GPGPU enabled CFD simulation for fully coupled fire and evacuation modelling

Markus Sauter

A thesis submitted in partial fulfilment of the
requirements of the University of Greenwich
for the Degree of Doctor of Philosophy

July 2015

DECLARATION

I certify that this work has not been accepted in substance for any degree, and is not concurrently being submitted for any degree other than that of Doctor of Philosophy being studied at the University of Greenwich. I also declare that this work is the result of my own investigations except where otherwise identified by references and that I have not plagiarised the work of others.

....
Markus Sauter (PhD Student)

Date:

....
Prof Edwin Galea (Supervisor)

Date:

....
Dr Angus Grandison (Supervisor)

Date:

ACKNOWLEDGEMENTS

First and foremost I would like to say thank you to Professor Ed Galea and the University of Greenwich for giving me the opportunity as well as the financial support to complete this degree.

I would like to thank my supervisors, Professor Ed Galea and Dr. Angus Grandison. Both have over the years provided me with endless support and advice and made sure that I did not get discouraged . Without their good guidance and mentoring this Phd would not have been as successful and enjoyable. Looking back, I could not have asked for a better supervisors.

I furthermore have to thank my colleagues Dr. John Ewer and Dr. Peter Lawrence who always had a minute to spare to shed some light into the endless SMARTFIRE and EXODUS code maze, as well as Professor Mayur Patel who always had an open door and good advise if I wanted to chat. A big thank you also goes to my, in the beginning fellow Phd student and now, brilliant post doc Dr. Andrew Kao with who I always had the most productive and memorable conversations down in the pub.

Thanks to my lovely, soon to be wife Kristin who over the years never lost faith in me and who always was my support away from family in Germany. Without her blessing I never would have been able to finish my Phd. My biggest thank you goes to my parents Marianne & Franz, who since I can remember have always supported me financially as well as being perfect role models. My father was always the successful hard-working person that I wanted to become and I therefore want to dedicate this thesis to him. Thanks Dad!!!

Last but not least I want to thank all my friends and the rest oft the FSEG family for making this journey something I will always enjoy remembering and be proud of, for the rest of my life.

A person who never made a mistake never tried anything new
Albert Einstein

ABSTRACT

Traditionally fire and evacuation models are run independently of one another to ascertain two key building safety parameters: ASET (available safe evacuation time) determined by the fire spread; RSET (required safe exit time) determined by the evacuation model. A building can be deemed to be safe if RSET <ASET. A more advanced method is to couple the models together to give a dynamic fire environment superimposed on the evacuation. This has typically been achieved using a one way couple where the fire is predetermined prior to the evacuation. A more advanced two-way couple can be used in scenarios, where the evacuation behaviour effects the fire environment, e.g. opening/closing doors by agents, extinguishment of fire by agents etc. Presently the time taken to run these simulations is dominated by the CFD fire model.

The problem with two-way coupling is that every change requires a recalculated CFD environment and as the evacuation simulation is based on Monte Carlo methods this leads to multiple calculations to achieve statistically significant results. A complete GPGPU implementation (solver, coefficients and other dependent variables) of the CFD based fire model has been developed which leads to a substantial execution speed-up. Many previously reported implementations are limited to the matrix solver and are thus limited to the speed of the host calculating the coefficients and thereby returning modest overall speedups. The speed-up gained through the parallel implementation enables the practical use of the two way coupling. The key point of the two way coupling is that the agents in the evacuation model dictates the way the CFD code calculates its values. By letting the agents directly interact with the geometry it eliminates the element of making assumptions when events happen and drastically reduces the number of required simulation runs for all permutations.

CONTENTS

FIGURES

TABLES

Glossary

ALU Arithmetic Logical Units. 28

AMD Advanced Micro Devices. 34

API Application Program Interface. 31

ASET Available Safe Egress Time. 11

BE Boundary Element. 48

BFC Body Fitted Co-ordinates. 13

CAA Computational Aeroacoustics. 18

CAD Computer Aided Design. 14

CFD Computational Fluid Dynamics. 2, 184

CHT Conjugate Heat Transfer. 13

CO Carbon Monoxide. 10

CPU Central Processing Units. 28

CUDA Compute Unified Device Architecture. 34

CV Control-Volume. 48

DDM Domain Decomposition Methods. 28

DE Differential Equation. 46

DNS Direct Numerical Simulation. 62

DSP Digital Signal Processors. 33

FD Finite-Difference. 48

FE Finite-Element. 48

FED Fractional Effective Dose. 15

FIH Individuals cumulative exposure to radiative and convective heat. 120

FMAD Fused Multiply-Add. 94

FPGA Field Programmable Gate Arrays. 33

FSEG Fire Safety Engineering Group. 20

FSEs Fire Safety Engineers. 1, 156

FV Finite-Volume. 48

GPGPU General Purpose GPU. 31

GPU Graphics Processing Unit. 4

GUI Graphical User Interface. 12

HCl Hydrogen Chloride. 10, 184

HCN Hydrogen Cyanide. 10, 184

ILP Instruction Level Parallelism. 27

ISA Instruction Set Architecture. 35

JOR Jacobi Over Relaxation. 48, 73

LAN Local Area Network. 8

LES	Large Eddy Simulation. 10
MC	Monte Carlo. 3
MIMD	Multi Instruction Multi Data. 26
MISD	Multi Instruction Single Data. 26
MPF	Message Passing File. 109
MPI	Message Passing Interface. 28
OpenACC	Open Accelerators. 34
OpenCL	Open Computing Language. 33
PBA	Performance Based Approach. 11
PDE	Partial Differential Equations. 8
RSET	Required Safe Egress Time. 11
SIMD	Single Instruction Multi Data. 26
SIMPLE	Semi Implicit Method for Pressure Linked Equations. 5
SM	Streaming Multiprocessors. 31
SOR	Successive Over Relaxation. 73
SP	Scalar Processors. 31
TLP	Thread Level Parallelism. 27
TPC	Thread Processing Clusters. 31

Chapter 1

INTRODUCTION

Fire related incidents in 2013/2014 included 322 fatalities and 9,748 non-fatal casualties, as well as an estimated 8 billion worth of damages to homes and business in the UK alone [1]. According to these statistics many of the incidents are preventable by closer complying with the existing regulations. In order to improve regulations and to gain more insight into the complex physical and chemical processes involved in fires, more research has to be conducted. Unfortunately by the destructive nature of fires it can be very difficult to collect accurate data, which on the other hand is necessary to gain more insight into the processes involved from ignition to extinction to prevent future incidents from happening. Fire experiments offer an alternative but are very costly to carry out. Nowadays fire modelling offers a very practical and inexpensive alternative for Fire Safety Engineers (FSEs) to simulate fires and all the processes involved.

The models have various purposes, to reduce the requirement to run actual experiments in order to save time and money as well as to detect exception cases which could have been missed otherwise. In that sense, Fire Safety Engineering represents a very challenging research field, the modelled cases are usually large in size and complex in terms of processes involved e.g chemical or physical reactions. Another problem is that real life experiments are often restricted by health and safety rules and engineers have to rely on accurate models that produce realistic results. If this wasn't already challenging enough fire safety engineers often need more than one model to represents different parts of the problem, therefore it is often necessary to use a Computational Fluid Dynamics (CFD) based fire model in combination with an evacuation model. This thesis will show a new concept for two of these models to efficiently interact and to eliminate as much human errors, required simulation runs and interaction within the modelling process. A so called two-way coupling will be developed. An evacuation model will be used in dynamic combination with a CFD fire simulation model. Instead of using pre-calculated fire scenarios in the evacuation model, the evacuation model will be used to control the CFD model throughout the simulation run. This means the CFD simulation will provide the evacuation simulation with hazard data for every upcoming time step. The two-way coupling will therefore be able to incorporate changes in the geometry e.g door changes immediately.

1.1 Research Aims And Objectives

The primary aim is to explore a full two way coupling of a fire and evacuation modelling tool to examine the interaction of all the processes involved in a fire and evacuation simulation. In order to establish the coupling each of the software models used, the CFD model and the evacuation model, have to be enhanced to incorporate the coupling processes. The most important change will be a new communication process that allows both software pieces to send and receive instructions from each other. When this is established, the evacuation model needs to be extended to give the agents further abilities to interact with the environment. In the CFD model, a mechanism to dynamically change boundary conditions is needed as well as the ability to send back the updated CFD data to the evacuation model (every time step).

Furthermore one of the CFD model downsides, the execution speed, will be improved. This will make the process a more accessible tool for fire safety engineers where one does not have to sacrifice accuracy (reduced mesh sizes) or scenario complexity in order to get results in a reasonable amount of time. These improvements will be achieved by the use of parallel processing techniques (in the form of advanced graphics processor hardware) to accelerate the CFD based fire simulation.

The necessary steps involved will be

- to port the existing CFD code, written in C++, to a faster hardware accelerated language.
- to develop a communication system between the two software tools.
- to extend the capabilities of the evacuation model to be able to control the CFD model.

1.1.1 Research Questions

- Will a fully two way coupled model result in more realistic results? - As it currently stands one way coupled models take the results of a CFD simulation and apply those to the evacuation model on a time step basis. If one wanted to incorporate changes in the fire scenario he would have to stop the evacuation model, re-run the CFD model and re-enter the fire data into the evacuation model. These techniques all require assumption when things will be happening, the fully dynamic coupling will apply changes on different dynamic triggers e.g. an agent decides to open a door because the temperature in an area reached a critical level. By not relying on predefined assumptions any more, more realistic simulation outcomes could be achieved. The evacuation model will be the decision maker for environmental changes and trigger boundary updates in the CFD code which incorporates the changes and recalculates the scenario.
- Why is speed so important? - Faster execution times are a desirable feature of a piece of software. It not only enables the fire engineer to perform more tasks in a specific time frame, it also enables additional features which would result in an exponential increase in computing power or execution time and making it therefore impractical to use. One example is that of the Monte Carlo (MC) based EXODUS model, which needs a great number of simulation runs to get statistically significant results, by changing the CFD environment during runtime a new set of MC-simulations is required for every change. Taking all the permutations of geometry changes into account which could be happening this will quickly constitute weeks of simulations if not even an impossible task to model.
- How will the speed-up be obtained? - The CFD model will be ported onto a new hardware execution platform, which is a modern Nvidia graphics card (GPU). In order to get the code to execute a rewrite of the existing code will be required. The more of

the existing model that can be ported the better the overall performance increase will be because the main known bottleneck will be data transfers to and from the Graphics Processing Unit (GPU). The main focus will be to optimize memory access patterns and to make full use of the available hardware resources.

- What will be the magnitude of the speed-up? - As mentioned above, the speed-up will be closely related to the proportion of the code that will run simultaneously on the GPU and the kind of problem that will be modelled. There will be many other factors influencing this performance outcome, but a reasonable expectation for a practical model would be a speed-up, of somewhere in between 10x to 40x speed-up regarding overall execution time.
- Are all parts of the CFD equally likely to execute efficiently on the GPU? - The general assumption is that all the cell/ face based calculation which will include the majority of the computational domain will run equally fast. The only exception will be the radiation model which will only calculate cell values that will fall on the rays assigned to each cell. The critical part here will be that rays will cross multiple partitions requiring a lot of communications between partitions.

1.2 Thesis Outline

This thesis consists of 9 chapters. Chapter 1 sets the background for undertaking this research and explains the motivation behind it.

Chapter 2, gives an overview of CFD modelling in general and specifically the tools used in this thesis. Previous work and literature relevant for this research will be introduced. Furthermore parallel techniques mentioned in the research question and the state of the art for current hardware and software technology are reviewed.

Chapter 3 introduces the equations and numerical methods used by CFD models to solve fire modelling problems. The general conservation equation is detailed and methods to solve these equations by discretisation will be described. The SIMPLE solution method for the coupled equations of heat, mass, momentum and pressure will be given. Turbulence modelling will be described in terms of the Reynolds average approach. Furthermore the radiation model, boundary conditions and the solver method will be introduced.

Chapter 4 describes the methods necessary to implement the parallel concept, particularly the mesh partitioning and the so called halo concept will be discussed in detail. Furthermore two solver methods are introduced, a classic approach and a performance improved version, which will be used in this thesis.

Chapter 5 describes the CFD test cases used to show the agreement of results from the serial and CUDA implementation of SMARTFIRE. And explains the differences in the two implementations.

Chapter 6 details the implementation of the coupling between the two software tools. Synchronization and control mechanisms are described which enable a full two way coupling.

Chapter 7 describes the test cases used in this thesis to show the different results obtained by using the two-way coupling and shows the results compared to non-coupled cases. Differences between a one way coupling and a full two way coupled model will be given.

Furthermore constrains are shown that illustrate the necessity of the two way couple for a fire safety engineer.

Chapter 8 contains conclusions from the work mentioned in previous chapters. Furthermore the research question will be re-evaluated, to see if all questions have been answered.

And finally in Chapter 9 suggestions for areas of further work will be suggested.

Chapter 2

BACKGROUND AND LITERATURE REVIEW

In this chapter the background to the problems addressed in Chapter 1 will be detailed. A review of CFD fire modelling is given. The literature directly relevant to the different areas which will be addressed later in this thesis is described. An overview of existing evacuation and CFD models will be given, as well as various combinations of coupled models. One of the main issues of this thesis is the performance of the serial CFD code. Therefore methods to parallelise the original code and improve the performance of the parallel version will be discussed, from a hardware point of view as well as from a broader parallel computing perspective.

2.1 CFD Modelling

CFD modelling attempts to represent the actual physics behind fluid flow in a computer model. The modelling process generally consists of the discretisation of the problem domain and solving it using the Navier-Stoke set of equations [2] in combination with additions to include turbulence and radiation. The first CFD models came up about 80 years ago but used a much simpler approach [3] compared to models used nowadays. The main limitation of the earlier models was not the actual mathematical complexity, it was the lack of computing power or computers in general available at the time. Even today the complexity of the problem, although vastly more complex than in the past, is limited by the available computational resources.

The partial differential equations (PDE) used by CFD codes to calculate the individual variables normally cannot be solved analytically and must therefore be solved numerically by discretising them [2]. This creates thousands to millions of smaller sub-domains, control volumes or cells. The computational difficulties come from the fact that to get the solution to converge, many iterations are required per time step. And for an actual transient problem case many time steps are needed [4].

Most of the computational intense variable calculations require a cell-based solution. Different variable states (newest, last and old) have to be stored in a vector or even matrix form, which requires considerable memory and CPU power [5]. Additional memory is needed to include turbulence phenomena which require mesh refinements to get more accurate results [6] or trans-sonic/ super-sonic flows.
Although fire models do not require all of these extensions e.g trans- and super-sonic flow , additional models like radiation are necessary but they are actually not part of Navier-Stokes equations. The radiation model plays an extremely important role in modelling realistic fire scenarios as thermal radiation can dominate the heat transfer. Various radiation models will be discussed later in this thesis in Section 3.1.6, but generally the more accurate the model the more computing power it will require.

The earliest publications where CFD models were used to model fire situations came up in the early 1980s [4; 7–12] and have since then become more and more popular so that fire safety engineer nowadays can access a large portfolio of modelled cases [13–22]. These models all contributed to a much greater insight in the actual physics and dynamics behind fires. The biggest constraint with CFD and fire modelling respectively is the time needed to calculate complex scenarios, over the years new techniques have been investigated e.g parallel processing in various forms. Variations include transputers [23], using PCs connected via a Local Area Network (LAN)[12; 24], multi-core CPUs in general [25; 26] or as described in this thesis, GPU-hardware acceleration.

Another hardware independent approach is using group solvers, it groups cells into logical sets therefore specific solver criteria can be set for these groups which can significantly reduce the execution time [27]. Group solvers essentially try to priorities groups and allocate resources to groups that are computationally more intense than others. Another method of reducing runtime is the concept of using a hybrid field/zone model [28] where parts of the computational CFD domain are replaced by a zone model.

Turbulence plays a very important role and will be included in any realistic scenario when modelling fluid dynamic problems, both in transport/mixing and energy transfer [29]. The complexity of modelling turbulence comes from the fact that it randomly fluctuates and its occurrence in three dimensions over different time and length scales. As with many stochastic modelling approaches it can be assumed to have an average velocity for the fluid along with the deviations of it. Reynolds started modelling turbulence by splitting the fluid motion into a mean and random component [30].

The second component is random, its cumulative effect equals zero, therefore it is valid to consider the mean component of the motion. The next step is now to substitute this mean-random representation of the velocity into the Navier-Stokes equations which gives a new almost similar system but with new unknown terms [31] known as the Reynolds stresses. These stresses are now part of the flow itself, more precisely the momentum equation of the mean flow instead of the more familiar viscous stresses. Therefore the turbulent fluctuation still has an effect on the mean flow and shows a difference compared to non-turbulence cases. The challenge now is that the number of unknowns exceeds the number of equations present making it an underdetermined system, the research that is still done on turbulence mainly focuses on determine these unknowns.

