% of the calculations requiring 1.0 ms or less. Figure 5-22 also shows the average cost and the maximum observed cost for each case in addition to the real-time target of 100 ms; the target deadline of 100 ms was sufficiently met in these tests. These timing results were obtained using Ubuntu 14.04 on a standard dual-core personal computing workstation with a 2.53 GHz processor.

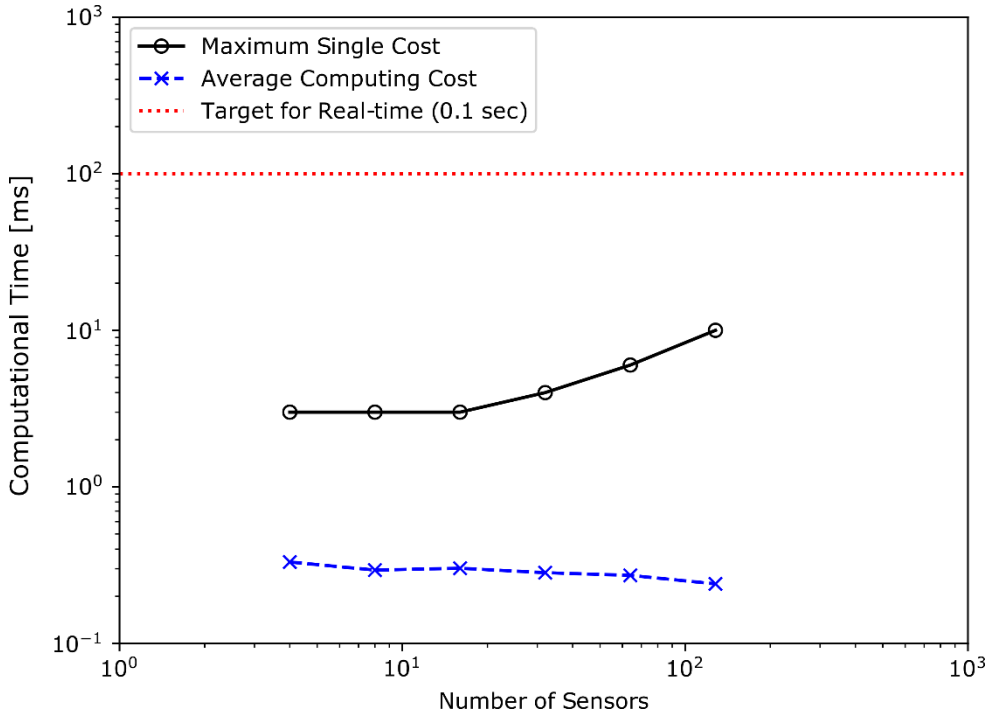


Figure 5-22: Computational cost quantified by the average time required for the event detection model to process the data; the maximum observed cost for a single data message is also provided as the upper bound

The reason for the high efficiency of the calculations, as seen in the average cost, is due in large part to the robustness of LCM in handling large volumes of message-passing tasks. Additionally, the actual calculations associated with the event detection model are very lightweight for real-time computing purposes. Specifically, there is no matrix algebra involved, there are no solvers, and file input/output (i.e., reading/writing files) is very limited.

5.8 Conclusion

The development of a real-time fire monitoring system has been presented. The components included the simulation of sensor data, formulation and implementation of an event detection model, and framework for coordinating data among these various components. Additionally, the potential for real-time visualization with BIM was demonstrated with a simple post-processing example using output from the event detection model to generate schedules. The components of this monitoring system were developed using various software packages and programming languages which were linked using LCM. The system was tested with a four-room example based on FDS-generated sensor data. In particular, monitoring was based on six fire signatures measured at the ceiling: upper layer gas temperature, radiative heat flux, and four species concentrations.

The current study lacks the ability to create fully automated visualization in real time, but the foundational work provided here will allow these features in the future. Full automation of the real-time visualization tool and the development of a user-friendly graphical user interface for receiving input from a user would be two main goals of the future work which would help bring the system closer to real-world application and improve user-friendliness. Tests were conducted using multiple sensors and an analysis of the performance for real-time applications was discussed

in this context. Coordinating data, managing computer memory, and ensuring the required calculation time is feasible for near real-time computing are all important factors that should be considered for future scalability. Most importantly, the authors do not have feedback from the intended user: firefighters and incident commanders. One way to increase the potential for adoption of new methods is to ensure that tools such as these are meeting the needs of the fire department in addition to being reliable solely from a research standpoint.

5.9 Acknowledgements

The authors would like to thank the Society of Fire Protection Engineers (SFPE) and Bentley Systems specifically for their funding provided through the Chief Donald J. Burns Memorial Research Grant. The use of certain commercial software or products identified are not intended to imply recommendation, endorsement, or implication by the sponsors or authors, that the software or products are the best available for the purpose.

5.10 References

1. Cowlard A, Jahn W, Abecassis-Empis C, et al (2010) Sensor Assisted Fire Fighting. Fire Technol 46:719–741. doi: 10.1007/s10694-008-0069-1
2. Silvani X, Morandini F, Innocenti E, Peres S (2015) Evaluation of a Wireless Sensor Network with Low Cost and Low Energy Consumption for Fire Detection and Monitoring. Fire Technol 51:971–993. doi: 10.1007/s10694-014-0439-9
3. Glasgow HB, Burkholder JM, Reed RE, et al (2004) Real-time remote monitoring of water quality: a review of current applications, and advancements in sensor, telemetry, and computing technologies. J Exp Mar Bio Ecol 300:409–448. doi: 10.1016/j.jembe.2004.02.022
4. Argent RM, Perraud J-M, Rahman JM, et al (2009) A new approach to water quality modelling and environmental decision support systems. Environ Model Softw 24:809–818. doi: 10.1016/j.envsoft.2008.12.010
5. Christodoulou S, Agathokleous A, Kounoudes A MM (2010) Wireless Sensor Networks for Water Loss Detection. Eur Water 30:41–48.
6. Wong BP, Kerkez B (2016) Real-time environmental sensor data: An application to water quality using web services. Environ Model Softw 84:505–517. doi:

10.1016/j.envsoft.2016.07.020
7. Tilak S, Hubbard P, Miller M, Fountain T (2007) The Ring Buffer Network Bus (RBNB) DataTurbine Streaming Data Middleware for Environmental Observing Systems. In: Third IEEE Int. Conf. e-Science Grid Comput. (e-Science 2007). pp 125–132
8. Brun-Laguna K, Watteyne T, Malek S, et al (2016) SOL: An end-to-end solution for real-world remote monitoring systems. In: 2016 IEEE 27th Annu. Int. Symp. Pers. Indoor, Mob. Radio Commun. IEEE, pp 1–6
9. Jang W-S, Healy WM, Skibniewski MJ (2008) Wireless sensor networks as part of a web-based building environmental monitoring system. Autom Constr 17:729–736. doi: 10.1016/j.autcon.2008.02.001
10. Alahmad M, Nader W, Neal J, et al (2010) Real Time Power Monitoring & Integration with BIM. In: IECON 2010 - 36th Annu. Conf. IEEE Ind. Electron. Soc. IEEE, pp 2454–2458
11. Kumar P, Skouloudis AN, Bell M, et al (2016) Real-time sensors for indoor air monitoring and challenges ahead in deploying them to urban buildings. Sci Total Environ 560–561:150–159. doi: 10.1016/j.scitotenv.2016.04.032
12. Doolin DM, Sitar N (2005) Wireless sensors for wildfire monitoring. In: Tomizuka M (ed) Smart Struct. Mater. 2005 Sensors Smart Struct. Technol. Civil, Mech. Aerosp. Syst. p 477
13. Zervas E, Mpimpoudis A, Anagnostopoulos C, et al (2011) Multisensor data fusion for fire detection. Inf Fusion 12:150–159. doi: 10.1016/j.inffus.2009.12.006
14. Liyang Yu, Neng Wang, Xiaoqiao Meng (2005) Real-time forest fire detection with wireless sensor networks. In: Proceedings. 2005 Int. Conf. Wirel. Commun. Netw. Mob. Comput. 2005. IEEE, pp 1214–1217
15. Jiang X, Chen NY, Hong JI, et al (2004) Siren: Context-aware Computing for Firefighting. In: Ferscha A, Mattern F (eds) Pervasive Comput. Second Int. Conf. PERVASIVE 2004, Linz/Vienna, Austria, April 21-23, 2004. Proc. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 87–105
16. Takahashi F, Greenberg PS, Carranza S, et al (2017) Real-time measurements of particulate and toxic gas concentrations. In: 15th Int. Conf. Fire Mater. 2017. pp 409–420
17. Sha K, Shi W, Watkins O (2006) Using Wireless Sensor Networks for Fire Rescue Applications: Requirements and Challenges. In: 2006 IEEE Int. Conf. Electro/Information Technol. IEEE, pp 239–244
18. Lim Y, Lim S, Choi J, et al (2007) A Fire Detection and Rescue Support Framework with Wireless Sensor Networks. In: 2007 Int. Conf. Converg. Inf. Technol. (ICCIT 2007). IEEE, pp 135–138
19. Milke JA, McAvoy TJ (1995) Analysis of signature patterns for discriminating fire detection with multiple sensors. Fire Technol 31:120–136. doi: 10.1007/BF01040709
20. Pfister G (1997) Multisensor/Multicriteria Fire Detection: A New Trend Rapidly Becomes State of the Art. Fire Technol 33:115–139. doi: 10.1023/A:1015343000494
21. Milke JA (1999) Monitoring Multiple Aspects of Fire Signatures for Discriminating Fire Detection. Fire Technol 35:195–209. doi: 10.1023/A:1015432409522
22. Gottuk DT, Peatross MJ, Roby RJ, Beyler CL (2002) Advanced fire detection using multi-signature alarm algorithms. Fire Saf J 37:381–394. doi: 10.1016/S0379-7112(01)00057-1
23. Milke JA, Hulcher ME, Worrell CL, et al (2003) Investigation of Multi-Sensor Algorithms for Fire Detection. Fire Technol 39:363–382. doi: 10.1023/A:1025378100781

24. Rose-Pehrsson SL, Hart SJ, Street TT, et al (2003) Early Warning Fire Detection System Using a Probabilistic Neural Network. Fire Technol 39:147–171. doi: 10.1023/A:1024260130050
25. Jones WW (2012) Implementing High Reliability Fire Detection in the Residential Setting. Fire Technol 48:233–254. doi: 10.1007/s10694-010-0211-8
26. Chen S-J, Hovde DC, Peterson KA, Marshall AW (2007) Fire detection using smoke and gas sensors. Fire Saf J 42:507–515. doi: 10.1016/j.firesaf.2007.01.006
27. Han D, Lee B (2009) Flame and smoke detection method for early real-time detection of a tunnel fire. Fire Saf J 44:951–961. doi: 10.1016/j.firesaf.2009.05.007
28. Aralt TT, Nilsen AR (2009) Automatic fire detection in road traffic tunnels. Tunn Undergr Sp Technol 24:75–83. doi: 10.1016/j.tust.2008.04.001
29. Wang L, Ye M, Ding J, Zhu Y (2011) Hybrid fire detection using hidden Markov model and luminance map. Comput Electr Eng 37:905–915. doi: 10.1016/j.compeleceng.2011.09.011
30. Hajian H, Becerik-Gerber B (2009) A research outlook for real-time project information management by integrating advanced field data acquisition systems and building information modeling. In: Comput. Civ. Eng. pp 83–94
31. Chen J, Bulbul T, Taylor JE, Olgun G (2014) A case study of embedding real-time infrastructure sensor data to BIM. In: Constr. Res. Congr. 2014 Constr. a Glob. Netw. pp 269–278
32. Chen K, Lu W, Peng Y, et al (2015) Bridging BIM and building: From a literature review to an integrated conceptual framework. Int J Proj Manag 33:1405–1416. doi: 10.1016/j.ijproman.2015.03.006
33. Rueppel U, Stuebbe KM (2008) BIM-based indoor-emergency-navigation-system for complex buildings. Tsinghua Sci Technol 13:362–367. doi: 10.1016/S1007-0214(08)70175-5
34. Volk R, Stengel J, Schultmann F (2014) Building Information Modeling (BIM) for existing buildings — Literature review and future needs. Autom Constr 38:109–127. doi: 10.1016/j.autcon.2013.10.023
35. Guo Q, Salinas A, Jeffers AE (2015) Inverse model for determining heat release rates. In: Int. Fire Saf. Symp. pp 431–439
36. Huang AS, Olson E, Moore DC (2010) LCM: Lightweight communications and marshalling. In: Intell. Robot. Syst. (IROS), 2010 IEEE/RSJ Int. Conf. pp 4057–4062
37. (2015) The OpenMP API specification for parallel programming.
38. Hartzell GE, Switzer WG, Priest DN (1985) Modeling of Toxicological Effects of Fire Gases: V. Mathematical Modeling of Intoxication of Rats By Combined Carbon Monoxide and Hydrogen Cyanide Atmospheres. J Fire Sci 3:330–342. doi: 10.1177/073490418500300504
39. McGrattan K, Hostikka S, McDermott R, et al (2015) Fire dynamics simulator user's guide. NIST special publication, 1019, (6)
40. Purser DA, McAllister JL (2016) Assessment of Hazards to Occupants from Smoke, Toxic Gases, and Heat. In: Hurley MJ, Gottuk DT, Hall Jr. JR, et al (eds) SFPE Handb. Fire Prot. Eng. Springer New York, New York, NY, pp 2308–2428
41. Alarie Y (2002) Toxicity of Fire Smoke. Crit Rev Toxicol 32:259–289. doi: 10.1080/20024091064246
42. Xu Z, Lu XZ, Guan H, et al (2014) A virtual reality based fire training simulator with

smoke hazard assessment capacity. Adv Eng Softw 68:1–8. doi: 10.1016/j.advengsoft.2013.10.004
43. Engineering T (2017) Pathfinder 2017: Technical Reference.
44. Tang F, Ren A (2012) GIS-based 3D evacuation simulation for indoor fire. Build Environ 49:193–202. doi: 10.1016/j.buildenv.2011.09.021
45. Chow CL, Chow WK, Lu ZA (2004) Assessment of smoke toxicity of building materials. In: 6th Asia-Oceania Symp. Fire Sci. Technol. pp 132–142
46. Lawson JR (1997) Fire fighters' protective clothing and thermal environments of structural fire fighting. In: Perform. Prot. Cloth. Sixth Vol. ASTM International, pp 334–352
47. Peacock RD, Reneke PA, Bukowski RW, Babrauskas V (1999) Defining flashover for fire hazard calculations. Fire Saf J 32:331–345. doi: 10.1016/S0379-7112(98)00048-4
48. Su JZ, Crampton GP, Carpenter DW, et al (2002) Kemano Fire Studies--Part 2: Response of A Residential Sprinkler System.
49. Su JZ, Crampton GP, Carpenter DW, et al (2003) Kemano Fire Studies--Part 1: Response of Residential Smoke Alarms. Natl. Res. Counc. Canada
50. Stroup DW, Bryner NP, Lee J, et al (2004) Structural Collapse Fire Tests: Single Story, Wood Frame Structures. Natl Inst Stand Technol Gaithersburg, MD, NISTIR 7094:84.
51. Babrauskas V, Peacock RD, Reneke PA (2003) Defining flashover for fire hazard calculations: Part II. Fire Saf J 38:613–622. doi: 10.1016/S0379-7112(03)00027-4
52. Guillaume E, Didieux F, Thiry A, Bellivier A (2014) Real-scale fire tests of one bedroom apartments with regard to tenability assessment. Fire Saf J 70:81–97. doi: 10.1016/j.firesaf.2014.08.014
53. Liang FM, Chow WK, Liu SD (2002) Preliminary Studies on Flashover Mechanism in Compartment Fires. J Fire Sci 20:87–112. doi: 10.1177/0734904102020002746
54. Lee EWM, Lee YY, Lim CP, Tang CY (2006) Application of a noisy data classification technique to determine the occurrence of flashover in compartment fires. Adv Eng Informatics 20:213–222. doi: 10.1016/j.aei.2005.09.002
55. Hartin E (2015) Fire development and fire behavior indicators.
56. Alarifi AA, Dave J, Phylaktou HN, et al (2014) Effects of fire-fighting on a fully developed compartment fire: Temperatures and emissions. Fire Saf J 68:71–80. doi: 10.1016/j.firesaf.2014.05.014