SMARTFIRE uses a $k - \varepsilon$ model [31] for these unknowns. The method was first used by Boussinesq [32] when looking at the transfer of momentum caused by Reynolds stresses to be actually caused by an ”eddy-viscosity”, in the same way as the usual viscous stresses. Based on this transport equations for the turbulent kinetic energy k and dissipation rate ε can be created and solved along with the remainder of the Navier-Stokes equations.

Another method of separating the velocities is by their scale. One part will be directly resolved by cell resolution while the other part relies on the ”eddy-viscosity”, mentioned above, for the smaller scales. This is called a Large Eddy Simulation (LES) and will be explained in detail in Section 3.1.5.3. The work was originally performed by Smagorinsky [33]. LES performs better than the $k - \varepsilon$ method if the resolution is fine enough [34] but significantly increases the computational overhead.

Fire safety engineers are interested in simulating or reproducing realistic fire scenarios with the presence of gaseous species i.e fuel, oxidant and product concentrations or more complex combustion/ pyrolysis processes releasing toxic threats i.e carbon monoxide (CO), hydrogen chloride (HCl) or hydrogen cyanide (HCN). These gases constitute the biggest threat to people within burning enclosures [35; 36]. The main reason for performing fire safety engineering is to save lives by optimizing building designs, procedures, protocols or materials used or to get more insight into past incidents.

The transport of the species mentioned above can easily be incorporated into the Navier-Stokes equations, especially if simplifying assumptions are made for a basic mass/volume fraction treatment. The problem actually lies on the creation side of the species, the model sources include the physical and chemical properties and processes necessary to accurately represent the phenomena [37; 38]. Specifically the time steps and the mesh size play an important role in the modelling process as they can be significantly smaller than for the other variables. Therefore simplification methods have been used such as assuming infinitely fast one step chemistry thereby allowing the use of mixture fraction [39–42]. Although accurate results for simple fuel/product/oxidant concentration can be achieved more detailed species formulations e.g soot production/ flame extinction cannot be obtained. To solve this constraint Floyd [43] came up with a list of extensions although it still remains a more simplified model than the actual chemical process.

Combustion will not be included in this thesis, therefore fires will be represented by heat (enthalpy) sources. This is backed up by a number of experiments that have been conducted over time which give a benchmark how accurate the models are [17; 44–46]. The challenge will always be to find an intermediate way between accuracy and the corresponding increased processing time and the need to run detailed models for cases where the actual chemistry is the driving force of the flow dynamics e.g fire proliferation and suppression [47]. Although processing power is less of a problem nowadays than in earlier days of modelling (as predicted by Moore [48]), the increase of processor speeds is not happening as fast any more as it used to [49]. The reason being the increasing difficulties with the transistor density on the silicon wafer, especially the energy supply and heat development.

Finally it must be added that despite the complex mathematics and physics behind the fire models, simulation tools still must be practical for engineers working in the industry to produce accurate results in reasonable time frames without the need of profound knowledge of how to manually optimize the models themselves. End users typically come from fire safety engineering, science and architecture/building design backgrounds.

2.2 Modelling Tools

Modelling tools are increasingly becoming a part of performance-based analysis to assess a level of detail similar to that from experiments but without the associated incurring costs. This section will introduce the most common egress and fire models also stating there different capabilities.

One category of models use predefined sets of parameters to specify how fast evacuations must complete. These parameters are typically set by regulators e.g for public buildings in the UK (2.5 minutes) or public transport (90 seconds) [50]. The other category of models which try to eliminate the shortcomings of using predefined parameters follow the so called Performance Based Approach (PBA), see Section 2.2.3. Although there are many different flavours of this approach, they essentially all follow the concept that there is an Available Safe Egress Time (ASET) and a Required Safe Egress Time (RSET) [51] where RSET <ASET.

ASET could be a time until a critical level is reach e.g smoke or temperature and RSET could be the time required to vacate a building. In order to achieve accurate ASET and RSET values multiple evacuation and fire scenario combinations have to be assessed. This procedure means that the evacuation parameters are individually determined for a specific geometry and scenario.

2.2.1 CFD Fire Models

There are also a number of CFD fire models available to date which include SMARTFIRE [52], FDS [53], FIRE3D [54], RMFIRE [55], SOFIE [56], SPLASH [57], CFX [58], FIRE [59], JASMINE [60], PHOENICS [61] and ALOFT-FT [62]. Where some of them can be used to create fire output data to feed into the egress models, which will be mentioned in the next section. The following list will give an overview of these models, showing their capabilities and limitations:

- SMARTFIRE: Fire field model, including automated mesh generator and interactive Graphical User Interface (GUI). The model can simulate fire, smoke, thermal radiation,gaseous combustion and toxic products in complex structures. It can also be run in a parallel version.
- FDS: The Fire Dynamics Simulator is a LES model for low-speed flows, it models turbulence, as well as combustion and radiation. The emphasis is on smoke and fire heat transportation for spreading heat from a fire source. FDS also offers a parallel execution concept.
- FIRE3D: CFD model to simulate jet and buoyant turbulent diffusion, combustion, thermal radiation and spray-flame interactions in compartments and open atmosphere.
- RMFIRE: A two dimensional field model for the transient calculation of smoke movement in room fires.
- SOFIE: Simulation of Fires in Enclosures, is a model developed by a number of European Universities, governments and research institutes. The model uses a non-orthogonal curvilinear coordinate system, a laminar flamelet model, fire spread models and a radiation model which uses the Discrete Transfer method.
- SPLASH: A quasi-field model describing the interaction of sprinkler sprays with fire gases.

- CFX: Developed by ANSYS Inc. General-purpose fluid dynamics software, applicable to dispersion, fire and explosion within internal and/or external environments. The model offers the use of structured and unstructured meshes, a coupled multigrid solver, gaseous combustion models (including EDM, flamelets), turbulence models (including LES and DES), radiation models (including Monte Carlo and Discrete Transfer) and Conjugate Heat Transfer (CHT). The model can also be run in parallel.
- FIRE: CFD model with water sprays, coupled to solid or liquid phase fuel to predict burning rate and extinguishment.
- JASMINE: A CFD or field model developed by the UK Fire Research Station (FRS), for predicting consequences of fire to evaluate design issues such as the assessment of smoke ventilation design and/or interactions with HVAC and other fire protection measures. JASMINE offers the capability to model combustion and solid body heat transfer as well as a six-flux radiation model. It is limited to orthogonal meshes which makes it difficult to model complex geometry.
- PHOENICS: PHOENICS simulates fire progression, smoke and pollutant dispersion in steady state or transient conditions. It comes with a variety of turbulence and gaseous combustion models and includes the capability to use Body Fitted Coordinates (BFC) grids. As with JASMINE, radiation is modelled using a six-flux radiation model which is not surprising as PHOENICS was the basis of the development for the JASMINE code.
- ALOFT-FT: Calculates the rise and dispersion of smoke from a large outdoor fire blown by a non-zero wind.

2.2.2 Evacuation Models

There are a number of evacuation models currently available which include, EXODUS [63; 64], PATHFINDER [65], GRIDFLOW [66; 67], FDS-EVAC [68], MASSEGRESS [69], LEGION [70], PEDGO [71], EVI [72], SIMULEX[73; 74], EVACNET4 [75], STEPS[76], MassMotion [77] and BFIRES [78].

The brief review of these models will focus on capabilities like, modelling methods, model structure, occupant behaviour and movement, use of fire data, output, CAD geometry input, visualisation, validation studies, limitations. As this thesis focuses on the evacuation process of buildings models used in the maritime and aviation field will be disregarded. The following list shows the capabilities of the models mentioned above [79].

- Exodus: Incorporates occupants performing actions, in addition to movement towards a specified goal, exiting a building, also decision-making and actions performed due to conditions in the simulation. The grid structure is represented by small grid cells which are used by the occupants to move from point A to B. In this process each agent acts as an individual, using different rules and conditional behaviours. These conditional behaviours are statistically determined, allowing variations in the different simulation runs. The movement model is a probabilistic density correlation where each individual gets a speed and flow assigned depending on the density of the space. Fire data can be imported from other tools allowing to perform specific fire actions at specific times. Geometry data can be imported from Computer Aided Design (CAD) files. Data is represented in a two dimensional way allowing the user to see bottlenecks in the simulation. The model input parameters have been validated against fire drills or other people movement experiments and trials.

- Pathfinder: Incorporates movement towards a specified goal, exiting a building. The grid structure is represented by small grid cells which are used by the occupants to move from point A to B. It does not incorporate behaviour, only the movement aspect. The movement model is a probabilistic density correlation where each individual gets a speed and flow assigned depending on the density of the space. The model also cannot import fire data, and simulations can only be run in drill-mode. Geometry data can be imported from CAD files. Outputs are represented in a two dimensional fashion, in the same way as EXODUS does.

- Gridflow: Uses a partial behaviour model, which primarily calculates occupant movement, but begins to simulate behaviours as pre-movement time distributions, agents characteristics, overtaking, smoke and smoke-effects. Space is represented as a continuous network applying a two dimensional space. Agents have implicit behaviour. The movement model is a probabilistic density correlation where each individual gets a speed and flow assigned depending on the density of the space. This model cannot import fire data and simulations can only be run in drill-mode. Allows the input of CAD files. Visualisation can be made in two in three dimensional view. Model has been validated against fire drills or other people movement experiments, trials and literature on past evacuation experiments.

- Fds-evac: Model offers agent based simulation of humans. The movement is represented by a panic model. Scenarios are defined by text based config files. Visualisation and data post-processing is done using the smokeview software package. Fire data can be imported by using the fractional effective dose (FED) sub-model. Fds-Evac has also been thoroughly validated against various scenarios [80].

- Massegress: Is a multi-agent simulation system to model building occupants as individual agents equipped with sensors, brains and actuators for egress analysis. Capable of conducting simulations for multi-storey buildings. Offers CAD file import to define geometries. Offers individual decision-making for agents as well as interaction among individuals (social interaction) and special group (crowd and environment) interactions. Model takes into account, crowd density, environmental constraint, perceived emotion and tension, queuing behaviour, herding behaviour and bi-directional flow [81].

- Legion: Uses a behavioural model, representing occupants actions in addition to movement. Each pedestrian is represented as a 2-dimensional object which moves in a 2-dimensional space. Behaviour is represented by an artificial intelligence model. The movement uses two models, a probabilistic density correlation and a conditional movement model where movement throughout the building is dependent upon the conditions of the environment, the structure, the other evacuees, and/or fire situation. The model allows people to import fire data at certain times throughout the simulation. CAD data can be imported. Visualisation can be produced in two or three dimensions. Model has been validated against fire drills or other people movement experiments, trials and other models.

- Pedgo: The model is a behavioural simulation model, decisions and movement of every single person is simulated. PedGo is able to model multi-agents in discrete time and space. The floor plan of the investigated geometry is subdivided into quadratic cells $(0.4mx0.4m)$, which represents the space a single person occupies while standing. Agents are able to evade obstacles and other agents [82].

- Simulex: Uses a partial behaviour model, which primarily calculates occupant movement, but begins to simulate behaviours as pre-movement time distributions, agents characteristics, overtaking, smoke and smoke-effects. Space is represented as a continuous network applying a two dimensional space. Agents have implicit behaviour. Movement is represented by a inter-person-distance model which forces the agent to always keep a certain distance from other agents, objects and components of the building. Simulations can only be performed in drill mode and no fire data can be imported. Geometry data can be imported from CAD files. Visualisation can be made in two or three dimensional view. The model has been validated against fire drills, other people movement experiments and trials, as well as against literature on past evacuation experiments.

- Evacnet: Evacnet is a flow-based approach which models a crowd as continuous flow of fluid by neglecting the features of individuals. Flow-based models are mainly useful in estimating the flow of movement/evacuation process for huge and dense crowds. The environment is abstracted into a mesh, consisting of nodes and arcs. The nodes represent the physical structures, such as rooms, stairs, and hallways. The node capacity is determined by the number of occupants the structure may accommodate. The arcs represent the passageways for the occupants flow. Similarly, the arc capacity needs to be determined to limit the amount of human traffic that can traverse the passageways in certain time periods. Another parameter to be determined is the traversal time, which denotes the time an occupant flow takes to traverse the arc. A flow-based model relies on a specialized algorithm, used in solving linear programming problems with network structure, to generate optimal evacuation plans. Results can be used to determine the shortest evacuation time based on the network structure and corresponding physical constraints. The model does not take any behavioural aspects into account besides movement. Movement is also greatly simplified no vector fields and differential equations are used, i.e. the speed of the crowd is treated as a piece-wise constant [83].

- STEPS: Is a model designed to simulate pedestrian movement under both normal and emergency conditions. It employs a modern agent-based approach which predicts the movement of individuals in space. STEPS uses the principles of cellular automata. The model can handle complex emergent modes of behaviour which arise from simple deterministic and non-deterministic principles. Agents attributes include walking speeds, awareness, patience, associations and pre-movement times. The model has been verified and validated by comparison with analytical solutions, internationally-accepted design codes and full-scale testings [76].

- Bfires: Uses a behavioural model, representing occupants actions in addition to movement, plus risk assessment capabilities. The grid structure is represented by small grid cells which are used by the occupants to move from point A to B. Behaviour is modelled using a set of probabilistic rules or conditions. Movements are represented by the users inputs for speed, flow and density for certain parts of the geometry. The model also allows the user to input fire data at certain times throughout the simulation. The model does not allow the user to import CAD data but offers a geometry building environment. The model also does not offer visual outputs. Furthermore no indication of validation of the model is provided.

2.2.3 Coupled Models

The primary objective of fire safety engineering must always be to save life and minimize injuries caused by fire related products or evacuation procedures, followed by the commercial aspect of making it an inexpensive way of replacing trials and real life testing to an agreed level of safety. Over the years, approaches to improve procedures have changed, from applying simple assumptions and quantifying results, collected from surveys and drills, to sophisticated computer models to approximate outcomes as realistic as possible.

One of the simplest methods are ”Required Safe Egress Time” and ”Available Safe Egress Time” (RSET and ASET). The general idea behind RSET is that people in a fire situation will require a specific time to evacuate, which can be pre-calculated. At the same time ASET is calculated. If the simple relationship ASET>RSET is fulfilled it is assumed that all occupants can evacuate safely. The concept was first used 1975 by NIST [84]. The occupants adapt a 'robotic' behaviour whilst moving towards the exits, intervening activities are not taking into account, e.g. slowing down due to their physical condition or counter-productive, unsafe, or seemingly unreasonable behaviour [51]. The two major problems with the concept are that the evaluation is highly subjective, conditions must be quantified, and evaluated afterwards against the computer-calculated results. As well as the complex interaction between physical and psychological variables which cannot be derived from experiments with humans as these would be potentially harmful.

The next logical step towards more realistic results is to improve the quality and capabilities of the available computer models. Different fire scenarios were introduced, enabling insight into new scenarios where life threatening outcomes were detected that would have stayed undetected until a real incident would have happened. The Fire Safety Engineering Group of the University of Greenwich [85] and the VTT Technical Research Centre of Finland [68] were amongst the first ones to introduce coupled CFD-evacuation model. The evacuation model (buildingEXODUS [64; 86]) reads in hazard data created by FSEG's CFD model (SMARTFIRE [52][18]) and CFAST [87]. The hazard data is averaged over predefined zones which have to match in both models. The hazard zones consist of two layers, represented by two values one at 1.7m (head height) and 0.5m (knee height). These two levels have an effect on the evacuees physical and psychological state and affect the walking and crawling abilities which results in slower evacuation times, injuries or even fatalities. The coupling offers the possibility to introduce different hazards e.g. toxic products, heat (including radiation) and smoke. The results of the coupled simulation showed that the fire conditions have a direct effect on the occupants physical condition and therefore the simulation results, making them more realistic. Another example of a coupled fire model is CRISP [88], a Monte-Carlo based model for entire fire scenarios (fire zone model). The sub-models represent physical 'objects' including rooms, doors, windows, detectors and alarms, items of furniture and hot smoke layers. Coupled models can not only be found in the field of fire modelling, they can also be used to model other physical problems e.g acoustic problems, as Djambazov showed by coupling a CFD and Computational Aeroacoustics (CAA) code where the coupling used a CFD code as the base to reduce the implementation work as well as a CAA model [89]. Another kind of coupling processes, although more manual, are so called Performance-Based Codes, where a set of defined goals and conditions (performance-based document) get assessed and potential hazards get identified. Solutions are developed using a suitable calculation method e.g a computer model. Afterwards, these outcomes get assessed again and a decision can be made if the chosen solution was suitable, or if another solution method is required [90–92]. This process allows performance-based documents to be implemented and insures that their goals are met.