Chapter 6
Conclusion

The computational framework for modeling the fire-structure interaction problem was presented here by considering a three-stage, sequentially partitioned approach. First, the fire simulation was conducted in CFD using FDS to compute the thermal boundary condition for the structure. Second, the fire-structure coupling methods were used to transfer the thermal boundary conditions to the structure and compute equivalent nodal fluxes in the FEA model. Third, the thermo-mechanical shell element was used to compute the resulting temperatures and displacements in the FEA model based on the input from the CFD-based fire simulation. Altogether, this workflow represents the complete fire-structure coupling framework.

Next, a system for real-time fire monitoring was developed to provide the computational means for using measured fire signatures in sensor-assisted firefighting. The components of data collection, real-time computing via sub-models, and finally visualization in the field were discussed. For the purposes of developing this system, sensor data and measurements were restricted to the simulation of sensor nodes as a component of the full system, rather than using physical hardware for this need. The real-time computing aspect using sub-models was demonstrated through a novel event detection model for analyzing live data and assessing the fire hazards. A proof-of-concept for real-time visualization in the BIM environment was presented as a demonstration of future use in such a setting.

6.1 The Fire-structure Interaction Problem

The research presented in this dissertation has contributed to two different sides of the problem of structural fires. From the analysis and computational mechanics standpoint, the coupled thermo-mechanical shell element provides the researcher or designer a tool for analyzing structures exposed to fire using a partitioned solution approach. The fire simulation software FDS is not typically employed for its heat transfer calculations and it does not include the ability to compute deformations in the structure. Thus, Chapter 2 showed that using the trapezoidal rule for numerical integration and linking the thermal boundary condition data from FDS to the conduction heat transfer model in Abaqus, solutions for a non-uniformly heated plate were provided with good accuracy and very efficiently. Relative errors of less than 1.5% were reached using 128 thermal shells in a 16×8-element configuration, which required less than one minute (47.4 seconds) of computing time as opposed to the 48.5 minutes that were needed for the much higher-solution solid-element model.

Coupling the FEA representations of the thermal and mechanical shell elements is an essential component of the fire-structure simulation framework for any analyst interested in both the thermal and mechanical response in the structure due to the presence of a local fire. Thus, in Chapter 3, using the virtual work approach, these two formulations were successfully coupled and verified using benchmark tests. In particular, for the thermal stress cylinder problem to compute the stress, the thermo-mechanical shell element demonstrated more than sufficient accuracy with an absolute error of $O(10^{-13})$ when comparing computed stresses with this analytical solution. Using the simply supported plate verification, the solution with the coupled shell performed with a difference of 0.5% when compared with the reference solution.

The methods for coupling surface fluxes in the CFD fire simulation to the structure in the full thermo-mechanical FEA model provide an intuitive and general approach to handling temporally and spatially non-uniform fluxes in problems involving thin-walled structures. With the applications in Chapter 4, the fire-structure coupling methods were demonstrated with the thermo-mechanical shell to compute temperatures and displacements simultaneously in the FEA model. The convergence behavior was shown in plate exposed to the local fire while the I-beam model represented a practical use for this type of element with multiple interface surfaces. Finally, the relative time step and mesh size were discussed in the context of these coupled simulations through the use of 225 FEA simulations based on various thermal boundary conditions. From the perspective of future modeling using this approach, this study mainly emphasizes that properly modeling the fire simulation in FDS is a critical starting point to obtaining meaningful results (i.e., using the proper CFD mesh size in FDS). The use of a 5-second subcycling time step provided nearly identical results to the 30-second step, which was an important affirmation of prior research employing the time-averaging subcycling approach.

6.2 Computational Framework for Real-time Fire Monitoring

Contributions to the real-time fire monitoring problem were intended for use by the fire department, incident commander, and firefighters; this work was presented in Chapter 5. In this novel system, the computational framework for using real-time fire signatures in conjunction with sub-models and visualization tools in a distributed setting will provide an improvement to the state of sensor-assisted firefighting. The system efficiently coordinates the data from sensors to sub-models using LCM, the output designed for visualization purposes will enable this technology to provide the fire department with dynamic BIM models in the future.

Lacking a robust WSN in this stage of the research, the fire-monitoring system used simulated sensors to asynchronously send messages to the main computational platform in a manner similar to realistic sensors. Data messages originating from the WSN were received at the main program for fire monitoring and subsequently coordinated to the proper sub-models in real time. Currently, one sub-model was developed and added to the system: an event detection model for assessing the hazards of the fire in real time. These hazards included measures of smoke toxicity, burn threats, and fire spread in the structure; the first two relying on using empirical formulas for the FED calculations from fire safety engineering in real-time to assess the hazards at the location of each unique sensor. The real-time goals were tested using up to 128 simulated sensors; this scenario would be comparable to installing a WSN in a small business. For the event-detection model calculations, the system met the real-time requirement by processing new sensor measurements and assessing the fire hazards efficiently: less than 0.5% of the data required at least 1.0 ms of computation time in the 128-sensor case. This efficiency is attributed to the lightweight implementation of the event detection model as well as the robustness of the LCM message-passing library.

In addition to these real-time components, a post-processing example showing the use of BIM as a visualization platform for issuing real-time warnings based on the changing threat levels computed by the event detection model. Although not immediately field-ready, this technology has provided the foundation for a real-time computing system in the arena of fire monitoring beyond ignition of the fire in the structure. A discussion of the software development process for this computational framework is presented in Appendix A.

6.3 Limitations and Future Work

The shell element presented for capturing the coupled thermo-mechanical response of structures exposed to fire was developed in the first task of the dissertation research. One limitation for the coupled fire-structure simulation framework provided here is that the thermo-mechanical shell element was restricted to the thermoelastic range for the current study. For use in more applicable real-world scenarios, the shell element would need to be upgraded to consider temperature-dependent material properties and potentially non-linear geometric deformations. A second limitation is the fact that there is not sufficient experimental data to validate the implementation of the shell element. This validation stage would more be more beneficial once the shell formulation was extended for use in non-linear analyses. Additionally, one of the practical limitations for scalability of this approach is the fact that a larger system of equations is solved for each increment of the coupled analysis in comparison to solving the two systems separately in a weakly coupled approach.

For real-time fire monitoring, the research was limited to the lab environment: specifically, the proposed system was not tested for a real-world fire-monitoring scenario. While the particular multi-criteria sensors discussed here may not yet exist, the methods for receiving real-time data from sensors have been in development in fields such as structural-health monitoring and water-quality monitoring for the last decade and could be adopted for the fire monitoring scenario as well. Regardless, hardware from the sensing perspective is one broad area of future work both for the research presented here and the future of real-time fire monitoring.

To this point, the event detection model was the only sub-model included while the future vision for the fire-monitoring system is to have multiple sub-models included and simultaneously providing real-time calculations leading to useful information for the firefighters. For example, the

inclusion of a multi-room inverse fire model would provide the system with a means for computing the heat release rate for each room from the measured fire signatures using a two-zone simulation model (CFAST). Including such an inverse fire model and other potentially useful real-time computing packages would provide the incident commander and firefighters with more information about the fire status through the duration of the fire. Ideally, another sub-model will be a fully automated feature for presenting real-time data in the BIM immediately for use in the field. Since BIM is a tool primarily used for the design and construction phases of building development, the ability to sync real-time data with the graphical display is a challenging feature to include. The future work for this component of the system would be to develop the proper software solution to allow for real-time data streams in the BIM platform chosen for this research: Bentley ABD.

The two tasks presented in this dissertation have contributed to the structural fire problem: (1) from the perspective of computational modeling in the pre-construction phases and (2) with the intention of aiding firefighter during the post-ignition fire state in the building. The limitations and future work presented here are the areas in which these contributions can reach beyond the academic level and into the practical application space. While both topics are different in their approaches and uses, they each serve to improve aspects of fire safety that make the built environment safer for occupants and firefighters in our modern infrastructure.

Appendix A
Software Development for Real-time Fire Monitoring

The strategy for firefighting has remained relatively unchanged for years and the approach is almost completely based on improvisation and experience as very limited information is available to the fire department on approach to the event. In order to help address this issue, a recently funded research project in the Department of Civil and Environmental Engineering at the University of Michigan aims to take a step towards equipping firefighters with more data and a better understanding of the fire scene before arriving to the site of the event. Specifically, this new research project aims to provide computational tools in the form of event-detection and fire-monitoring software that can be used in an advanced sensor network of the future.

As a subset of the larger research project, the following report focuses on providing a reliable foundation for the new research effort by using strategies from software engineering and applying trusted methods from scientific computing. The result of this effort was the creation of version-controlled repositories for hosting the research project, migration to a TriBITS build system, the inclusion of a simple unit test to ensure required components are available on the user's Linux system, and the creation of documentation which outlines the installation process for developers on the research project. The addition of these features will provide current and future developers on the project with a stable development environment and structured approach to the actual fire safety contributions that follow. By establishing the use of high-quality software engineering tools, developers on the main research project will be able to provide the eventual future users with robust research-quality software to perform the overarching goal of monitoring

fire events in real-time using a sound computational foundation. For context to the reader, this technical report was completed prior to the work presented in Chapter 5.

A.1 Introduction

The hazard of structural fires is still a real and modern danger. Structural fires cause numerous deaths per year in the United States and can result in extensive property damage as well [1]. The human and property impact of fires also affects those called to respond to the fire event: the firefighters. Environmental threats to building occupants and first responders include extreme temperatures, toxic gases (e.g., carbon monoxide, hydrogen cyanide), disorientation due to poor visibility coupled with unfamiliar surroundings, and a changing environment that may result in falling objects, structural collapse, or entrapment [2]. In addition to these severe environmental hazards, firefighters often lack critical information that might be useful in making decisions at the scene of the fire event. Additionally, toxic gases such as carbon monoxide and hydrogen cyanide are not detectable without the aid of technology. Limited knowledge about the contents of a building and the availability of oxygen may lead to a misperception regarding the nature of the fire (e.g., fire intensity, spread rate, and the potential for flashover). In addition to posing a direct threat to firefighters, these challenges can potentially slow the rescue of building occupants and prolong the progression of fire.

The realization of a multi-criteria wireless sensor network (WSN) and real-time visualization network is an ambitious undertaking. The currently funded research project seeks to establish the computing infrastructure for such a multi-criteria WSN for real-time fire monitoring. A challenge lies in the need for technologies that enable rapid event detection, simulation, and visualization of the data so that actionable information may be extracted in real time. This work is motivated by complementary studies which aim to integrate sensor data within fire forecasting

simulations (e.g., Imperial College's *Fire Navigator* [3]). The novelty of the research lies in the use of multiple fire signatures to identify fire events in real-time and to visualize the information in a Building Information Modeling (BIM) software to facilitate rapid decision-making during firefighting operations. If successful, the proposed research will open the door to future studies regarding the development of sensor technologies and full-scale validation, and its commercialization and widespread deployment via industry partners in the fire protection market. With a brief background of the research topic provided, this technical report will now address the computing needs for such a project.

The objective of the work presented in this technical report was to advance the efforts of the existing research, which aims to provide this real-time fire monitoring tool for future fire department command centers and firefighters, through the use of software-engineering tools. Due to the ambitions of the overarching objectives in the main research project, the completed work presented herein was focused on making a step towards the realization of this goal from a computational perspective.

In a typical fire scenario, the fire department is forced to send groups of firefighters into a structural fire event with minimal information about the current status of the fire. Fire detection is a simple binary response of a fire alarm: the fire is either present or it is not. Thus, the trigger for the fire department to respond is just the activation of the alarm at the site of the event. This method of fire response has been the approach for decades but with one major issue that is difficult to address: the built-in delay between initial alarm activation and arrival of the firefighters to the scene of the fire event. This is the case of the improvised response of firefighters (as seen in the top two images of Fig. 5-1 previously). While it is impossible to completely eliminate the delay between alarm activation and the arrival of first responders to the scene, it is practical to consider

equipping the fire department with more information regarding the status of the fire on their approach to the scene. Thus, the system to be designed for fire-real fire monitoring targets the post-ignition state in the building, rather than the fire-detection problem.

The current phase of the main research project, from which a subset of tasks were selected for the content of this technical report, aims to develop a *computational framework* for a robust fire monitoring tool without a particular physical sensor available at this time. The main computational features of such a tool include the following (also shown in Fig. 5-2):

1. An internal network for connecting various components of the monitoring system; for example, software which may pass incoming data from a live sensor to one computational application and then communicate a computed result to another application for further use (i.e., an internal message-passing interface for handling data transfers between independent applications).
2. A novel application for computing environmental variables within a burning building during a live fire event; for example, this application will produce real-time calculations for the observer (i.e., fire department) regarding key fire parameters such as fire spread and smoke toxicity levels. This will be the event detection model for assessing hazards in the building based on real-time data.
3. The ability to add new applications in future development, such as an inverse fire model (IFM) to estimate the heat release rate in real-time based on the measured fire signatures; this will provide extensibility for future use as new applications like the IFM are developed for the system.

There are many other important features that must be considered to practically develop such a monitoring tool, such as a visualization component for reporting computed results and other user-

friendly interfacing features. However, the main focus of the current work is the computational component strictly used for producing numerical calculations. As a result, this report discusses the contributions to the development workflow for tasks that will be performed by the main computing workstation (as seen in Fig. 5-2 previously).

Each of the computational components described above have had some previous work completed related to their implementation in the main research project. Part 1 above relies on a message-passing library and toolkit called Lightweight Communications and Marshalling (LCM) [4] which provides compatible C-style data structures to be passed between various applications written in C++, Fortran, Python, and other languages. LCM is an open-source package, from which the basic internal communication network has been established for use in the current project. The aim of Part 2 is to contribute a novel application for computing particular fire parameters based on incoming data from the fire event for characterizing phases of the fire spread. This component will be a module running simultaneously with other application (such as the IFM) to provide hazard assessment about the current fire scenario in real-time. The IFM mentioned in Part 3 works with a fire modeling software produced by NIST called the Consolidated Model of Fire and Smoke Transport (CFAST) [5] which is capable of performing fire simulations based on two-zone models; work on this component was completed in a previous project to predict heat-release rates during the fire event using a dedicated source code.