However even with a coupled version the evacuation scenarios are still ”static” which means the models do not take into account that occupants could interact with the geometry or fire which could affect the outcome. Dynamic geometrical changes will often change the ventilation inside a building and will for example occur by opening doors, collapsing walls or shattering windows. FDS-Evac eludes to the possibility of a full two way coupling between the fire and evacuation model in their manual [93; 94] but there are no apparent examples of this capability. A two-way coupling was also described in the FireGrid project from the University of Edinburgh [95], which uses the BRE Monte-Carlo model CRISP [88], which includes a fully-coupled simulation of fire growth, smoke spread, active and passive fire protection systems, and the egress and interaction of people with the fire environment. The model couples sensors and fire modelling to achieve a more realistic outcome. However CRISP uses a zone model and therefore has several disadvantages in regards to modelling complex geometries and non uniform zones.

Although some models offer the capability to combine evacuation modelling with fire modelling, non of the models offers the possibility to look at scenarios where agents can influence the fire input directly. In order to enable the evacuation simulation to perform these tasks, a complete two-way coupling of egress and CFD model is required which will be discussed in detail in Chapter 6.

This thesis will focus on a fully two-way coupled fire and evacuation model (SMARTFIRE and buildingEXODUS) which will be able to allow the occupants in the evacuation model to interact with the environment, causing the CFD environment to constantly adapt itself to the changing scenario. SMARTFIRE and buildingEXODUS have been chosen mainly because of freely available entire source codes (only to FSEG students), the modern object oriented program design methodologies implemented in C++, the already existing one-way coupling capabilities, as well as the parallel execution capability for the full CFD model.

2.2.4 SMARTFIRE

SMARTFIRE is an advanced CFD fire simulation environment (C++) developed by the Fire Safety Engineering Group (FSEG) at the University of Greenwich [96][52].

The SMARTFIRE software package consists of a desktop fire-laboratory enabling rapid turnaround of sophisticated fire simulation analyses, including problem specification tools, execution and analysis tools which allows the user, not very familiar with the background of modelling physics problems, to concentrate on design issues rather than CFD details. This is mainly achieved by embedding as much expert knowledge as possible into the software, to reduce the possibility of fundamental errors being introduced into the scenarios.

The main four components are:

- The core CFD engine
- Graphical user interface (GUI)
- Automated meshing tool
- Intelligent control system

The interaction between the components mentioned above can be seen in Figure 2.1, where the CFD engine represents the core of the SMARTFIRE model.

SMARTFIRE includes several additional features that are needed to model fire problems, compared to conventional CFD codes, e.g a six-flux radiation model [97], a radiation model (multi ray) [98], provision for handling the heat transfer through walls and a volumetric heat release model or gaseous combustion model (eddy-viscosity dissipation model) [99] to represent fires, smoke and turbulence (two equations $k - \varepsilon$ closure with buoyancy modification). All the mathematics to describe the different parts of the CFD code will be covered in Chapter 3.

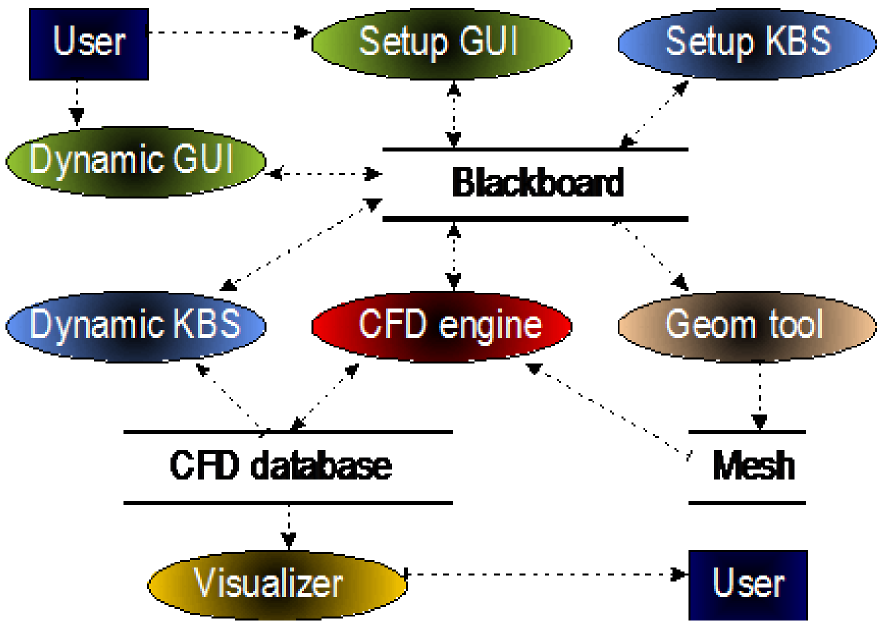


Figure 2.1: Sub-models used in SMARTFIRE.

A GUI represents the link between the user and the CFD engine, enabling him to specify the geometry/ problem by setting up

- Walls
- Material properties
- Compartments and obstacles e.g desks, stairs or partitions
- The fire (locations and properties)
- The radiation model type
- Gaseous properties (absorption coefficients)
- Location of vents (along with any fans, inlets or outlets)

After the problem has been set up, the GUI also enables the user to generate the control volumes (mesh) for the geometry, either manually or using the automated meshing tools. It should be mentioned that manually setting up the mesh requires a certain expertise to achieve the right solution, which is already build into the automated routines.

Finally the GUI handles the execution of the CFD calculations using the CFD engine. The problem domain will be numerically solved using a set of differential equations that describe the laws governing the physical processes involved. The solving of the actual variables uses a set of initial approximations whose accuracy is evaluated repeatedly. The solution has been obtained, as soon as the differences of consecutive accuracies fall under a pre-defined threshold (convergence). Convergence is only obtained if all variables fulfil the mentioned criteria.

SMARTFIRE's GUI offers a multi-window output layout so the user can track and alter outputs while the solution is obtained (using graphs, visuals and different data outputs). This thesis will not be using these features, reasons will be given in Chapter 4. However this is a feature not found in many CFD codes as making changes often requires the termination and restart of the CFD engine.

The work mentioned in this thesis will only focus on re-writing the CFD engine and recycle the other input and output features of SMARTFIRE as the engine constitutes the computational bottleneck of the simulation while the other parts are mostly idle.

Finally, SMARTFIRE has been specifically designed and developed as a Fire Simulation Tool to be used by Fire Safety Engineers and is based on more than 20 years of CFD fire modelling experience and research of FSEG [100].

2.2.5 EXODUS

Exodus is a set of software tools designed to simulate the evacuation of a large number of people (agents) from a variety of enclosure types by considering circulation/ pedestrian dynamics in complex environments. The set currently consists of three models airEXODUS [101] for aviation environments, maritimeEXODUS [102] for marine applications, and buidlingEXODUS [86] for applications in the built environment.

BuildingEXODUS [64] is used to model urban environments e.g. supermarkets, hospitals, cinemas, rail stations, airport terminals, high rise buildings, schools etc. One of the main purposes for simulating evacuation behaviour in buildings is to demonstrate compliance with building codes, evaluate the evacuation capabilities of buildings or spaces and to investigate crowd movement efficiency.

Various types of geometries can be created, including multi-levels and complex staircases, the software offers the ability to either draw the layout manually, or the import of different file types e.g DXF files from CAD software tools. Next, EXODUS sets up a two dimensional mesh of nodes over the floors (typically $0.5m^2$ per node) which represents the space normally occupied by an individual person. Individual nodes are different to the CFD mesh nodes and linked by arcs, which do not occupy any space.

EXODUS can simulate people-people, people-fire and people-structure interactions. Each agent in the simulation is treated as an individual and various data like exit times, speeds, etc. are recorded if the agents evacuation process is not prematurely terminated by exposure to fire (heat), smoke or toxic gases.

Internally, EXODUS consists of an object oriented, rule based engine written in $C{+}{+}$ which controls the simulations. Therefore the behaviour and routes of all agents are determined by a set of heuristics or rules. To introduce more flexibility into the modelling process these heuristics have been grouped into five sub-models which interact with each other, see Figure 2.2.

The sub-models are:

(i) Occupant

(ii) Movement

(iii) Behaviour

(iv) Toxicity

(v) Hazard

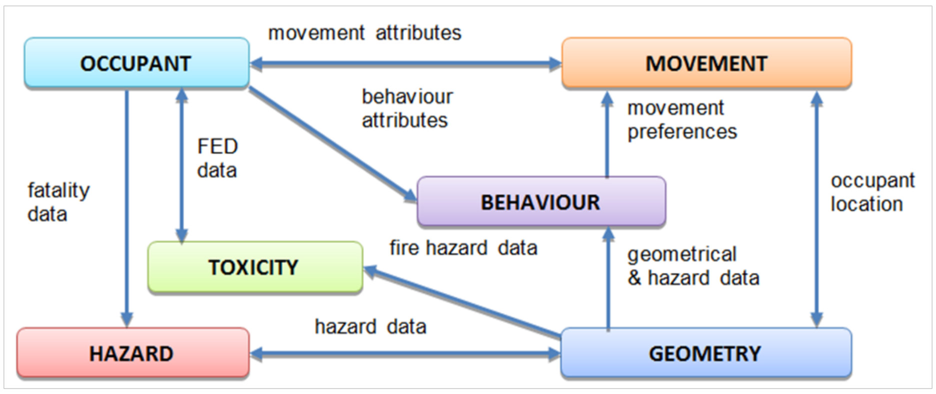


Figure 2.2: Sub-models used in buildingEXODUS [64].

Sub-models explained:

MOVEMENT: Controls the physical movement of each individual from their current position to their next position. It also controls waiting periods, overtaking, side-stepping and other evasive behaviours.

BEHAVIOUR: Controls how agents react to the currently present situation based on each agents personal attributes which then interacts with the movement model mentioned above. There are global and local behaviour levels, the local level controls how the agents react to their local environment whereas the global level holds the overall strategy for the population employed by each individual e.g. exit via the nearest exit or the most familiar one.

OCCUPANT: Defines the agents attributes lists, there are static attributes like age, gender, fast walking speed, walking speed, response time, agility, presence of life jacket, etc. and dynamic attributes that can also change during the simulation, triggered by other sub-models.

HAZARD: Controls the atmospheric and physical environment. The atmospheric aspects comprise the distribution of fire hazards CO2, CO, HCN, O2 depletion, Heat (radiative and conductive) and Smoke, as well as the irritant gases HCl, HBr, HF, SO2, NO2, CH2CHO (Acrolein) and HCHO (Formaldehyde). The physical aspects include setting of opening and closing times for LSAs. The hazard sub-model is basically responsible for the fire hazard and represents the interface to load-in SMARTFIRE data. The SMARTFIRE cell-by-cell based accurate results will be averaged over pre-defined zones which match in EXODUS and SMARTFIRE respectively. Generally two values are used for each hazard, one at 1.7m height and one at 0.5m height, which represents head height (standing height) and knee height (crawling height) of an average person [64]. Agents will automatically start crawling if the temperatures at head height exceed a critical value [86]. Agents will also be effected by smoke and their walking speeds will slow down [103][104].

TOXICITY: Determines the effects, including incapacitation, on an individual agent, exposed to heat, smoke and toxic products distributed by the hazard sub-model. The physiological effects on an individual exposed to the toxic and thermal environment are determined by using the Fractional Effective Dose (FED) concept [86][105].FED models assume that the effects of certain fire hazards are related to the dose received rather than the exposure concentration. The model calculates, for these fire hazards, the ratio of the dose received over time to the effective dose that causes incapacitation or death, and sums these ratios during the exposure. When the total reaches unity, the toxic effect is predicted to occur. Within buildingEXODUS, as the FED approaches unity the agents mobility, agility, and travel rates can be reduced making it more difficult for the affected agent to escape. Radiative heat also effects the agents through a pain threshold which is $80s(kW/m^2)^{4/3}$ and an radiative incapacitation threshold at $1000s(kW/m^2)^{4/3}$ which is considered more representative of the required dose for incapacitation of an average adult [64]. This means, buildingEXODUS takes the toxic and physical hazards associated with elevated temperature, thermal radiation, HCN, CO, CO2 and low O2, HCL, HBr, HF, SO2, NO2, Acrolein and Formaldehyde into account. The effects are passed back to the behaviour model which then effects the movement model.

2.3 Parallel Processing

Gill, in 1958, defined ’Parallel processing’ as ”Using multiple processors operating together on a single problem” [106]. The concept has already been recognised as an important technology by Flynn in 1996 [107] which led to his conclusion ”The continued drive for higher- and higher-performance systems thus leads us to one simple conclusion: the future is parallel.”

There are various different forms of parallelism within a computer system. One important requirement for a parallel operation is the independence of streams, where a stream is defined as a sequence of objects,data or instructions. In order to execute streams concurrently they have to be independent [107]. The three most common forms of parallelism are:

1. SIMD: Single Instruction Multi Data.
2. MISD: Multi Instruction Single Data.
3. MIMD: Multi instruction Multi Data.

SIMD architecture contains both array and vector processors which have been developed to operate on regular data structures like vectors and matrices. Array processors have multiple processor elements which process multi data elements, where vector processors have a single processor that sequentially operates on multi data elements. To run efficiently the array processor needs a massive set of data (therefore often called massive-parallel-processor), the vector processor on the other hands works good on smaller data sets by pipelining, see Appendix A.1, and increasing the clock rate to reduce latency. Over the years vector processors became increasingly popular and nowadays they are able to handle both SIMD and MIMD parallelism. MISD processors hardly exist, mainly due to restrictions in program designs and the mapping of applications onto the processors. Last MIMD forms the most basic form of parallel processor in contrasts to SIMD each processor works completely independently while they communicate through a shared memory space. With this layout two problems arise, the ordering of memory references across resources and maintaining cache coherence.

After having looked at the different types of hardware architecture it has to be determined whether a program is suitable for parallel execution. Most programs are initially designed to run in a sequential fashion, including loops and if statements that will cause the separation of the execution flow (branching). It is not a new concept to execute tasks in parallel, one of the first sophisticated systems was the SOLOMON machine 1962 [108][109] which was effectively a number of identical individual machines which could communicate through a supervisor unit. However the limitation was the dependency of the machine to the supervisor machine what meant they either executed the same task or were idle. The relationship of parallel processing and time sharing has been first pointed out by Gill [106]. Bernstein [110] showed that in order to decide whether a task is suitable for parallel execution it is also important to look at the algorithm performed as well as the way it is implemented. This means even if execution in parallel is possible the algorithm could still prevent effective parallel execution e.g. some recursive (sequential) operations can also be implemented in a non recursive fashion and executed in parallel.

As mentioned before in Section 2.1, Moores law of doubling the number of transistors per square inch per year does not fully apply any more and chip designers are facing problems like physical limitations (mostly power consumption and heat dissipation). Therefore they had to move away from Instruction Level Parallelism (ILP) to Thread Level Parallelism (TLP) where instead of being limited to physical space more and more execution units can be added whilst keeping clock rates low so power consumption and heat development stays low [111]. The Top500 list [112] shows as well that from year to year more cores are getting added to the world's top computer systems. The downside of increasing the number of cores is that the complexity of programming increases and programmers cannot rely on compilers to automatically take advantage of the new resources without major rewrites of the code.

The next step is to divide the problem up into individual parts (domains) that can be executed simultaneously. The most suitable and most frequently used approach is called domain decomposition, where a problem gets split into smaller problems (sub-domains), to iterate over them to coordinate the solution between adjacent sub-domains. An alternative approach is called functional decomposition, where the decomposition of the computation is done before dealing with the data. Both techniques are complementary and are often used in conjunction. A big advantage of decomposition methods is that the problems on the created sub-domains are almost independent, which makes them perfectly suitable for parallel processing. This thesis will focus on the domain decomposition strategy. A common approach for the coordination between the sub-domains is to use a Message Passing Interface (MPI), which handles the data exchange between the boundaries of the sub-domains. An MPI version of SMARTFIRE has been implemented by Grandison in 2003 [24]. MPI essentially spreads the problem over a number of machines available in a network, where communication, distribution of data and obtaining results is handled by sending messages.

This thesis will focus on a newer kind of parallelism, introduced in 2007 by Nvidia, which uses the hardware acceleration of the GPU to achieve SIMD parallelism. This technique will be detailed in Section 2.4.

2.3.1 Domain decomposition

Domain Decomposition Methods (DDM) solve boundary value problems by dividing a domain into smaller sub-domains, each a boundary value problem in itself. DDMs can be used with any discretization method in two ways, disjoint sub-domains or overlapping sub-domains [113]. A detailed description what was used in this thesis will be given in Section 4.1.

2.4 General-Purpose Graphics Processing Unit

GPUs were originally designed to do large amounts of repetitive work, i.e one task applied to a very large number of pixels. GPUs therefore have many more Arithmetic Logical Units (ALU) than Central Processing Units (CPU). As a result, they can do larger amounts of calculations than CPUs. CPUs on the other hand were designed to make decisions, directed by inputs from the software running. Therefore a large bulk of the CPU's structure is dealing with being able to switch to a different task on a moment's notice when needed. It also has to enforce privilege levels between user programs and the OS and handle memory management for executed programs.

The workload on GPUs used to do graphical operations on pixel, in fact is equivalent to many individual mathematical calculations to determine coordinates for screen outputs. This shows that the strength of the GPU hardware is to do a lot of calculations very efficiently which makes it very suitable for other computational intensive applications.

Exactly these advantages, combined with the pipeline concept described in Section A.1 allow the GPU to handle non-graphical operations very well.

2.4.1 History of the modern GPU

The era of modern computer graphics and therefore GPUs started in the early 1980s. IBM introduced the first video processing units in 1981. The so called *monochrome display adapter* was only able to handle monochrome colours, text only and always had to process groups of pixels at a time[1]. Even though these cards were quite limited in their abilities they still offered more capabilities compared to the present CPUs at the time. Following that, the next generation of hardware was called *All points addressable* and offered higher resolutions, greater colour depth and finally the ability to control individual pixels. But these cards were still limited as the main processing unit was still the CPU, which performed the whole workload and equated to increased performance at the expense of increased CPU workloads.

IBM changed the industry again in 1984, introducing the first processor based graphics card for PCs. The so called *PGA* [2] incorporated its own Intel 8088 processor which took over the video related tasks form the CPU. This was a very important step in the GPU evolution since it helped to further the paradigm of having separate processors performing the graphic relevant tasks. The lifespan of the card wasn't very long as it required 3 card slots on a very specific system, a special monitor and was expensive with around 4k$ [114] although it was targeted at businesses in the engineering and science field.

System requirements got larger and more graphic computational power was required when VisiOn introduced the first GUI in 1983 [115]. Even so the GUI made it more accessible,

[1]9x14 pixels

[2]Professional Graphics Adapter

the two main reason why GPUs got affordable over time, and why technology improved so much over the years is because of the so called the pipeline concept (see Appendix A.1) and the gaming industry, which demands high performance hardware at affordable prices. However as the hardware got more powerful other areas like engineering and science became interested in the technology. Figure 2.3 shows the development of GB/s over the years for CPUs and GPUs, it can be seen that since 2003, the GPUs clearly outperform the CPUs.

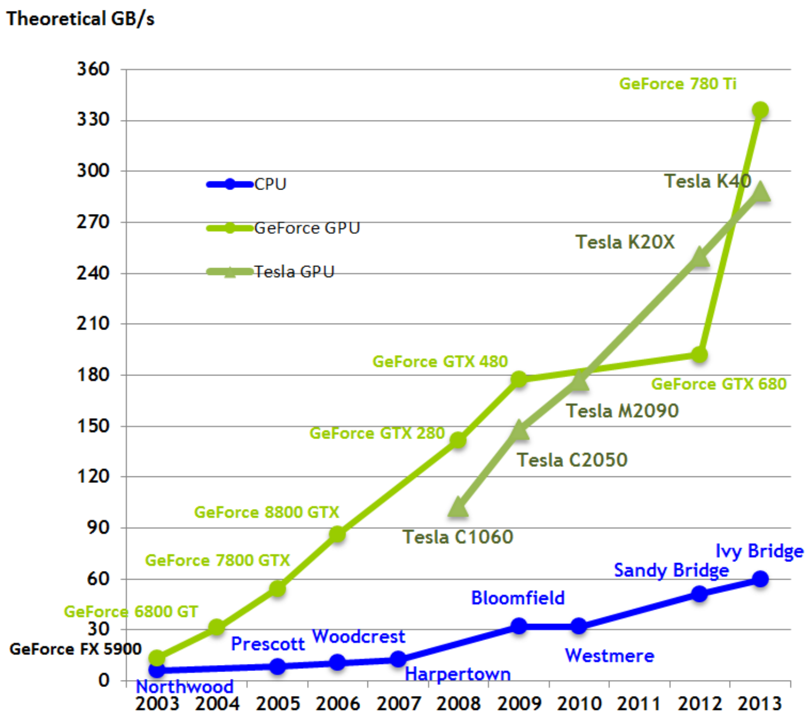


Figure 2.3: Increase in GB/s for CPU and GPU [116].

Over time hardware changed from more specialised logic units dedicated to do one kind of job e.g vertex processing or pixel processing to more general units that can be used for all kinds of jobs. Precision has also increased by moving away from indexed arithmetic to integer and fixed point, then to floating point and most recent to double precision operations and data storage concepts. In the next step general purpose programming languages like C and C++ have been introduced [117; 118] to lower the entry barrier for programmers. There is clearly a trend that GPUs will cover more and more tasks in the near future and this trend will continue [119; 120]

Around 2003 big companies, most of all Nvidia, aggressively pushed GPUs towards being general purpose parallel computing devices[121], which they for the first time called General Purpose GPU (GPGPU) [122–124]. People used graphics Application Program Interfaces (API), which were not originally intended to do scientific computing, to implement all sorts of applications to the GPU, amongst them: pharmaceutical, financial, physics and medical applications. The current development indicates that GPUs are becoming much more capable processors and in difference to CPUs, still increase their hardware potential dramatically with every generation [125].

With Nvidia's introduction of the unified shader model 2008 (used in their 8800 chips), vertexes and shaders from there on relied on the same instruction set and therefore the same hardware. These chips consist out of eight Thread Processing Clusters (TPC), each contain two Streaming Multiprocessors (SM). Each SM furthermore eight Scalar Processors (SP) each with it's own ALUs and FPUs which can run multiple threads. The main idea here was to distribute the workload equally to all available hardware resources on the card[126].

Furthermore, GPUs have become much more complex than CPUs, if you look at the transistor count GPUs have more than doubled the amount of a general purpose CPU. By looking at Moore's Law, which states that transistor count doubles every 12 months[1] [48]. In the last decade, GPU manufacturers have far exceeded Moore's Law by doubling the transistor count every 6 month [127].

In the past, there has been a point where low volume specialized CPUs, like the ones found in high end vector supercomputers, had become too expensive to manufacture. Therefore the market had to swing to commodity processors and clusters. Which brought great reductions in price-to-performance, but then scaling problems accrued and the size of clusters became a problem in terms of power, space, and even the speed of light. One solution would be to continue to add cores to commodity processors, but basically this means adding more general purpose scalar nodes. The GPU's advantage here is that it has been specifically optimised for these kind of tasks (SIMD Single Instruction Multiple Data). The Nvidia Fermi

[1]Nowadays its more every 18 months.

architecture's HPC features, bring vector processors back into supercomputing by enhancing the commodity scalar core with a commodity vector processor [128].

Performance Statistics

Despite having a great amount of processing power manufacturers are working hard on improving energy efficiency along with increased processing power which is stated as performance per watt. Figure 2.4 and 2.5 show the $flops$ rates CPU and GPU implementations produced. It can clearly seen that the GPU outperforms the CPU by far, where the speed-up was depending on the case they ran. It can also be seen in Figure 2.4 that GPUs only fully utilise their performance if the workload is high enough.

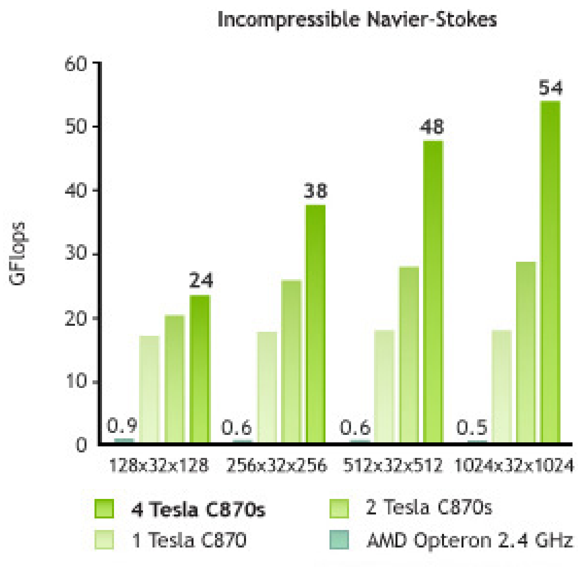


Figure 2.4: GPU vs CPU Navier Stokes benchmark [129]

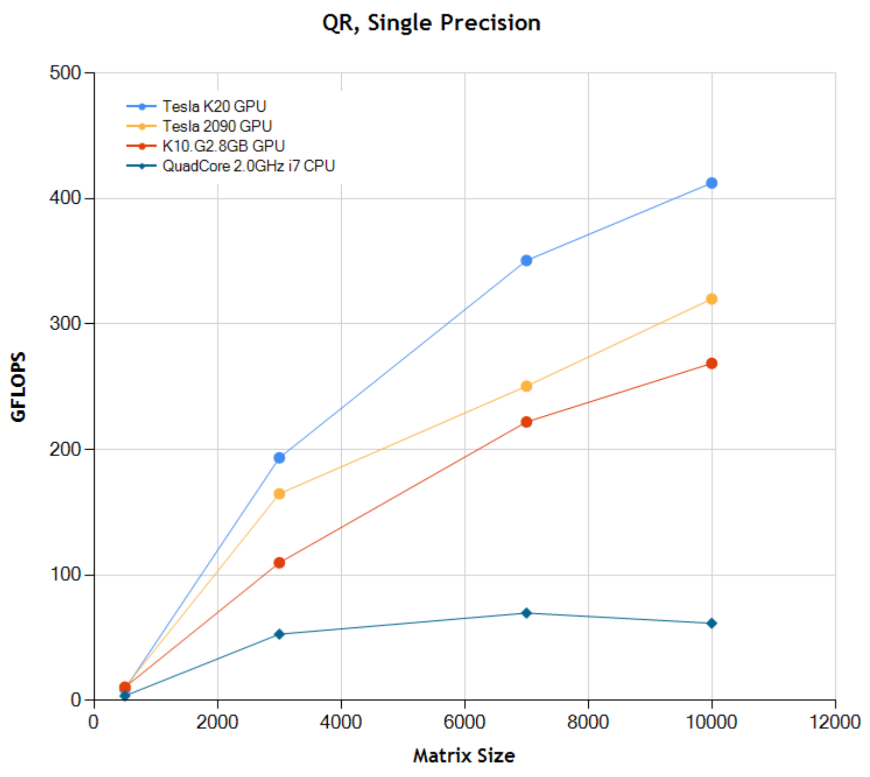


Figure 2.5: GPU performance vs CPU performance [130]

2.4.2 GPU APIs

To utilise the processing power of GPUs and to reduce the amount of low-level hardware programming, a number of APIs have been developed. The three main APIs are Nvidia's CUDA [117], an OpenSource API called OpenCL [131] and OpenACC. Each of the APIs has its advantages and disadvantages which will be discussed in the following sections.

2.4.2.1 Open Computing Language (OpenCL)

Open Computing Language(OpenCL) is a framework that targets a multi heterogeneous platform approach which includes CPUs, GPUs, Digital Signal Processors (DSP) and other platforms like Field Programmable Gate Arrays (FPGA) and Intel's Phi[132]. OpenCL consists of two languages, both based on the C99 standard, one for writing parallel functions and the other parts that handle the platform controls. OpenCL provides the developer with the ability to write parallel computing using task-based and data-based parallelism which will natively run on different platforms. It has been adopted by Apple, Intel, Qualcomm, Advanced Micro Devices (AMD), Nvidia, Altera, Samsung, Vivante and ARM Holdings. However it has been shown [133] that code portability does not always guarantee performance portability and that the real effort still lies in tuning and often rewriting function for different platforms. Therefore the future advantage of OpenCL will mainly lie in the reduction of the development time and therefore the time to market for platforms that are difficult to program in a traditional way e.g FPGA with VHDL[1] or Verilog[134].

2.4.2.2 OpenACC

Open Accelerators (OpenACC) is an accelerator programming standard that enables Fortran, C and C++ programmers to take advantage of heterogeneous CPU/accelerator systems (including GPUs).Nvidia, CRAY and CAPS recently (May 2014) introduced an OpenACC version (PGI 14.4) for GPU programming that supports C++. In contrast to a full programming model OpenACC uses compiler re-directives, similar to OpenMP [26], which try to offload computational heavy parts of the code, e.g. loops to an attached hardware device like a GPU. By offloading these loops OpenACC takes care of the data movement between CPU and device. The main advantage here is that computationally intensive parts of the codes can be quickly identified and parallelized with much less coding effort. The additional code directives only get executed by compilers supporting the OpenACC API and get ignored otherwise. Hoshino published a paper in May 2013 [135] in which he evaluates the performances of a Compute Unified Device Architecture(CUDA) CFD code compared to an OpenACC implementation, he came to the conclusion that on average OpenACC performed 50% slower than the CUDA implementation but that for other types of applications, by using more manual optimisation, results closer to 98% of the CUDA performance can be achieved. The main disadvantage is the lack of full hardware support , e.g no shared memory use available. The CFD code used was UPACS, a Fortran code project that offers multiple different solvers on structured meshes. By the time this research was conducted there was no GPU capable OpenACC version and the shared memory concept is one of the key concepts for the solver implementations discussed in Section 4.3.1.

[1]VHSIC Hardware Description Language

2.4.2.3 CUDA

This work exclusively focuses on the uses of NVIDIA's CUDA parallel computing architecture [117], but the concepts described in this thesis can also be implemented using OpenCL[118]. This is mainly due to the fact that by the time when this work was conducted OpenCL was fairly new and had several disadvantages compared to CUDA, e.g no template support, no 3rd party libraries were available e.g thrust [136] or CULA[137] and the development environment was far less practical than CUDA. The CUDA debugging tool, NSight [138], was far superior to the tools available for OpenCL. The biggest disadvantage of CUDA is that it exclusively runs on NVIDIA hardware.

One of NVIDIA's purposes to introduce CUDA was to lower the entry barrier for programmers to cope with the complexity of low level programming. CUDA offers essential functionality for parallel applications, such as scattered read and write accesses in memory and atomic operations which were not easily possible in previous APIs. A full list of operations is available in NVIDIAs programming guide [117]. CUDA C is essentially a C extension which adds extra constructs to deal with the hardware. CUDA provides a computing model and an Instruction Set Architecture (ISA)[117]. It is important that the hardware and the implementations are scalable due to the variety of hardware and software available which was traditionally very difficult to achieve in traditional parallel programming.

CUDA Programming Model

Before the CUDA hardware model will be introduced, the focus is on the programming model. One important point here is that Nvidia guarantees the compatibility to earlier generations in the future. That means code developed now will run without or only minor changes on future hardware generations even if they are a lot more complex and technically advanced.

The program structure always requires a serial part of the program that calls parallel routines, the kernels. Kernels can be simple or very complex functions. Kernels execute as parallel processes by launching a set of parallel elements, called threads [1]. Threads are grouped into a hierarchy of thread blocks which form a grid, as in Figure 2.6. To make problem abstraction easier blocks can be one, two or three dimensional and grids can also be one, two or three dimensional.

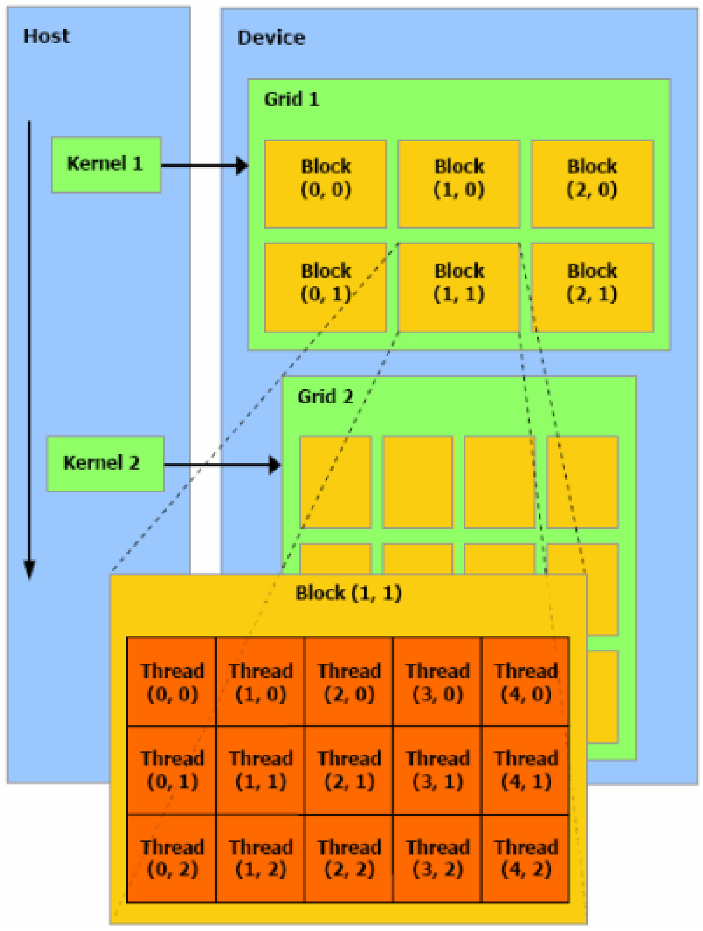


Figure 2.6: Fermi Grid - Block - Thread layout

The actual execution configuration, number of blocks and number of threads per block have to be set by the programmer and are a crucial factor how efficient the code will run. Nvidia provides a number of optimisation and profiling tools which provide support for the programmer i.e Nvidia Visual Profiler, nvprof and the CUDA Occupancy Calculator spreadsheet. By executing a kernel, each thread is given an unique identifier, the thread id and

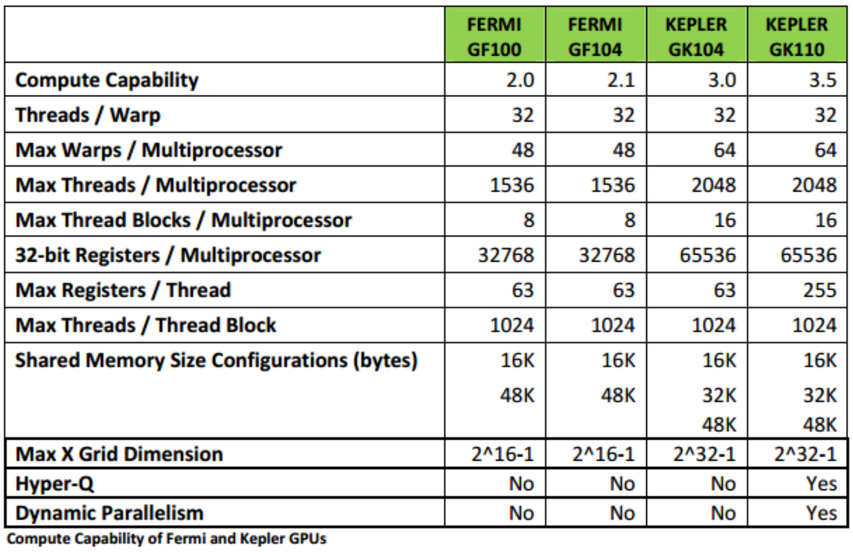
[1]Thread here is not the same as in the classic CPU multi-thread concept.

each block is given a unique block id based on the chosen configuration. This is important as it ensures that all threads are accessible when needed. The CUDA commands threadIdx, blockIdx and blockDim in Listing 2.1 are used to calculate exactly this identifier. This principle corresponds to the row-major order and column-major order methods for storing multidimensional arrays in linear memory in conventional coding [139]. At present Nvidia's latest hardware generations called Fermi [121] and Kepler [140] support block sizes of up to 1024 threads per block see Figure 2.7.