Due to the scale of the overarching research project, this technical report focused on using pieces of these previous project developments and aimed to highlight the regions of the ongoing research where scientific computing practices can augment the overall goals of the main project. As a result, the effort turned towards establishing the fundamental components of the proposed new software based on trusted methods of software engineering. This early investment intended to

provide a stable environment for current developers on the project as well as future contributors. Specifically, the work for this project consisted of (1) the provision of a third-party library (TPL) repository to host all necessary TPLs for the system, (2) the creation of a structured approach to automatically installing TPLs necessary for the research software, (3) the establishment of a version control system for handling the development stages of the research project, and (4) the creation of initial documentation for developers and users to understand the installation of the project software.

A.2 Development of Research Software

One motivation for investing in the development of software infrastructure for this research project was to avoid the validation-centric development pattern which can lead to an unintentional waterfall-style workflow. A challenge in creating useful scientific software is that the researcher from time-to-time must be the project manager, planner, user, developer, and owner (to some degree) at various stages of the workflow. In the research setting, it is easy for an individual researcher to take on all these roles throughout the project without considering the bigger picture such as who the intended future users may be, who the next developers on the project may be, or what the limits or bounds of the project may be. Understanding a software project in a holistic and symbiotic way can have numerous benefits for the current and future developers, external collaborators, and the end users.

Project planning and laying the foundation for high-quality deliverables is a valuable component of working on long-term projects, software or otherwise. There are parallels in construction projects of all sizes where the amount of planning is generally proportional to the complexity and duration of a project. Practicing good software engineering technique and considering the life-cycle of a research-level software project is an important aspect for providing

a solid framework for future development. Often overlooked and under-credited, developing project infrastructure and providing documentation with policies for future use can be critical for producing mature and portable applications.

Sometimes the instinctual approach to developing a software solution is to immediately begin writing code and testing it simultaneously with the hope of fulfilling some sample solution or target value (for example, implementing a solver for a linear system $\mathbf{Ax} = \mathbf{b}$). This "validation-centric" style is not suitable for larger, complex systems and is not practical when different teams may be working on various aspects of a long-term software project. Generally speaking, the casual observer may place an emphasis strictly on the net output of a developer's effort: for example, in terms of the number of lines of code produced in a given time. Once again, a typical instinctual response to completing an assignment may be to get started with the code-writing process immediately without considering the overall structure. This may be a good approach for some situations but this method is not scalable.

Orso provides a relatable exercise for thinking about the scalability of work on a large project in terms of the software engineering development process [6]. Imagine situations that may arise in a university setting: a homework exercise (roughly 10^2 lines of code), a small individual project (10^3 lines of code), and finally a group project or term project (10^4 lines of code). Success in these situations and the ability to complete the work is directly linked to the "programmer's effort", as Orso describes it [6]. However, for more complex systems, such as the creation of a word processor (10^5 lines of code), development of an operating system (10^6 lines of code), or even a new distributed system (10^7 lines code), the ability to complete the project is not simply a programming effort but rather a "software engineering effort" in this case [6]. The main idea is that planning to develop a complex system is just as important as the actual programming work

required on such a project, including in the scientific computing setting. Thus, using the methods of software engineering to coordinate long-term efforts on a complex project is an important step towards giving each developer on the project a better chance of producing more meaningful code.

The goals of this project were aimed at laying the foundation for a new software project from a recently funded proposal with these qualities in mind. To quantify the level of complexity of the proposed research project in terms of the examples given here, the total volume of the project will fall somewhere between the term project and word processor mentioned earlier, with several smaller source code packages interacting with more complex third-party libraries and other operating systems. Thus, the focus on software engineering principles and providing a robust system for producing high-quality and portable software during the duration of the overall research project was a long-term goal of this work.

A.3 Software Engineering Tools

By considering the main research project in the context of the software construction process, it is possible to categorize various aspects of the overarching research goals in terms of discrete software engineering components. In Fig. A-1, a pyramid representing the development process highlights some of the main features of software projects [7]. Speaking about the software construction in this regard, the problem statement at the foundation of the project can be defined for the current study using language from the original proposal: "*To establish a computational infrastructure for handling a future multi-criteria wireless sensor network to be used in real-time fire monitoring.*" With this perspective in mind, the current research study aims to develop the computational tools necessary for providing such a monitoring framework of the future, specifically in terms of data handling, computing various environmental statistics, and visualizing the results for the end user.

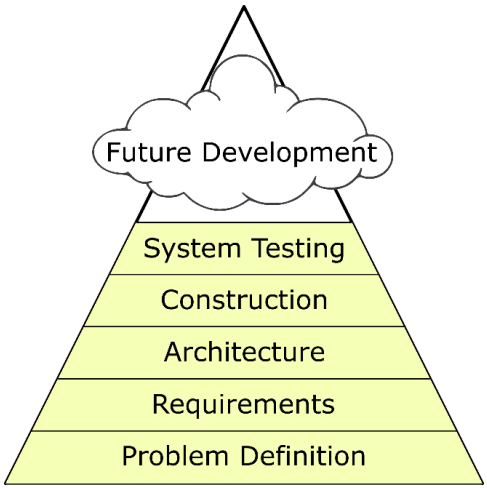


Figure A-1: The pyramid of software development

Software requirements of the research work, the second step of the pyramid in Fig. A-1, include the development of novel source code for handling real-time fire data, the integration of existing software for computing key fire parameters, the use communication software for managing the transfer of data between applications, and finally the presentation of computed fire parameters and data in a visualization module using BIM technology. The proposed architecture for accomplishing the project requirements involves splitting the implementation into two distinct domains. First, a computational domain will host all the applications related to handling incoming raw sensor data, calculating various fire parameters, and formatting output for the visualization module. Second, the visualization module will use that output to produce an overview of all environmental statistics related to the fire including visualizations of the relevant fire-event data. Specifically, these two domains (the "computation" domain and the "visualization" domain) will be distributed on two systems (see Fig. 5-2). Due to funding-specific restrictions, which could be considered another project requirement, the visualization module must be completed using commercial BIM software produced by Bentley Systems called AECOsim Building Designer; this software is restricted to the Windows operating system. However, the compute machine must be

developed on a Linux machine in order to make use of some of the TPLs such as LCM (which will be discussed in later sections).

Using existing software such as LCM and CFAST with the soon to be developed source code for computing environmental parameters related to the fire status, the work for this course project employed the tools of scientific computing specifically to migrate various components of the research project to a TriBITS build system. The goal of this effort was to provide for general portability for this future monitoring tool. Performing this migration enlisted the use of version control software for hosting the main research code. This project also focused on addressing the implementation of source code in terms of the overall software architecture by linking packages and TPLs within the TriBITS build system. The main deliverables were a version control system for handling the research source code of a new fire monitoring project and its required libraries, the development of installation scripts for making the research package more portable, and finally the creation of documentation to support current and future developers on the project as well as the end user.

A.3.1 Version Control and Build System

The first contribution to the new research project was to establish a version control system for both the project source code as well as the necessary third-party libraries (TPLs). One repository was dedicated to the project code and related files to create a TriBITS build system (discussed next). A second repository was created for maintaining the TPLs of the project. First, a private repository called `fire_main` was created on GitLab to host the packages of source code for the main research project. It is here that a TriBITS-based project framework was established from the various components of the research code. The top-level repository contains the necessary CMake files for building the TriBITS project, as would be expected for any traditional build

system. An overview of the top-level repository is shown in Fig. A-2 below. A subdirectory named `packages` was introduced to hold the source code of all the in-house developed research software such as the applications for processing incoming fire data to compute environmental statistics related to the fire scenario and eventually formatting the output for the visualization module.