Listing 2.1: Vector addition example in CUDA

```
__global__ void vec_add(int *vec_a, int *vec_b, int *vec_c,
                                                 int *length)
{
int identifier = blockIdx.x * blockDim.x + threadIdx.x;
if(identifier < length)
{
vec_c[identifier] = vec_a[identifier] + vec_b[identifier];
}
}
```

Figure 2.7 shows the CUDA version of a vector addition operation, where vec_c = vec_a + vec_b. The three vectors have 'length' number of elements. It can be seen that 'identifier' accesses the same element of each vector per operation.



	FERMI GF100	FERMI GF104	KEPLER GK104	KEPLER GK110
Compute Capability	2.0	2.1	3.0	3.5
Threads / Warp	32	32	32	32
Max Warps / Multiprocessor	48	48	64	64
Max Threads / Multiprocessor	1536	1536	2048	2048
Max Thread Blocks / Multiprocessor	8	8	16	16
32-bit Registers / Multiprocessor	32768	32768	65536	65536
Max Registers / Thread	63	63	63	255
Max Threads / Thread Block	1024	1024	1024	1024
Shared Memory Size Configurations (bytes)	16K 48K	16K 48K	16K 32K 48K	16K 32K 48K
Max X Grid Dimension	2^16-1	2^16-1	2^32-1	2^32-1
Hyper-Q	No	No	No	Yes
Dynamic Parallelism	No	No	No	Yes

Compute Capability of Fermi and Kepler GPUs

Figure 2.7: Compute Capability of Fermi and Kepler GPUs [140]

The next paragraph will demonstrate the basic use of the C extensions mentioned in Section 2.4.2.3. Listing 2.1 shows a simple vector addition of vec_a and vec_b into vec_c. Functions executed on the GPU have to be defined in files with the file ending .cu or .cuh to let the compiler know how to handle the code. Inside these files, functions must start with either the identifier $\_\_global\_\_$ or $\_\_device\_\_$. Global functions are kernels which are triggered by host calls. Device functions are functions which can be called from within a kernel call. CUDA offers the option to launch kernels from within a kernel since kepler was introduced [140]. This can be useful if the grid-block layout has to be changed during runtime. Line 4 in Listing 2.1 dimGrid and blockId are vectors of type dim3 which specify the grid and block-dimensions. If one dimension is not specified it will automatically be set to 1.

The real difference between the serial (Listing 2.2) and CUDA versions of the code (Listing 2.1) is that the CUDA identifier replaces the for-loop of the C code example. A serial execution has therefore been exchanged by 'length' number of parallel executions, which all calculate their own entry of vec_c. It can also be seen here that the operation is perfectly parallel none of the threads depends on the results of another thread.

Listing 2.2: Vector addition example in C

```c
int vec_add(int vec_a, int vec_b, int vec_c, int length)
{
        for(int i=0; i<length: i++)
        {
                vec_c[i] = vec_a[i] + vec_b[i];
        }
        return vec_c;
}
```

The programmer does not have to handle the parallel execution and thread management manually. All thread creations, scheduling, and terminations are handled by the underlying CUDA API. Nvidia's top range tesla GPUs have specifically been designed to do scientific computing and are more sophisticated than normal consumer GPUs and can handle all thread management directly in hardware [141]. However the programmer still has the ability to interfere with the API's thread management system by using synchronisation commands i.e *__syncthreads()* [117], these commands are barriers and used to guarantee task completion up until this barrier before continuing. This method is often used to ensure that memory loads are finished before memory accesses take place. Using barrier functionality on block level is not well suited for the CUDA parallel model as it goes against the concept of task parallelism among threads within a block. This is mainly because this would limit the number of processes that a kernel could run on.

All memory used on the GPU has to be allocated/ freed see Listing 2.3 and populated see Listing 2.4 from within the serial host code of the program.

Listing 2.3: Host command to allocate and free a CUDA variable

```
cudaMalloc((void **) &var_device,length));
cudaFree(var_device);
```

Listing 2.4: Host command to populate a CUDA variable

```
cudaMemcpy(var_device,var_host,length,↩
   cudaMemcpyHostToDevice);
cudaMemcpy(var_host,var_device,length,↩
   cudaMemcpyDeviceToHost);
```

Listing 2.3 allocates the variable "var_device" with the dimension "length" on the GPU. Followed by Listing 2.4 which shows two memory copies host-to-GPU and GPU-to-host. These memory copies are necessary as CUDA can only handle calls by reference where only pointers will be passed to data that cudaMemcpy has copied somewhere into the GPU's memory before. As soon as the host copies data onto the device, the data on the device becomes invisible to the host and vice versa the only connection the two still have are the pointer addresses where the data is stored on the host and on the device. In exceptional circumstances where it cannot be avoided one can create pinned memory which is both visible and accessible to the host and the device but its performance is very poor compared to the non pinned version and should therefore always be avoided. Finally the kernel can now be called as shown in Listing 2.5.

Listing 2.5: Host command to launch a CUDA kernel

```
kernel<<<dimGrid, dimBlock>>>(... parameter list ...);
```

Due to this separation of hardware and the use of different compilers debugging applications can be very tricky as executing the code, often does not cause the program to terminate even if the actual operation went wrong already, e.g division by zero will lead to "#INF" but won't crash the program.

Another thing to avoid are recursive function calls as providing stack space for thousands to millions of threads that are active would require substantial amounts of memory [126]. Typical recursion operations like quicksort [142], are normally best implemented using multiple kernel calls. The first kernel computes the results per block and the second kernel calculates the results over the blocks.

It is also worth mentioning as well that GPU application will only pay off if they have a certain problem size to be executed on, otherwise the overhead created by the memory transfers will outnumber the serial CPU execution time by far.
The main abstractions of the programming model are namely, hierarchy of blocks and threads, the shared memory model (will be explained in the next section) and barrier synchronisations. These abstractions allow the programmer to control parallelism at a very detailed level. However programmers have to start thinking ”in parallel” right from the beginning of designing a piece of code. They have to break their problems down into domains and domains into sub-elements respectively which then can be executed independently.

The concept of partitioning will be discussed in more detailed in Section 4.1. Threads are lightweight entities whose context can be switched with zero overhead [143]. The zero overhead is achieved by a concept which Nvidia calls ”the GigaThread engine”, which allows thousands of threads to be active during execution [144].

CUDA Hardware Model

After having looked at the programming model this section will describe how CUDA processes data on the hardware level. A GPU contains various kinds of different memories, registers and processors. The choice of appropriate memory is one of the main challenges CUDA programmers face, as the faster the type of memory, the less of it is available. Different kinds of memories suit different kinds of problems better than others. The different types of memories are global, constant, texture, local, shared memory and registers. The memory types that have been used throughout this thesis are global memory and shared memory. The other types of memory have their perfect use in image and video processing and were therefore not important. The biggest type of memory available is global memory which is a few GB but with a very long latency, in contrast to shared memory which only has a few KB per block but is 10 x faster to access. CFD problems typically need hundreds to thousands of MB of memory therefore combinations of the faster and limited memories and the larger and slower ones have to be used. But this does not hold a problem as threads can access different kinds of memories during their runtime. The most common combination is threads using global and shared memory.

Global memory is visible to all threads, shared memory is only visible to the threads within one block and has the lifetime of the kernel call (see Figure 2.8). Shared memory variables in contrast to the global variables mentioned in the previous section, must be defined inside the kernel definition and start with __shared__ [117]. On a tesla GPUs, these memories are physically separated, shared memory is a low-latency on-chip RAM, while global memory resides in the fast DRAM on the graphics board. As shared memory is so much faster it has to be close to the processor quite like the L1 cache on CPUs. Shared memory is only appropriate to use if the trade-off between the additional memory reads and writes pay off. In this thesis shared memory was used to store the values of the currently solved variable ϕ and were therefore accessed 10 x-100 x per sweep. All other values had to be kept in global memory due to their size.

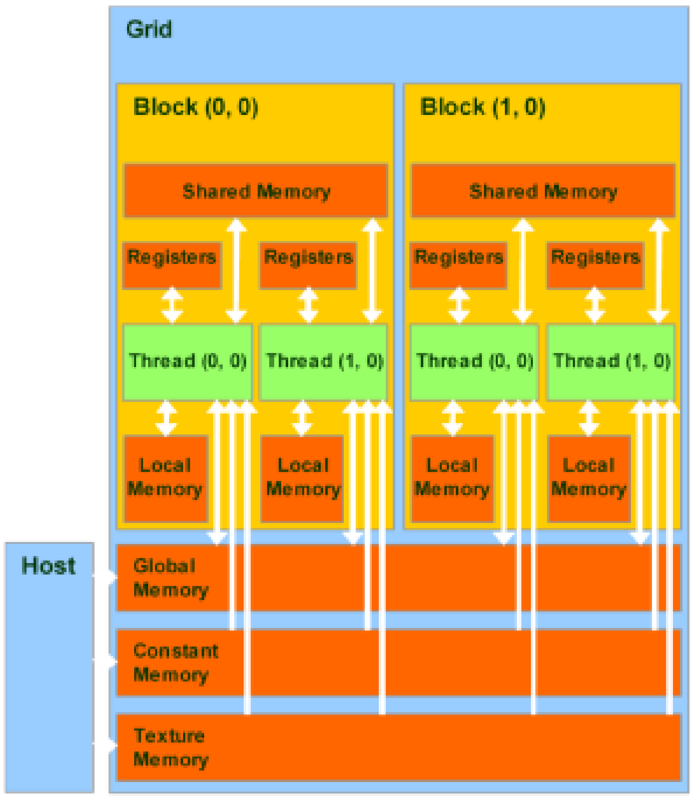


Figure 2.8: Fermi memory model

Looking at Figure 2.8 it can also be seen that there is a physical divide between the blocks, which means that every information exchange between blocks has to be done through the slow global memory what will be discussed in detail in Section 4.2. However since threads in a block may share memory and synchronize via barriers, they will reside on the same physical processor or multiprocessor. In a typical CUDA program the number of thread blocks will greatly exceed the number of processors available. As soon as hardware resources are exceeded warps will queue up and the scheduler will pick them up as soon as resources are available again. Warps are threads executed in groups of 32. On a tesla GPU, the 8 cores in a group are quad-pumped to execute one instruction for an entire warp, 32 threads, in four clock cycles.

Programmers do not have to deal with hardware resources or processing orders manually as this is all handled by the internal schedulers and processors. This has the advantage that changing the number of cores or resources in the future will not require programs to be re-designed and helps to avoid possible deadlocks. The only requirement to the programmer is that therefore blocks must be completely independent from each other so that the scheduler can pick them up for execution at any time and in any order. Nvidia has heavily invested in implementing more functionality to move away from that strict divide e.g atomic memory operations on global memory visible to all threads by atomically incrementing queue pointers. The last and most restricted type of memory to mention is local memory which is only accessible by each individual thread and normally used to store constants.

CUDA's biggest limitation is definitely the separation of CPU memory and GPU memory (which will be solved by CUDA V6.0). This process represents the biggest bottleneck in every CUDA code and will drastically reduce the efficiency. Therefore one of the main challenges a developer has to face is to reduce memory transfers between the host PC and GPU to a minimum. One approach is to create data structures on the device that can be re-used over and over again during runtime, limiting memory transfers to a bare minimum.[1].

Threads are executed in an asynchronous[2] fashion. Each SM can handle multiple thread blocks simultaneously as long as the hardware limits are not exceeded. Execution limitations can be seen in Figure 2.7.

Each streaming multiprocessor has a scheduler which manages when each of the warps will be executed by their SIMD cores. If any thread within the warp takes a different branch while executing, a situation occurs which is called branch divergence[3]. Branch divergence can have a big effect on execution speeds if many of the warps are effected. Section 4.3.1 will provide techniques to avoid and improve such effects.

To hide latency SMs can automatically switch context to other warps while memory transaction are still performed. To hide as much latency as possible a good approach is always to have a high occupancy and therefore it is better to use more blocks than less. The same applies to memory accesses as a warp can read/ write up to 32*4 bytes = 128 byte of data

[1]2 memory transfers will be the minimum as data has to be copied onto and off the device

[2]Threads can complete operations before others are even started

[3]Serialisation of all 32 threads within the warp

in one instruction for coalesced memory. For example if 10 threads want to access memory locations which are not stored consecutively in memory, this will be performed in 10 individual memory transaction compared to one. Programming issues disregarding these restriction are mainly the reason for code inefficiency [145].

Chapter 3

MATHEMATICAL MODELLING FOR CFD - FIRE SAFETY ENGINEERING

In this chapter the basic mathematical methodologies used in most CFD codes and in SMARTFIRE in particular will be discussed. The general conservation equation will be described and integrated to illustrate the general method of discretisation. The discretisation of the solved physical equations along with details of the handling of various boundary conditions and special treatment of the source terms will be discussed. The solved equations are the momentum (Equation 3.1.2), enthalpy (Equation 3.1.3), and mass continuity (Equation 3.1.4) Differential Equation (DE). Fire models require additional modelling for turbulence and radiation respectively which will also be described. The basic solution algorithm used in SMARTFIRE and other fluid dynamic codes called SIMPLE will be detailed.

The Rhie and Chow (or Pressure Weighted) interpolation method will be described including a discussion on the relative merits and demerits compared to an alternative approach that utilises a staggered formulation. Finally, the numerical solver will be discussed.

3.1 Mathematical equations for fire CFD modelling

3.1.1 The General Conservation Equation

The general conservation equation takes the following form:

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla.(\rho\underline{u}\phi) = \nabla.(\Gamma_\phi \nabla\phi) + S_\phi \tag{3.1.1}$$
$$Transient + Convection = Diffusion + Source$$

where ϕ represents the dependent variable to be solved, ρ is the density of the fluid and u is the velocity of the fluid. Γ_ϕ is the diffusion coefficient for the variable ϕ , which may represent such quantities as viscosity or conductivity. S_ϕ is the additional source terms for the variable ϕ.

3.1.2 The Momentum Equation

The conservation of momentum is shown in Equation (3.1.2).

$$\frac{\partial(\rho u_i)}{\partial t} + \nabla(\rho u_i \underline{u}) = \mu div(grad(u_i)) - \frac{\partial P}{\partial x_i} + S_i \tag{3.1.2}$$

In terms of the general conservation Equation (3.1.1) $\phi = u_i$. The source term S_i absorbs all the other terms, although the pressure gradient term is treated separately and is described in detail in Section (3.4.4).

3.1.3 The Continuity Equation

The mass continuity equation takes the form as shown in Equation (3.1.3):

$$\frac{\partial\rho}{\partial t} + \nabla(\rho\underline{u}) = \dot{m} \tag{3.1.3}$$

where $\dot{m}$ is a source of mass. In terms of the general conservation equation $\phi = 1$ and $\Gamma = 0$.