Specifically, one initial package was included in the subdirectory called `packages/RTFM` which was intended to serve as the model format for including future packages into this project. `RTFM` is the package which will contain the initial version of the source code for the *real-time fire monitoring* application developed in future research. Presently, only a simple message-passing test was added to this repository as an exploratory code for checking the build system and use of TPLs, as discussed next. It serves as a basic unit test to ensure the TPLs were installed properly and the TriBITS build system can locate and use their related header files, libraries, and executables.

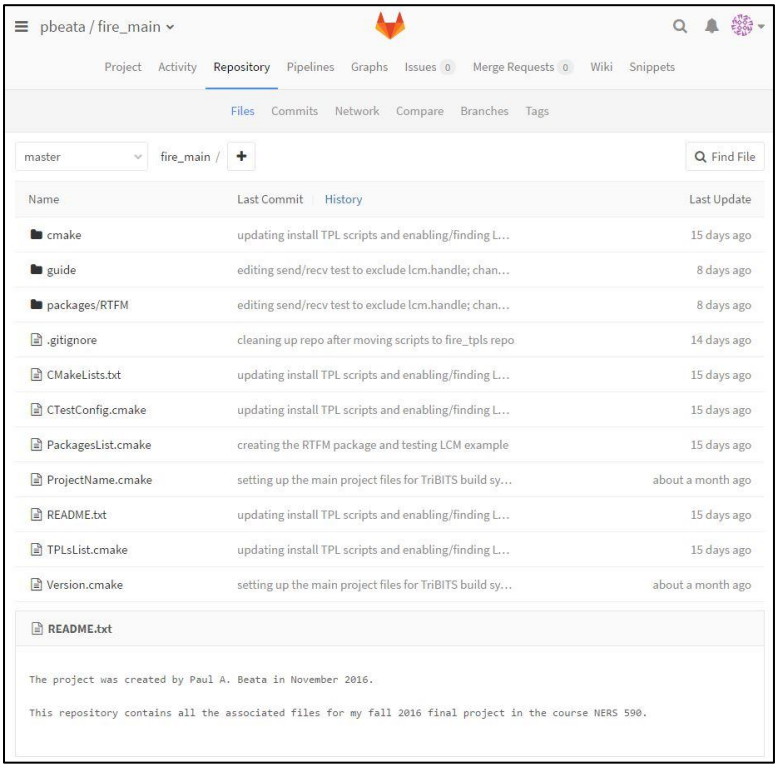


Figure A-2: A view of the top-level repository for `fire_main`, which is the main repository hosting the research project

Additionally, the subdirectory in `fire_main/guide` was provided in order to organize project documentation. At this stage of the project, only an installation guide has been provided for future developers and users as well as a simple `cmake` script which includes the necessary flags for building the project with the `cmake` command. Both of these documents in the `guide` subdirectory will be described in later sections.

A.3.2 Third-Party Library Installation

The second contribution using version control was the creation of a public repository on GitHub to hold the necessary information for installing the TPLs of the research software; it was named `fire_tpls`. As with many complex projects, the research software that must be developed for the funded work will require the use of several TPLs. Currently, the project has a strict requirement of two specific TPLs: Lightweight Communications and Marshalling (LCM) and the Consolidated Model of Fire and Smoke Transport (CFAST).

The LCM software is hosted on GitHub and will serve as a message-passing library for communicating raw and processed data between the various applications (i.e., between various `packages`, as discussed in the segment on `fire_main`). CFAST was developed by NIST for performing zone-model simulations of compartment fire scenarios; this software is also hosted on GitHub. As a result of this convenient hosting of both pieces of software on GitHub, users of those applications can acquire them easily using the `git clone` command from a terminal or by downloading a zip file with all their contents. While this process may be straightforward for an experienced user or developer, the steps for building the individual software packages may be difficult for a new researcher in the team.

Regardless of the experience of the user, it is beneficial for future developers and users of the new fire-monitoring software to have a systematic, scripted method for installing TPLs and

linking them to the packages of the larger project-wide build system. Thus, to provide an automated approach for installing the TPLs of this research project, the `fire_tpls` repository was structured to include new scripts for TPL acquisition and installation on a new machine used in the future of this project. For example, if a new developer (or future user) is building the research software on a new machine, the script `install_tpls.sh` was developed for this project and can be called from the command line to install all specified TPLs. The implementation of the automatic TPL installation process for the LCM software (and in general for additional TPLs) is given more attention in the Results section to follow. This was one of the primary efforts presented in this report and its details are provided by outlining its main features during the detailed explanation provided later. The top-level view of the newly created `fire_tpls` repository can be seen in Fig. A-3 below.

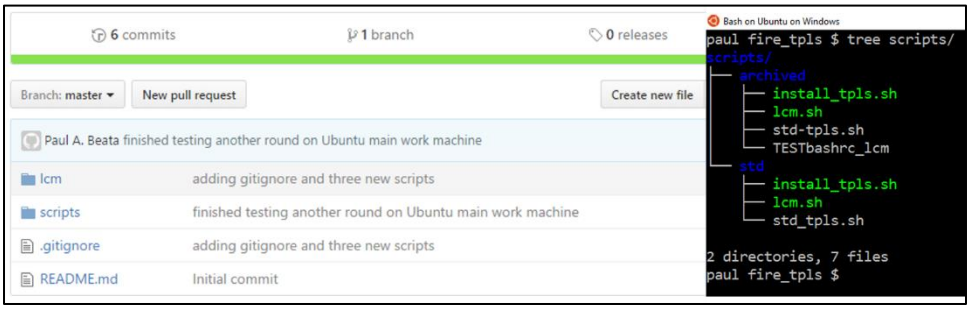


Figure A-3: A view of the top-level repository for `fire_tpls` which hosts the required TPLs for the research project

A.3.3 Documentation and Policy

As mentioned previously, the `fire_main/guide` subdirectory contains information for developers and users of the new research software. Included in the guide subdirectory is the system documentation file. This document contains a guided step-by-step procedure for installing the required software of the research project. Instead of providing a list of terminal commands for the

user to apply blindly, the instruction set in the system documentation was designed to guide the user through the installation process using a narrative style. This helps to explain the installation process for a new user and also demystify the build process by describing the required environment variables and general infrastructure of the project along the way.

Creation of the System Documentation guide was enhanced through the process of testing the install process on multiple systems. For example, after one version of the documentation was completed, the new set of instructions was used to attempt installing the project software (TPLs and main source code) on a different machine. During the installation process, new notes were added to the documentation based on the experience. This lead to a better understanding of the software from a developer perspective and provided a set of instructions which better captured the user's installation experience on different machines. The system documentation for installing the research code is provided in the repository for this project.

The term "policy" was used here as a generic descriptor for the process of creating the repositories and installation scripts for the research. This system was developed with expansion and portability in mind. Thus, when future TPLs are needed for the research software, the current structure of the build system will be able to support them through the use of new installation scripts based on the ones developed for this project. Similarly, for newly developed source code, additional packages may be added to the main repository and linked with the same methods used for the initial test package used here. While these concepts appear merely as concepts in this section, the details are covered in the deliverables section that follows where proof of these features is provided.

A.4 Research Deliverables

The goal of this section is to demonstrate these foundational additions to the main research project for real-time fire monitoring. Since several components were integrated into one build system, it follows that the results should highlight the infrastructure investments made during the timeline of this technical report. As discussed throughout this report, the type of the work completed for this project was very much in the space of software engineering and so installation procedures, script-writing, and general organizational concepts were the main focus of the results presented herein. Thus, the nature of the results are tied to these topics as opposed to producing some desired numerical values. What follows is a straightforward presentation of the deliverables provided by this project with a discussion about the meaning and implications to follow.

A.4.1 Automatic TPL Installation

As discussed earlier, one of the main efforts in this project was the development of an automatic TPL installation script for acquiring and building necessary external packages for the research software. Two known required TPLs exist for the main research project currently: LCM and CFAST. Thus, two scripts were created for acquiring and installing these external software packages which are hosted on GitHub. The LCM script `lcm.sh` was fully integrated into the `fire_tpls` repository, was part of the TriBITS build system, and was tested during this course project. The CFAST script `cfast.sh` was developed but not fully integrated into the project repository nor included in the TriBITS build system as of the current report. However, the CFAST script was tested and debugged on several machines. The details of the CFAST script will not be presented here as that work has not been implemented into the main project repository.

One important note about the `fire_tpls` repository is that the actual source code for a particular TPL is not tracked in this repository: instead the installation scripts provided here

automatically use the proper `git clone` commands to retrieve the required versions of the TPLs from their own respective repositories at the time of building and installing. This is a typical approach for larger projects that rely on external software; there is no need to keep the external source code in the main repository because it is already being hosted elsewhere by the original developers of that particular package. Thus, an automatic TPL installation process was made possible by a simple, extensible logic. Regarding this extensibility logic, the main installation script called `install_tpls.sh` was structured in such a way that future additional TPLs may be added into this installation process in a systematic and reliable manner.

A new user must specify only one additional environment variable in the terminal to begin the installation process: the installation directory for the TPLs. This directory must be chosen as a stable location in the system which the user does not expect to change. The suggested default option for this choice is simply to create a new directory in the default `$HOME` location. This new directory must be exported as the environment variable `$TPL_INSTALL_DIR` during the installation process. For example, the documentation developed for this process recommends creating the `$HOME/installed_tpls` directory to serve as this install location. Error checking was provided in the `install_tpls.sh` script to ensure that the user does in fact specify the `$TPL_INSTALL_DIR` variable before the installation process begins.

A few specific features were included in the first script for installing individual TPLs; specifically, the script for installing LCM is described here. Version information for the LCM package is stored in a separate file called `std_tpls.sh` which keeps the cloning and installing process updated for future users. The current version of LCM is 1.3.1 but when 1.3.2 is released in the future, the developers on this project simply need to update the `std_tpls.sh` script to reflect this change in the version in one location only as opposed to rewriting the `lcm.sh` script.

It is a small manual edit in the repository that a developer will need to change only as frequently as the releases of LCM change (or it can remain the same version number if the project dictates). Other features of the `lcm.sh` script include the automatic creation of the build directory for LCM in a standardized way, the use of the `wget` command to retrieve the source code for LCM from its GitHub repository (once again, eliminating the need to track that source code in the `fire_tpls` repository), and a clean-up step which removes the unnecessary zip file for LCM once installed. Sample output for the automatic installation of LCM using the scripts from the new `fire_tpls` repository can be seen in Fig. A-4 below.

```
PBEATA scratch $ ../../fire_tpls/scripts/std/install_tpls.sh
_SCRIPT_DIR = ../../fire_tpls
PACKAGE_DIR = /mnt/c/Users/PaulA/Desktop/Ubuntu/NERS590/FIRE/fire_tpls

******************************
*** Running install_tpls.sh ***
******************************

PACKAGE_DIR      = /mnt/c/Users/PaulA/Desktop/Ubuntu/NERS590/FIRE/fire_tpls
TPL_INSTALL_DIR  = /home/pbeata/installed_tpls

STD_SCRIPTS_DIR = /mnt/c/Users/PaulA/Desktop/Ubuntu/NERS590/FIRE/fire_tpls/scripts/std
BUILD_LCM = 1

 ~ lcm.sh called ~

Building lcm-1.3.1 ...

LCM_DIR = /home/pbeata/installed_tpls/common/lcm-1.3.1
BUILD_LOC = /mnt/c/Users/PaulA/Desktop/Ubuntu/NERS590/FIRE/fire_tpls/lcm/lcm-1.3.1_build

   START CONFIGURE AND MAKE
   END CONFIGURE AND MAKE

 ***warning: modified $HOME/.bashrc with 2 new lines***

 ~ lcm.sh completed ~
```

Figure A-4: Terminal output from the automatic TPL installation process

Another important consideration for this project was to change the default LCM build instructions and provide a more portable set of commands within the `lcm.sh` script to avoid using `sudo` on shared machines, such as Flux or Comet computing clusters or the University of Michigan CAEN computers. Using the default build instructions for LCM leads the user to employ the `sudo make install` and `sudo ldconfig` commands in the terminal during installation. This is a concern for the portability of the research software as LCM is a required TPL although

some networks may not grant the user `sudo`-level access. To handle this issue, the default `configure` step in the LCM build process was modified with a `--prefix` flag in the `lcm.sh` script which specifically redirects the default build location to a subdirectory of the `$TPL_INSTALL_DIR` discussed earlier (for the current user only to avoid the `sudo` issue). Then the `make` command and the `make install` steps follow without any need to use the `sudo` privileges in the process.

Similarly, to avoid the `sudo ldconfig` command, which is used to configure dynamic linker run-time bindings, a new `.bashrc_lcm` file for the `$HOME` directory is automatically created from the `lcm.sh` script. The `.bashrc_lcm` file for LCM, and in general for future TPLs named `.bashrc_<TPL>`, contains a list of environment variables which tell the system where the libraries and header files are located for a particular TPL. This file is populated only once during the installation of each individual TPL (i.e., one `.bashrc_<TPL>` file must be generated per TPL) and is utilized by adding a `source` command to the main `.bashrc` file in the user's `$HOME` directory. The addition of one `source` command per TPL to the main `.bashrc` file to load the environment variables in each individual `.bashrc_<TPL>` file is a lightweight method for ensuring that all libraries and headers are "discoverable" in the system while using the new research software.

A.4.2 TriBITS Build System

Installing the TPLs for the project was one important component of the project but it would not be useful unless it was integrated into the main project repository and build system. This section describes the inclusion of the TPLs in the TriBITS build system and also discusses how a local source code package was used to provide a simple unit test of the software system based on a small message-passing example. Typically, the TriBITS source code would not be included in

the main research repository; thus it must be cloned into the source directory before building. This step is straightforward and described for the user in the documentation so that there is no confusion about a broken TriBITS links. Once in a new build directory, the `fire_main/guide` subdirectory provides the script for the user to perform the `cmake` command. Figure A-5 contains the full build script which equips users with the proper `cmake` command flags to discover the installed TPLs (presently, only for LCM).

```
#!/bin/bash -e

rm -rf CMake*

source $HOME/.bashrc_lcm

cmake -DCMAKE_INSTALL_PREFIX=$HOME/rtfm \
      -DmyProj_ENABLE_TriBITS=ON \
      -DmyProj_ENABLE_TESTS=ON \
      -DTPL_ENABLE_LCM=ON \
      -DLCM_INCLUDE_DIRS=$LCM_DIR/include/lcm \
      -DLCM_LIBRARY_DIRS=$LCM_DIR/lib \
      -DmyProj_ENABLE_RTFM=ON \
      -DmyProj_VERBOSE_CONFIGURE=ON \
      ../fire_main \
      &> configure.out
```

Figure A-5: The build script for the main TriBITS project with flags for including LCM

Once the script shown in Fig. A-5 is submitted in the build directory for the project, the typical installation toolchain continues with the `make` command, followed by unit tests (via `ctest`), and finally the `make install` command to complete the process. Note that for the unit tests, the typical TriBITS unit tests can be accessed in the top-level directory simply by running the `ctest` command after the `make` step in that directory. However, the more interesting case for the current project would be to ensure that LCM was installed correctly and that any packages depending on LCM and its libraries are properly linked. Thus, by navigating into the `packages/RTFM` subdirectory of the build directory for this project, the user can run the `ctest` command locally (i.e., within `packages/RTFM`) to perform a simple message-passing unit test.