3.1.4 The Enthalpy Equation

It has the same form as the general differential equation and is detailed below:

$$\frac{\partial(\rho h)}{\partial t} + \nabla.(\rho \underline{u} h) = \nabla.\left(k\nabla\left(\frac{h}{c_P}\right)\right) + S_h \tag{3.1.4}$$

To calculate the spread of heat throughout the domain it is necessary to solve the enthalpy equation, (see Equation 3.1.4). Where k is the thermal conductivity, c_P the specific heat and S_h represents any source of enthalpy.

3.1.5 Turbulence Model

3.1.5.1 Time Averaged Approach

In previous sections, the conservation equations for momentum continuity and the general conservation equations have been shown. These equations describe an instantaneous value for the flow quantities but scientists are generally more interested in the time averaged quantity than the actual values due to random fluctuations. The instantaneous equations are therefore time averaged. The time dependent quantities can be considered to consist of a time averaged component $\widetilde{\phi}$ and a random fluctuating component ϕ".

The mean value $\bar{\phi}$ of a variable ϕ can be defined as

$$\bar{\phi} = \frac{1}{\Delta t}\sum_{t-1/2\Delta t}^{t-1/2\Delta t} \phi dt \tag{3.1.5}$$

The Favre (density weighted) average for compressible flows is defined as

$$\tilde{\phi} = \frac{\overline{\rho\phi}}{\bar{\rho}} \tag{3.1.6}$$

and the instantaneous value of ϕ can be described by

$$\phi = \tilde{\phi} + \phi" \tag{3.1.7}$$

Applying Favre averaging to the conservation equations leads to the following, with the ˜ dropped for clarity and superposed bars to indicate the Favre averaging.

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho \underline{u}) = 0 \tag{3.1.8}$$

$$\frac{\partial(\rho u_i)}{\partial t} + \nabla(\rho u_i \underline{u}) = \mu div(grad(u_i)) - \frac{\partial P}{\partial x_i} + S_i - \frac{\partial}{\partial x_j}\left(\rho \overline{u_i^" u_j^"}\right) \tag{3.1.9}$$

$$\frac{\partial(\rho h)}{\partial t} + \nabla.(\rho \underline{u} h) = \nabla.\left(k\nabla\left(\frac{h}{c_P}\right)\right) + S_h - \nabla.\left(\rho \overline{\underline{u}^" h^"}\right) \tag{3.1.10}$$

The effect of Favre averaging on the general conservation Equation 3.1.1 is similar to the enthalpy equation.

$$\frac{\partial(\rho \phi)}{\partial t} + \nabla.(\rho \underline{u} \phi) = \nabla.(\Gamma_\phi \nabla \phi) + S_h - \nabla.\left(\rho \overline{\underline{u}^" \phi^"}\right) \tag{3.1.11}$$

After the averaging the continuity equation remains unchanged. The momentum Equation 3.1.9 and the general conservation Equation 3.1.11 have both gained an additional term on the right hand side. The term $\rho\overline{(u_k^" u_l^")}$ is known as the the Reynolds (turbulent) stress. The term $\rho\overline{\underline{u}^" \phi^"}$ is known as the Reynolds (turbulent) flux. These terms need a turbulence model to be calculated. SMARTFIRE uses the standard buoyancy modified $k - \varepsilon$ turbulence model which will be described in detail in the next paragraph. The various methods of modelling turbulence from a fire safety engineering (FSE) point of view have been reviewed by Kumar [146].

The $k - \varepsilon$ turbulence model is based on the viscosity hypothesis [32], and two additional variables k (the turbulent kinetic energy) and ε (the dissipation rate) and leads to the following equations

$$\frac{\partial(\rho k)}{\partial t} + \nabla.(\rho \underline{u} k) = \nabla\left(\left[\mu_{lam} + \frac{\mu_t}{\sigma_\varepsilon}\right]\nabla k\right) + P + G - \rho\varepsilon \tag{3.1.12}$$

$$\frac{\partial(\rho \varepsilon)}{\partial t} + \nabla.(\rho \underline{u} \varepsilon) = \nabla.\left(\left[\mu_{lam} + \frac{\mu_t}{\sigma_\varepsilon}\right]\nabla \varepsilon\right) + \frac{\varepsilon}{k}\left[C_{l\varepsilon}(P+G)(1+C_3) - C_{2\varepsilon}\rho\varepsilon\right] \tag{3.1.13}$$

where (Equation 3.1.12) is the turbulent kinetic energy equation and (3.1.13) is the dissipation rate equation. In these equations, P represents the turbulent production rate:

$$P = 2\mu_t \left\{ \left(\left[\frac{\partial u}{\partial x}\right]^2 + \left[\frac{\partial v}{\partial y}\right]^2 + \left[\frac{\partial w}{\partial z}\right]^2 \right) + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right)^2 \right\} \tag{3.1.14}$$

and G represents the buoyancy term:

$$G = -g\frac{\mu_t}{\rho}\frac{\partial \rho}{\partial y} \tag{3.1.15}$$

The eddy viscosity hypothesis assumes that the turbulent stresses are proportional to the mean velocity gradients in a similar manner to viscous stresses in laminar flows. This is expressed as,

$$\rho\overline{u_i^" u_j^"} = \mu_t\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\rho k\delta_{ij} \tag{3.1.16}$$

Similar, in the general conservation equation the turbulent flux is assumed proportional to the gradient of the variable ϕ.

$$\rho\overline{\underline{u^"}\phi^"} = \frac{\mu_T}{\sigma_T}\left(\frac{\partial \phi}{\partial x}\right) \tag{3.1.17}$$

The unknown fluctuating quantities have been replaced with known time averaged values. The last term in Equation (3.1.16) is effectively a pressure term and becomes absorbed into the pressure gradient term of the momentum Equation. The value of turbulent viscosity, μ_t,is calculated from Equation (3.1.18).

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon} \tag{3.1.18}$$

The constants used with the $k - \varepsilon$ model are detailed in Table 3.1 below.

Table 3.1: Constants used in $k - \varepsilon$ model

C_μ	σ_k	σ_ε	C_{l_ε}	C_{2_ε}	C_3
0.09	1.0	1.3	1.44	1.92	1.0

3.1.5.2 Direct Calculation

Time averaging only applies if the scientist is not prepared to run the simulation using the finest time and length scales to resolve all turbulent eddies. This is known as Direct Numerical Simulation (DNS) and is the most realistic approach to modelling turbulence as a turbulence model is no longer needed. The downside is that very fine meshes and time steps are needed to resolve all turbulent eddies. The consequences are that even simple models are expensive to run as they take a long time. An intermediate method, the large eddy simulation (LES) [147] tries to maintain the accuracy of DNS without the required mesh sizes. Details of the LES will be discussed in the following Section (3.1.5.3).

3.1.5.3 Large Eddy Simulation Model

LES models are used to model turbulence in computational fluid dynamics but were not part of this research. To give a full overview the LES background can be found in Section (A.3) of the Appendix.

3.1.6 Radiation Model

Radiation is the method of heat transfer without any physical contact between sources and objects, in contrast to conduction and convection in Section (3.4.2). The process of radiation neither requires exchange of mass nor a medium. Radiation contributes a significant proportion of heat transfer in FSE, and therefore cannot be neglected. The following chapter will briefly discuss the two radiation models used by SMARTFIRE, a detailed description is given in [148]. Equation (3.1.19) describes the radiative transfer, while a grey system is assumed, i.e there is no wavelength dependence.

$$\frac{\partial}{\partial l}I(\Omega,r) = -(\alpha+s)I(\Omega,r) + \alpha I_b(r) + \frac{s}{4\pi}\sum_{\Omega'=4\pi} I(\Omega',r)\Phi(\Omega' \rightarrow \Omega)d\Omega' \qquad (3.1.19)$$

where Ω is the ray direction, l represents the physical path length along Ω at position r,α is the absorption coefficient and s is the scattering coefficient of the medium, $I_b(r)$ is the black body radiation intensity and $\Phi(\Omega' \rightarrow \Phi)$ is the scattering phase function.
Solving Equation (3.1.19) is an extremely complex task therefore a number of assumption

have been made and various simplified radiation models have been developed [149]. Which are zone method [150], Monte Carlo Methods [151], composite flux models (six-flux) [152–154] and discrete transfer models [155]. Both Monte Carlo and zone method are computational very expensive to run unless very coarse meshes are used. As fire modelling usually demands a fine mesh these two approaches are not practical.

3.2 Numerical Procedure

The process towards making models suitable for numerical evaluation is called discretisation which transforms a set of simultaneous continuous non-linear partial differential equations (PDEs) of ϕ into simultaneous linear algebraic equations which give the values of the variable ϕ at a number of discrete points in time and space.

Discretisation converts PDEs into a set of algebraic equations that are commonly expressed in matrix and vector form as $\underline{\underline{A}}\underline{\phi} = \underline{B}$. This matrix can now be solved, for simple cases by finding the inverted matrix $\underline{\underline{A}}^{-1}$ or by using iterative methods such as the Jacobi Over Relaxation (JOR) methods which will be described later in Section 3.9.1.

3.3 Discretisation Scheme

There are a number of different discretisation methods available in CFD including Control-Volume (CV)/ Finite-Volume (FV), Finite-Element (FE), Finite-Difference (FD) and the Boundary Element (BE) method. This section will focus on the finite-volume method.

The FV approach is based on Gauss' divergence theorem:

$$\iiint\limits_V \nabla \underline{F} dV = \oiint\limits_S \underline{F}.\underline{\hat{n}} dS \tag{3.3.1}$$

where V is some arbitrary volume in space, $\underline{\text{F}}$ is a vector field extending throughout the volume, S is the surface that encases the volume V and $\underline{\hat{\text{n}}}$ is the unit normal vector over the surface S. By looking at Equation (3.3.1) it is clear that there is a relationship between a volume integral and a surface integral, which is the fundamental part of the FV method.

Therefore the large computational domain can be divided into smaller sub-domains, finite volumes (control volumes) around which the surface integral can be used. The main advantage of the finite volume method is that the variable of interest is conserved for the corresponding control volume. This means the variable is conserved no matter how coarse or fine the mesh is, locally and globally.

3.3.1 The Computational Grid

The smaller control volumes discussed in Section (3.3) will now be referred to as cells. The cells are chosen to fulfil two criteria, they have to completely fill the computational domain and have to ensure the cells do not overlap. The cells are considered to surround a point in the computational domain, the variables that will be solved will have their solution in these cell-centres. The boundaries of these cells form a grid, also known as a mesh.

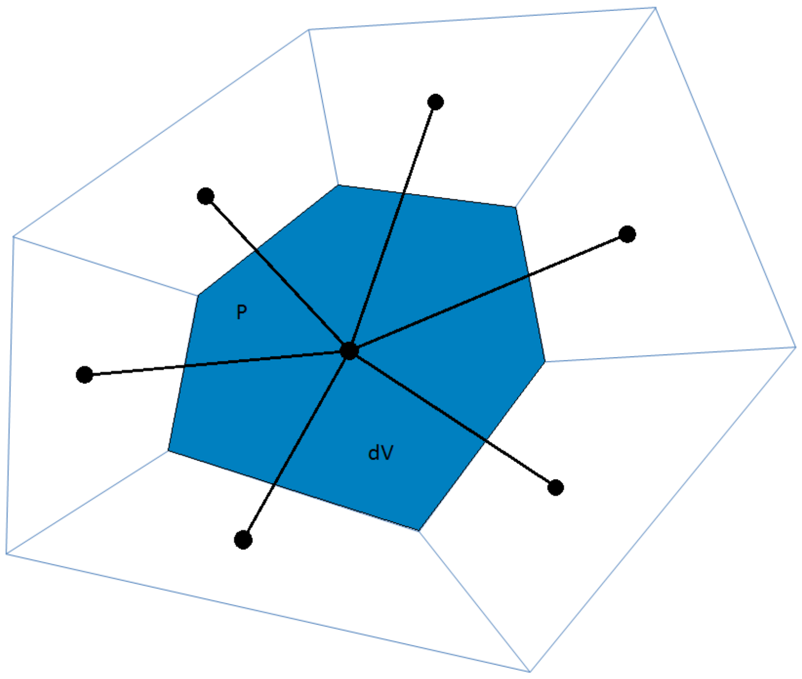


Figure 3.1: A finite difference grid.

In principle the cells can have any arbitrary shape, but more complex shapes introduce more complex calculations. An example of such a finite difference grid is illustrated in Figure 3.1

above. It shows the graphical representation of the computational molecule for the point P. The value of ϕ at P is related to the values of ϕ of the surrounding neighbouring points.

Structured and Unstructured grids

Computational fluid dynamics typically uses two different types of grids, structured grids and unstructured grids.

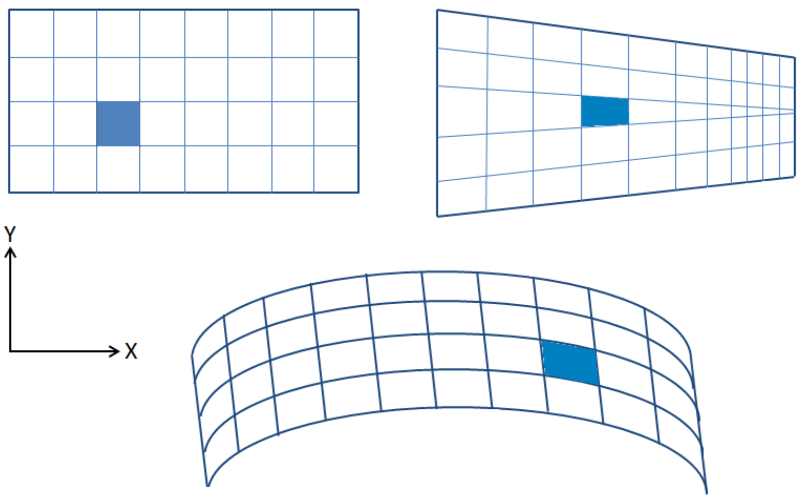


Figure 3.2: Structured Grids.

A structured grid (see Figure 3.2) is formed by the intersection of families of curvilinear lines that coincide with the shape of the physical domain at its boundaries. Which means it consists of a regular, rectangular grid that has been stretched and pulled to fit the dimension of the physical domain. The simplest case is a regular rectangular 2D grid. Each cell can be referred to with respect to their co-ordinates and each neighbouring cell can be identified by applying an offset to the coordinates, see Table (3.2) . The shaded blue rectangular areas in Figure (3.2) therefore have the coordinates (3;2),(4;3) and (8;2) from top left in clockwise direction. This also means that in a structured mesh one can always find a cell's corresponding neighbours by using regular offsets.

Table 3.2: Neighbouring cells coordinates

Cell neighbour	Coordinate
West Neighbour	(X-1,Y)
East Neighbour	(X+1,Y)
North Neighbours	(X,Y+1)
South Neighbours	(X,Y-1)

An unstructured grid (see Figure 3.3) can be formed from any arbitrary set of cells provided they are non-overlapping. The main difference compared to the structured grid is that there is no structured co-ordinate system, each cells therefore must have a unique identifier typically going from 1 to n where n is the number of cells in the physical domain. Hence the co-ordinate system doesn't apply any more the information about the neighbours is also lost, therefore each cells must have an adjacency table of its neighbours. For cell number 13 in Figure (3.3) this would be (7;17;21).

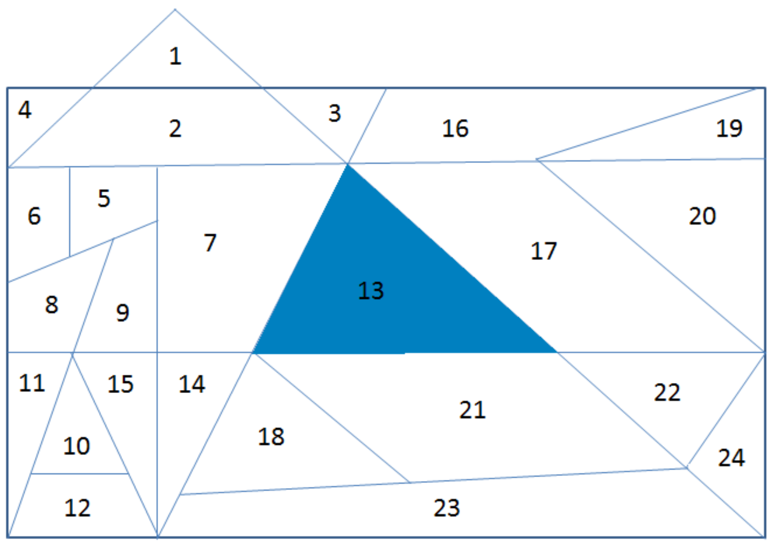


Figure 3.3: Unstructured Grids.

The unstructured grid allows for a greater flexibility in the formulation of the grid but requires an extra overhead because of the adjacency table for each cell. It also allows the grid to be applied to more complex geometries like arches and curves (see Figure 3.4) that couldn't be done with an ordinary NSWE[1] co-ordinate system. SMARTFIRE uses an unstructured grid.