This test will fail if LCM was not installed and linked to the main TriBITS project properly. Successful completion of this single unit test is shown in Fig. A-6 after performing the previously discussed installation process on a personal laptop.

```
pbeata RTFM $ ctest
Test project
/mnt/c/Users/PaulA/Desktop/Ubuntu/NERS590/FIRE/BUILD/packages/RTFM
    Start 1: RTFM_send_message
1/2 Test #1: RTFM_send_message ................   Passed    0.06 sec
    Start 2: RTFM_listener
2/2 Test #2: RTFM_listener ....................   Passed    0.04 sec

100% tests passed, 0 tests failed out of 2

Label Time Summary:
RTFM    =   0.10 sec

Total Test time (real) =   0.13 sec
```

Figure A-6: Results of `ctest` in the `packages/RTFM` subdirectory

A.4.3 Discussion of Results

The Results section provided more details about the automatic TPL installation and the TriBITS build system; some additional discussion of the presented material follows here. First, it should be noted that the main installation script was designed such that the user is free to specify which TPLs should be installed as all might not be needed for a particular machine. To account for this situation, if the user intends to install only LCM, all they must do is declare the environment variable `BUILD_LCM=1` in the terminal during the installation process. This triggers the main installation script to call the LCM-specific script for installing this particular TPL (namely, `lcm.sh`). The extensible logic found in the `install_tpls.sh` script is shown in Fig. A-7.

```
# LCM
if [ "${BUILD_LCM}" != "" ] ; then
  ${STD_SCRIPTS_DIR}/lcm.sh
fi

...

# <TPL_name>
if [ "${BUILD_<TPL_name>}" != "" ] ; then
  ${STD_SCRIPTS_DIR}/<TPL_name>.sh
fi
```

Figure A-7: A sample of the `install_tpls.sh` demonstrating the TPL installation logic

Using this approach will allow the future developers to add more TPLs to this repository by following the logic used in `install_tpls.sh` whereby each new TPL is screened by an `if` statement to check for any `BUILD_<TPL>` environment variables equal to one, as with the LCM example shown here. Similarly, the current and future developers can use the `lcm.sh` script as a template for providing the automated installation steps for new TPLs. This method for installing the TPLs sequentially was derived from the approach used in the CASL/VERA project which has a similar TPL repository on GitHub called `vera_tpls` (https://github.com/CASL/vera_tpls).

In general, it is not considered good practice to silently add several lines to the main `.bashrc` file of a machine. Thus, each addition of a `source` command to the system's `.bashrc` file for the purposes outlined earlier is accompanied by a short warning in the terminal during installation: `***warning: modified $HOME/.bashrc with 2 new lines***`. One new line provides the name of the TPL as a comment and the second line is the actual `source` command; this eliminates the concern of blindly adding new content to the end of a hidden file like the `.bashrc` file. As a final measure of safety, the installation script first uses the `grep` command to ensure that there are no other instances of a particular `source .basrhc_<TPL>` command in the `.bashrc` file before actually writing to it. This

protects the user's `.bashrc` file from multiple additions of the same lines in the case of multiple installs of the same TPLs on one machine.

With regards to the system documentation, there should have been a more thorough consideration for prerequisite packages and libraries (such as `g++`, `git`, and `cmake`). This was discovered after trying to install the research software on a machine with a fresh install of Ubuntu. Sometimes these critical components are not included in certain systems and a new developer on the project might not know whether they exist on their current machine. Thus, one feature that was not addressed during creation of the documentation (but should be added in future work) was performing a check of prerequisites on a new machine for fresh installs of the product. At the very least, a list could be added to the documentation which tells the developer or user to use the flag known as `--version` or the `which` command for each prerequisite before starting the installation process just to ensure that the proper packages are available.

A.5 Conclusions

The contributions made during the development of this project's foundation focused mainly on providing the software infrastructure for the current research in developing a computational tool for real-time fire monitoring. While this work was not dedicated to implementing a new feature in the form of source code for a particular fire monitoring application, it was successful in the establishing the infrastructure for a successful research-level software development process by using trusted methods from scientific computing and the fundamentals of software engineering.

With these goals in mind, the work completed during this phase of the project produced a version control system for the new research project. Not only were two repositories simply

initialized for the work, but rather they were set up to encourage future use by considering potential additions of packages and TPLs, in a systematic way, at later stages of the project. With an understanding of the problem definition and initial requirements for the research project, a TriBITS build system was defined which included linking the required TPLs and packages. Additionally, a simple unit test of the build system, initial documentation, and an automatic TPL installation method were all generated from this initial investment.

The work presented in this technical report was designed with the future success of the ongoing research project as a motivating factor. The report outlined the main concerns that needed to be addressed in order to provide a foundation for the real-time fire monitoring project: portability was one desired feature of the main research project. Thus, the future research extending from this foundational work will include ensuring that these contributions such as automatic TPL installation will be sufficient for larger shared resources such as Flux and Comet in the upcoming stages. The methods used in this course project were employed with portability in mind and this extension should prove that this effort was worthwhile.

The main features of fire-monitoring system presented here are expected to encourage good software engineering practice in the future stages of the ongoing research. The use of version control as a tool for keeping a revision history of future project tasks will be a particularly important component. Specifically, the TriBITS build system should continue to provide the current and future developers with a stable repository which can be added to as more novel source code is generated. By using the `packages` subdirectories and demonstrating a templated way for installing multiple TPLs, the future development of this research software will now have the proper structure for handling additions of each kind.

With the addition of new in-house packages and more TPLs, the future developers can add to the newly formed documentation collection as well. However, a feature that was lacking from the foundation developed in this project was the provision of an actual policy document to help guide the architecture and workflow in the future software construction effort. The project needs policies in place that will promote this continued maintenance of the documentation by future developers in the research group and encourage the ongoing use of version control for new source code generation.

One crucial component of the software development which must be addressed and was not considered in the current report was the concept of thorough unit testing. Most of the work was focused on organizing the new research into a stable collection of code, scripts, and documentation. Due to the current stage of the research, there was not much actual source code to perform unit testing with and the only unit test used in the process was a simple check to ensure that one package could successfully use LCM. Thus, a major focus of the future work will be to develop unit tests alongside the generation of novel source code during the upcoming stages of the research. This process will require the discipline of the development team to provide proper unit testing in order to ensure the reliability of the newly developed software.

A.6 References

[1] M. Karter Jr., "Fire Loss in the United States During 2013", *National Fire Protection Association Journal*, 2014.

[2] Federal Emergency Management Agency, "Firefighter Fatalities in the United States in 2011", 2012.

[3] N. Daniel and G. Rein, "The Fire Navigator: Forecasting the Spread of Building Fires on the Basis of Sensor Data", *Fire Protection Engineering Extra*, no. 3, 2016.

[4] A. Huang, E. Olson and D. Moore, "Lightweight Communications and Marshalling for Low-Latency Interprocess Communication", Massachusetts Institute of Technology, 2009.

[5] Peacock, R., Jones, W., Reneke, P., & Forney, G. (2005). CFAST–Consolidated Model of Fire Growth and Smoke Transport (Version 6) User's Guide. *NIST Special Publication, 1041*.

[6] A. Orso, "Software Development Process: Introduction and Overview", https://classroom.udacity.com, 2016.
[7] B. Kochunas, "Software Engineering Practices & Development Workflows", Ann Arbor, MI, 2016.