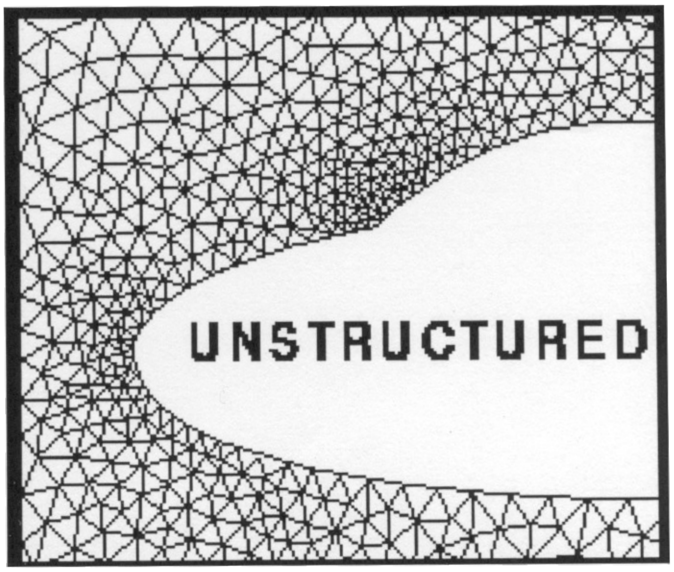


Figure 3.4: Unstructured mesh [156]

3.3.2 The Discretised General Conservation Equation

In this section, the FV formulation will be applied to the general conservation equation. To obtain the finite difference equations the general conservation equation is integrated, for a general variable ϕ located at a point P, (see Equation 3.1.1) over a typical cell volume δV located around that point P, and over a small period of time δt.

By rearranging the general conservation Equation (3.1.1) and integrating with respect to volume the following is obtained:

$$\iiint_V \frac{\partial(\rho\phi)}{\partial t} dV + \iiint_V \nabla.(\rho\underline{u}\phi - \Gamma_\phi \nabla\phi) dV = \iint_V S_\phi dV \tag{3.3.2}$$

[1]North South West East

In Equation (3.3.1) replace $\underline{\text{F}}$ with the quantity ($\rho\underline{\text{u}}\phi$- Γ_ϕ ∇_ϕ) to obtain:

$$\iiint\limits_V \nabla(\rho\underline{u}\phi - \Gamma_\phi \nabla\phi)dV = \oiint\limits_S (\rho\underline{u}\phi - \Gamma_\phi\nabla\phi)\underline{\hat{n}}dS \tag{3.3.3}$$

and,

$$\oiint\limits_S (\rho\underline{u}\phi - \Gamma_\phi\nabla\phi)\underline{\hat{n}}dS = \sum_{\substack{\text{All faces}\\ \text{of FV}}} \iint\limits_f \rho\phi\underline{u}\underline{\hat{n}}df - \iint\limits_f \Gamma_\phi\nabla\phi\underline{\hat{n}}df \tag{3.3.4}$$

and,

$$\iint\limits_f \rho\phi\underline{u}\underline{\hat{n}}df = \rho_f(\underline{u},\underline{\hat{n}})_f A_f\phi_f \tag{3.3.5}$$

and,

$$\iint\limits_f \Gamma_\phi\nabla\phi\underline{\hat{n}}df = \Gamma_\phi A_f\left(\frac{\partial\phi}{\partial x}n_x + \frac{\partial\phi}{\partial y}n_y + \frac{\partial\phi}{\partial z}n_z\right) \tag{3.3.6}$$

Using Equation (3.3.5) and (3.3.6) and substituting into Equation (3.3.2), Equation (3.3.7) is obtained:

$$\frac{\left[(\rho\phi) - (\rho\phi)^{old}\right]_P}{\delta t}V_P + \sum_{\substack{\text{All faces}\\ \text{of FV}}}\left(\rho_f(\underline{u}\hat{n})_f\phi_f - \Gamma_\phi\left(\frac{\partial\phi}{\partial x}n_x + \frac{\partial\phi}{\partial y}n_y + \frac{\partial\phi}{\partial z}n_z\right)\right)A_f = S_{\phi,P}V_P \tag{3.3.7}$$

In the above formulation it is assumed that the transient term can be approximated by the average rate of change between the current and previous time steps. It represents an implicit formulation for the variable ϕ_P. This means that the value of ϕ_P depends on the neighbours' ϕ values at the new time step. The explicit temporal discretisation will be described in Equation (A.2.1) in Section 3.3.3 in the Appendix.
To obtain the computational molecule ϕ_f and $\frac{\partial\phi}{\partial k}_f$ Equation (3.3.7) must be converted into values approximated by ϕ_P and ϕ_A, where A is a neighbouring node.

By assuming orthogonality the partial derivatives of ϕ can be estimated by using the following relation:

$$\frac{\partial\phi}{\partial x} = \frac{\phi_A - \phi_P}{d}\hat{d}_x \tag{3.3.8}$$

where $\underline{d}$ is the unit vector from node P to the adjacent node A and d is the distance between these two nodes. Similar relations exist for $\frac{\partial\phi}{\partial y}and\frac{\partial\phi}{\partial z}$.
If the mass flux through a surface is represented by FA(=$\rho_f(\underline{u}.\underline{n})_f A_f$) where ρ_f is the upwinded density, $(\underline{u}.\underline{n})_f$ is calculated using Rhie and Chow interpolation [157]

the diffusion coefficient can be represented by

$$DA = \left(\Gamma_{\phi,f} A_f \frac{n_x d_x + ny d_y + n_z d_z}{d}\right) \tag{3.3.9}$$

where $\Gamma_{\phi,f}$ is an average of the cell centred values.

Equation (3.3.7) can now be expressed as:

$$\frac{\left[(\rho\phi) - (\rho\phi)^{old}\right]_P}{\delta t} V_P + \sum_{\substack{\text{All faces}\\ \text{of FV}}} ((FA)_f \phi_f - (DA)_f(\phi_A - \phi_P)) = S_{\phi,P} V_P \tag{3.3.10}$$

In Equation (3.3.10) there still exists a facial value for ϕ_f. This is fixed by applying the upwind scheme [2]. If the convection flux is leaving the cell $\rightarrow \phi_f = \phi_P$, if convection flux is entering the cell $\rightarrow \phi_f = \phi_A$. Therefore Equation (3.3.10) becomes:

$$\begin{aligned} &\frac{\left[(\rho\phi) - (\rho\phi)^{old}\right]_P}{\delta t} V_P + \\ &\sum_{\substack{\text{All faces}\\ \text{of FV}}} \left[max((FA)_f, 0)\phi_P - max((-FA)_f, 0)_{\phi_A} - (DA)_f(\phi_A - \phi_P))\right] \\ &= S_{\phi,P} V_P \end{aligned} \tag{3.3.11}$$

There are a number of possible estimates for the value of ϕ_f [2] in Equation (3.3.10). These estimates are introduced by the addition of a function $A(|P|)$ where P is the Peclet number $(= FA_f/DA_f)$ and leads to:

$$\frac{\left[(\rho\phi) - (\rho\phi)^{old}\right]_P}{\delta t} V_P + \sum_{\substack{\text{All faces}\\ \text{of FV}}} \left[\begin{array}{c} max((FA)_f, 0)\phi_P - max((-FA)_f, 0)\phi_A \\ -(DA)_f A(|P|)(\phi_A - \phi_P) \end{array}\right] = S_{\phi,P} V_P \tag{3.3.12}$$

Table 3.3: Table of Differencing Schemes

Differencing Scheme	Formulae for $A(\lvert P\rvert)$
Upwind	1
Hybrid	$Max(0, 1 - 0.5\lvert P\rvert)$
Central Difference	$1 - 0.5\lvert P\rvert$
Power Law	$Max(0, (1 - 0.1\lvert P\rvert)^5)$
Exponential	$\lvert P\rvert / e^{\lvert P\rvert} - 1$

The various differencing schemes used for $A(|P|)$ in Equation (3.3.12) are given in Table (3.3):
Now by using the following notation $a_A = A(|P|).(DA)_f - max((-FA)_f, 0)$ and noting that $max((FA)_f, 0) = max((-FA)_f, 0) + FA_f$ Equation (3.3.12) becomes:

$$\left\{\frac{(\rho V)_P}{\delta t} + \sum_{All faces} a_A + \sum_{All faces} ((FA)_f)\right\}\phi_P = \sum_{\substack{\text{All adjacent}\\ \text{nodes}}} a_A\phi_A + \frac{(\rho\phi)_P^{old}V_P}{\delta t} + S_{\phi,P}V_P \tag{3.3.13}$$

The source term S_ϕ can be a function of ϕ and should be written in the linearised form:

$$S_{\phi,P} = S_C + S_P\phi_P \tag{3.3.14}$$

The continuity equation has the following form:

$$\frac{\partial\rho}{\delta t} + \nabla(\rho\underline{u}) = 0 \tag{3.3.15}$$

The discretised form of the continuity equation is therefore:

$$\frac{\left|(\rho) - (\rho)^{old}\right|_P}{\delta t}V_P + \sum_{\substack{\text{All Faces}\\ \text{of FV}}} (\rho_f(\underline{u}.\underline{\hat{n}}_f))A_f = 0 \tag{3.3.16}$$

By substracting the discretised continuity Equation (3.3.16) from the first coefficient of Equation (3.3.13) the following is obtained

$$\left\{\frac{(\rho V)_P^{old}}{\delta t} + \sum_{Allfaces} a_A - S_P V_P\right\}\phi_P = \sum_{\substack{\text{All adjacent}\\\text{nodes}}} a_A\phi_A + \frac{(\rho\phi)_P^{old} V_P}{\delta t} + S_C V_P \quad (3.3.17)$$

The above equation can be written as:

$$a_P\phi_P = \sum_{\substack{\text{All adjacent}\\\text{nodes}}} a_A\phi_A + b \quad (3.3.18)$$

where

$$\begin{aligned} a_P &= \left\{\frac{(\rho V)_P^{old}}{\delta t} + \sum_{\text{All faces}} a_A - S_P V_P\right\} \\ b &= \frac{(\rho\phi)_P^{old} V_P}{\delta t} + S_C V_P \end{aligned} \quad (3.3.19)$$

This is the computational molecule for point P (see Figure 3.1).

A set of Equations like (3.3.18) exists at every discrete point in the computational domain and this can be written in matrix form as:

$$\underline{\underline{A}}\,\underline{\phi} = \underline{b} \quad (3.3.20)$$

3.3.3 Explicit Discretisation

Explicit discretisation was not used for this thesis, but can be found for the sake of completeness in Section (A.2) of the Appendix.

3.4 Discretisation of The Momentum Equation

The conservation of momentum is shown in Equation (3.1.2). The next sections will describe discretisation of the individual terms of Equation (3.1.2) in detail.

3.4.1 Transient Term

The transient term of the momentum equation can be discretised over the control volume V as follows:

$$\begin{aligned}\iiint_V \frac{\partial(\rho u)}{\partial} dV &\approx V\left(\frac{\partial(\rho u)}{\partial t}\right)_P \\ &\approx V\left(\frac{(\rho_P u_P)_P - (\rho_P u_P)_P^{old}}{\Delta t}\right)\end{aligned} \tag{3.4.1}$$

In Equation (3.4.1) V stands for the control volume about the node P, Δt is the time step size, the "old" superfix refers to the values of the previous time step and ρ_P is the density at the node P.

3.4.2 Convection Term

$$\begin{aligned}\iiint_V \nabla(\rho u \underline{u}) dV &= \oiint_S \rho u \underline{u}.d\underline{S} \\ &= \sum_j \int_{F_j} \rho u \underline{u}.\underline{n} dS\end{aligned} \tag{3.4.2}$$

By using the divergence theorem the volume integral has been changed to a surface integral in Equation (3.3.1). The summation is applied over all faces that form surfaces S of the control volume V. The integration over the face is

$$\int_{F_j} \rho u \underline{u}.\underline{n} dS = A_F \rho_F (\underline{u}.\underline{n})_F u_F \tag{3.4.3}$$

The convection flux at the face is defines as $FA = \rho_F(\underline{u}.\underline{n})_F A_F$. If the face is an internal face to the domain Rhie and Chow interpolation (see Section 3.6.1) is used to provide an estimation for the velocity at the face. In case of a symmetry or wall boundary for the face there is no mass flux across the boundary $\rightarrow$ therefore $FA = 0$. If the boundary face is an inlet boundary condition then

$$FA = A_F \rho_F \sum_{k=x,y,z} v_k n_k \tag{3.4.4}$$

where $\underline{v}$ is the inlet velocity. There is no contribution due to an outlet fixed pressure boundary condition.

3.4.3 Diffusion Term

$$\begin{aligned}\iiint_V \nabla(\mu\nabla u)dV &= \oint_S \mu\nabla u.\underline{n}dS \\ &= \sum_j \int_{F_j} \mu\nabla u.\underline{n}dS \end{aligned} \tag{3.4.5}$$

By substitution in Equation (3.3.9) in Section 3.3.2 the following is obtained:

$$DA = (\mu A)_f \frac{n_x d_x + n_y d_y + n_z d_z}{d} \tag{3.4.6}$$

The following relation handles the boundary condition of a wall or an inlet.

$$DA = \frac{\mu_f A_f}{d_l} \tag{3.4.7}$$

3.4.4 Pressure Gradients

The pressure gradient is calculated using Gauss' divergence theorem:

$$\begin{aligned}\iiint_V \nabla_k P dV &= \oiint_S \underline{P}.\underline{n}_k dS \\ &= \sum_j \int_{F_j} \underline{P}.\underline{n}_k dS \end{aligned} \tag{3.4.8}$$

3.5 Discretisation of the Radiation Model

The following sections will briefly discuss the discretisation of the two less computationally expensive models, as mentioned in Section (3.1.6), the six-flux and discrete transfer model respectively.

3.5.1 Six Flux Model

In this model the discretisation around a solid angle of 4π results in 6 fluxes along the positive and negative co-ordinate directions.

Traditional six-flux models during discretisation combine the fluxes in positive and negative directions. The big advantage of this model is that it is very easy to incorporate it into the FV scheme used by many CFD codes. SMARTFIRE uses Jia's modified Six-Flux Radiation model [158] in which scattering is neglected ($s = 0$). This can be done as the main purpose of SMARTFIRE is to simulate scenarios e.g enclosed fires where scattering is unimportant [159]. According to Jia the traditional formulation can lead to ill-posed problems which is avoided by solving the original first order equations instead. The main disadvantage of this method is that six equations have to be solved instead of the traditional three.

The six flux model was not used in this work, instead the radiation model described in the next Section (3.5.2) was utilised.

3.5.2 Discrete Transfer Model (Multi-ray)

In situations where a high accuracy of the heat flux at solid surfaces is required e.g flame spread over solid surfaces or structural interactions with the fire the six-flux model is not sufficiently accurate to be used. It is intended for situations where the radiative heat loss from flames is the dominant factor e.g non-spreading fires. The discrete transfer model with it's increased angular resolution is able to fulfil these requirements, e.g spreading flames.

The method was first introduced by Lockwood and Shah [155]. SMARTFIRE offers the possibility to discretise the 4π solid angle along 6,14,24 or 48 rays. The radiation equation without scattering along one ray direction is given by

$$\frac{\partial}{\partial l} I(\Omega, r) = -\alpha I(\Omega, r) + \alpha I_b(r) \tag{3.5.1}$$

If the computational domain is divided into individual zones (not the sub-domains used for parallel processing as in Section (4.1), zones refer to Exodus/SMARTFIRE hazard zones), which essentially represent an averaged value over the cells of a zone, where optical properties and the temperature are constant, so Equation (3.5.1) turns into the following recurrence equation

$$I_{n+1} = I_n e^{\alpha \delta l} + I_b \left(1 - e^{-\alpha \delta l}\right) \tag{3.5.2}$$

Where I_{n+1} stands for the radiation intensity leaving the zone and I_n the radiation intensity entering the zone via the ray. δl stands for the length of the ray segment within the zone. In SMARTFIRE zones are simply mapped 1 to 1 to each of the computational cells which are part of the zone, other CFD models define the mesh for zones differently, coarser, but still average the values over the cells included within the zone.
The soldid 4πangle is divided up into n parts (n being the number of rays) and in each part a ray is projected. As a rule of thumb, the more rays being used the more accurate the radiation distribution at the cost of computational complexity and power. The ray distribution has to follow certain rules which have been described by Lathrop and Carlson [160].
The only two equations left to fully describe the discrete transfer radiation model is the wall boundary condition

$$I = \frac{\varepsilon_w I_B}{\pi} + (1 - \varepsilon_w) I^- \tag{3.5.3}$$

where Γ is the incident radiation on the wall. And the energy transfer into the enthalpy equation

$$S_{h,DTM} = \sum_{rays} (I_{n+1} - I_n) \overrightarrow{\Omega}.\Delta\overrightarrow{A}\Delta\Omega \tag{3.5.4}$$

3.6 Staggered And Co-Located Meshes

When the first solution schemes for pressure and velocity were first implemented it was discovered that the velocity and pressure fields oscillated. The reason for this was the strong coupling between pressure and velocity (as the velocity goes into the solution for pressure). Therefore when the equations were discretised alternate nodes would be decoupled and the so called checkerboard effect accrued. To overcome these effects two methods have been devised.

First, the staggered grid approach, where all the values are solved at cell centre except the velocity components which are solved at the cell faces. The main advantages for this method are:

- No interpolation as velocity is calculated at the cell faces already.
- The checkerboard effect gets eliminated by using the adjacent cells for the descretisation instead of the difference between alternating nodes.
- The velocity calculation at the cell faces is applicable as the velocity is situated half way between adjacent pressure cells, which are the driving force of the velocity.

The main disadvantage is the need to store a grid for each velocity component u,v,w. So increased memory-usage and book-keeping is required. Also unstructured meshes can make it particularly difficult to define a staggered mesh.
The other approach is the co-located (unstaggered) grid and the use of a special interpolation for the velocities devised by Rhie and Chow [157] which will be described in Section 3.6.1.

The main advantages of Rhie and Chow's method are

- Less memory required.
- The geometric bookkeeping is much simpler when using unstructured meshes.
- Removes the checkerboard effect.

The main disadvantage here is that the cell centred value calculated need not satisfy continuity. Only the facially interpolated values are constrained to this.

3.6.1 Rhie And Chow Interpolation

Rhie and Chow's method [157] is used to interpolate cell centred velocities to their cell faces without the checkerboard effect mentioned by Patankar [2], which was avoided by using the staggered grid approach. The method uses the discretised momentum equation.

The discretised form of the u momentum equation for a control volume about a node P can be expressed as

$$a_P u_P + (\nabla P)_P = \left(\sum a_{nb} u_{nb}\right)_P + S_P \tag{3.6.1}$$

or for an adjacent node

$$a_A u_A + (\nabla P)_A = \left(\sum a_{nb} u_{nb}\right)_A + S_A \tag{3.6.2}$$

Following the conservation principle of the finite volume formulation the u velocity component at a point on the face (f) between these two nodes must also have a corresponding discretised momentum equation of a similar form

$$a_f u_f + (\nabla P)_A = \left(\sum a_{nb} u_{nb}\right)_f + S_f \tag{3.6.3}$$

Equation (3.6.1) and (3.6.2) can be used to give an approximate solution of Equation (3.6.3). It is assumed that the terms on the right hand side of Equation (3.6.3) may be approximated by a linear weighted average of Equation (3.6.1) and (3.6.2).

$$a_f u_f + (\nabla P)_f = \overline{\left(\sum a_{nb} u_{nb}\right)_f} + \overline{S_f} = \overline{a_f u_f} + \overline{(\nabla P)_f} \tag{3.6.4}$$

The over line represents the weighted linear average for the variable. It will now be further assumed that $a_f \approx \overline{a_f}$ so that

$$u_f = \overline{u_f} + \overline{d_f}\left(\overline{(\nabla P)_f} - (\nabla P)_f\right) \tag{3.6.5}$$

where

$$\begin{aligned} \overline{u}_f &= \alpha u_P + (1-\alpha) u_A \\ \overline{\nabla P}_f &= \alpha (\nabla P)_P + (1-\alpha)(\nabla P)_A \\ \nabla P_f &= A_f n_x (P_P - P_A) \\ a_f &= \alpha a_P + (1-\alpha) a_A \\ \overline{d_f} &= \frac{1}{a_f} \end{aligned} \tag{3.6.6}$$

3.7 Solution Methods

After deriving all the equations, the system $\underline{\underline{A}}\underline{\phi} = \underline{b}$ now needs to be solved . The variable to be solved is represented by ϕ and is generally unknown except for the boundary conditions. As most equations are heavily coupled, the system can not be solved sequentially and must therefore be solved iteratively.

The major challenge solving these physical equations comes from the strong coupling between the pressure and velocity fields. The momentum equation is, as shown in Section 3.6, dominated by the pressure term, therefore an accurate knowledge of the pressure field is essential. However the difficulty with the pressure field is that it is only indirectly specified by the continuity equation. For a good estimate of the pressure field an accurate measurement of the velocity field for the continuity equation is required.

To solve the problem Patankar and Spalding came up with an iterative solution scheme called "Semi Implicit Method for Pressure Linked Equations" or short SIMPLE [161]. The method has been well adopted by CFD modellers and other variations have been derived such as the SIMPLER [2] and SIMPLEC [162] methods.

The basic work flow for the SIMPLE method is

- Initial guess for all the variables.
- Solve the enthalpy equation.
- Solve radiation, turbulence and any extra equations, if required.
- Solve the momentum equations for each phase obtaining a new velocity field. This velocity field will generally not satisfy the joint continuity equation.
- Solve the continuity equation to get the pressure correction.
- Calculate the pressure correction to the velocity components of each phase. These corrected velocity field will generally not satisfy the momentum equations.
- Iterate over again from point 2 (solve enthalpy equation) until convergence is achieved.

- Repeat previous steps for all time steps.
- End.

3.7.1 The Mass Continuity Equation

As mentioned in Section (3.1.3) the mass continuity equation takes the following form

$$\frac{\partial \rho}{\partial t} + \nabla(\rho \underline{u}) = 0 \tag{3.7.1}$$

and the discretised form:

$$\frac{\rho_P V_P - \rho_P^{old} V_P^{old}}{\Delta t} + \sum_{faces} (\rho \underline{u}.\underline{n})_f A_f = 0 \tag{3.7.2}$$

To satisfy mass continuity during the iterative process a pressure correction is applied to the pressure field as Equation (3.7.2) will not always sum up to zero. The pressure correction is applied by using Rhie and Chows interpolation method mentioned in Section (3.6.1). Rhie and Chow states that

$$u_f = \overline{u}_f + \overline{d}_f \left(\overline{\nabla P} - \nabla P \right)_f \tag{3.7.3}$$

this leads to:

$$u_f = \frac{n_x A_f}{\overline{a_{P,u}}} \left(\left(P + P^{'} \right)_P - \left(P + P^{'} \right)_A \right) + \overline{u}_f + \overline{d \nabla P_f} \tag{3.7.4}$$

where $P^{'}$ stands for the pressure correction at point P, by using Equation (3.7.4) a computational molecule for $P^{'}$ can be formed, which then can be solved like any other matrix equation.

3.7.2 Pressure And Velocity Correction

The pressure correction from the mass continuity equation from Section (3.7.1) need to be applied to the pressure field calculated in the previous iteration. Furthermore the corrections for the velocity field must also be obtained. It is know that $u = u^* + u'$, where u stands for the exact velocity field ,u^* the current velocity field and u' the error. Referring to the discretised momentum Equations (3.7.5) and (3.7.6):

$$a_P u_P = \sum a_A u_A - \nabla P + S_P \tag{3.7.5}$$

$$a_P u_A^* = \sum a_A u_A^* - \nabla P^* + S_P \tag{3.7.6}$$

subtracting Equation (3.7.5) from Equation (3.7.6) gives:

$$a_P u_P^{'} = \sum a_A u_A^{'} - \nabla P + \nabla P^* \tag{3.7.7}$$

The term $\sum a_A u_A^{'}$ can be dropped as it tends to zero towards convergence. Which leads to the following equation

$$a_P u_P^{'} = (\nabla P - \nabla P^*) \tag{3.7.8}$$

For an internal cell

$$\nabla P = \sum_{All faces} A_f \left(\alpha P_P + (1-\alpha) P_A\right) n_{x,f} \tag{3.7.9}$$

and

$$\nabla P^* = \sum_{All faces} A_f \left(\alpha P_P^* + (1-\alpha) P_A^*\right) n_{x,f} \tag{3.7.10}$$

then

$$\nabla P^{'} = \nabla P - \nabla P^* = \sum_{All faces} A_f \left(\alpha P_P^{'} + (1-\alpha) P_A^{'}\right) n_{x,f} \tag{3.7.11}$$

The same applies for velocity components v and w.

3.8 Boundary Conditions

To correctly model fire and CFD problems in general a set of boundary conditions are required. The boundary conditions normally used are:

- Inlet
- Wall
- Pressure Boundary

3.8.1 Inlet

The inlet specifies the velocities that are going into the domain. It is also necessary to specify the other properties for the fluid (ϕ_{inlet}). Looking at a cell that has an inlet a term FA_{inlet} is specified and DA_{inlet} can be calculated which leads to the calculated a_{inlet} coefficient. The following sources will then be added to any of the discretised equations. Equation (3.8.1) shows a fixed value boundary condition.

$$\begin{aligned} S_P &= a_{inlet} \\ S_C &= a_{inlet}\phi_{inlet} \end{aligned} \tag{3.8.1}$$

3.8.2 Wall Boundary Condition

Walls are used to confine the flow within a defined geometry. No mass flux can travel through a wall $\rightarrow$ $FA_{wall} = 0$. It is also assumed that velocities parallel to the walls are zero.

3.8.2.1 Momentum Equation

If a cell has a face which represents a wall boundary the shear force F_S in the discretised momentum equation is described by

$$F_S = -\tau_{wall} A_{wall} \tag{3.8.2}$$

where $\tau_{wall} = \mu \frac{u_P}{\Delta y_P}$ stands for the near wall laminar flow and Δy_P for the distance from node P to the wall.

Which leads to the following additional source term being added to the discretised momentum equation

$$S_P = -\frac{\mu}{\Delta y_P} A_{wall} \tag{3.8.3}$$

In case of turbulent flow, the above relation for laminar flow will be used as P might lie in the laminar sub-layer. If P falls within the turbulent layer special log-law functions are used to model the shear force. To determine where P lies Equation (3.8.4) is used.

$$y^+ = \frac{\sqrt{\rho \tau_w}}{\mu} \Delta y_P \tag{3.8.4}$$

The y^+ in Equation (3.8.4) above determines the near wall flow is laminar or turbulent. The limits for y^+ are shown in Table (3.4).

Table 3.4: y^+ to determine near wall flow

$y^+ < 11.63 \rightarrow laminarflow$
$y^+ > 11.63 \rightarrow turbulentflow$

The following Equations (3.8.5) and (3.8.6) show the turbulent shear stress. The source term in the discretised momentum equation is derived in the same fashion as that for laminar shear force.

$$\tau_{wall} = \frac{\rho C_\mu^{1/4} k_P^{1/2} u_P}{u^+} \tag{3.8.5}$$

u^+ is defined as

$$u^+ = \frac{1}{\kappa} ln(Ey^+) \tag{3.8.6}$$

where κ and E are model constants.

3.8.2.2 Enthalpy Equation

The convective heat flux into a wall for a wall surface temperature, T_w is defined as

$$q_s = h_c (T_P - T_w) \tag{3.8.7}$$

where h_c is the laminar heat transfer coefficient

$$h_c = \frac{k}{\Delta y_P} \tag{3.8.8}$$

k stands for the turbulent kinetic energy and

$$T^+ = \frac{\sigma_T}{\kappa} ln \left(E_T y^+ \right) \tag{3.8.9}$$

T_w can either be prescribed or calculated based on the material and properties of and the net heat flux on the wall. This leads to an iterative process as the net flux depends on T_w. An efficient calculation can be found in detail in Jia [158].

3.8.2.3 Turbulence Equation

3.8.2.3.1 Kinectic energy equation No modifications are required for the k equation apart from the turbulent viscosity at the wall which depends on the wall shear stress instead of the standard Equation (3.1.18).

3.8.2.3.2 Dissipation Rate equation The ε value in the near wall cell for the dissipation rate is given by

$$\varepsilon = \frac{C_\mu^{3/4} k^{3/2}}{\kappa \Delta y_P} \tag{3.8.10}$$

3.8.2.4 Pressure Boundary

A pressure boundary is needed in case the exact flow details are not known but instead the pressure boundary values are. Typically used in FSE to allow mass to vent in/out of a defined fire domain. It is important to make sure that the area which the modeller is interested in is far enough away from the pressure boundary so it can not falsely impact the flow values. This generally requires extending the region beyond the actually modelled region. As pressure P is prescribed at the boundary the pressure correction $P^{'}$ is zero.

3.9 Solvers

After all the above equations are discretised, using an implicit formulation, they now need to be solved. This can be done through numerical matrix solvers. Another but mostly not practical way of solving the matrix would be through direct methods, such as the Gaussian elimination. The method requires a lot of memory and the time spent solving the system. So most CFD codes make use of iterative solvers, including SMARTFIRE. Iterative solvers calculate values based on previously calculated values, this is repeated until the system reaches a state where the values are not changing any more (within a specified accuracy level), this state is called convergence. SMARTFIRE uses different kind of solvers, e.g Jacobi Over Relaxation (JOR), Successive Over Relaxation (SOR) or the conjugate gradient method. This thesis will only focus on the Jacobi Over Relaxation solver, a detailed explanation for that will be given in Section (4.3).

Serial SMARTFIRE generally uses the JOR solver only for the momentum equation, the other variables use the residual-sor implementation for pressure.

3.9.1 JOR Method

The Jacobi Over Relaxation (JOR) method is based on the iterative method. The $i+1$ iteration value of variable ϕ in element P $\left(\phi_P^{i+1}\right)$ is obtained by using:

$$\phi_P^{i+1} = \frac{1}{A_{PP}}\left(b_P - \sum_{k\neq P} A_{pk}\phi_k^i\right) \tag{3.9.1}$$

The JOR methods adds an over relaxation term

$$\phi_P^{i+1} = (1-\alpha)\,\phi_P^i + \frac{\alpha}{A_{PP}}\left(b_P - \sum_{k\neq P} A_{pk}\phi_k^i\right) \tag{3.9.2}$$

Although over relaxation is used to speed up the convergence $(\alpha > 1)$ it is more common in FSE to under relax the solver $(\alpha < 1)$ to give greater stability. The disadvantage of this method is that it is slow compared to others but very stable.

The basic work flow for the solver is

1. for all cells (1 to number of cells), apply Equation (3.9.2) to calculate a value for the chosen variable ϕ_P.
2. repeat previous step until convergence is achieved or the maximum number of iterations is reached.

The new value of ϕ_P^{i+1} is dependent on the previous neighbouring values of ϕ_k^i and is independent of ϕ_k^{i+1}. This means the cell ordering makes no difference to the obtained value of ϕ_P^{i+1}. The differences that can occur are due to machine precision and numerical problems which will be discussed in Chapter (5).

Chapter 4

PARALLEL IMPLEMENTATION OF SMARTFIRE

After discussing all the different parts of the CFD model in the previous chapters, an efficient way of implementing it will be given in this chapter. The most straight forward way of running a CFD simulation is to run it using a serial implementation. A serial simulation processes the problem cell after cell. The other method is running the model in parallel which is due to the accessibility of powerful computer systems becoming more and more important. A parallel implementation will split up the problem into sub-problems which then can be solved simultaneously. The goal of parallel techniques is solving a problem in less time or solving a larger problem in the same time. This thesis focuses on improving the execution speed by executing the necessary calculation on GPU hardware. The main concept is transferring the decomposed problem onto the GPU hardware, where each Block can run simultaneously, see Figure 4.1. Typical application areas are science and engineering e.g meteorology, physics, bioscience as well as data mining, financial modelling, film animation etc.

This chapter describes the steps necessary in the process of applying parallel concepts to prepare the CUDA GPU execution in SMARTFIRE.

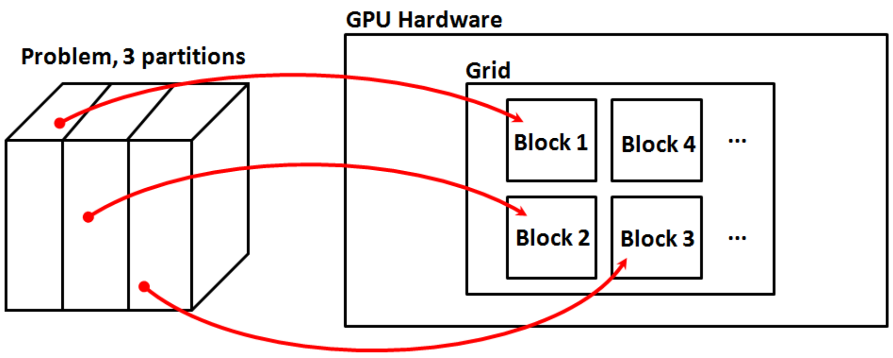


Figure 4.1: Problem, decomposed to 3 domains gets mapped to GPU hardware.

4.1 Mesh Partitioning

As described above to run problems in parallel the problem has to be divided in individual partitions that then can be solved simultaneously.

To efficiently divide up a large mesh, the number of cell edges on a partition wall has to be minimized. This is done by optimising the surface of each partition until the lowest surface area is achieved. As mentioned before, SMARTFIRE uses an unstructured mesh, see Figure 3.3. Due to the fact that graph partitioning algorithms for unstructured meshes are very complex, one of the well established available tools can be used e.g Metis [163], Jostle [164], Chaco [165] and Scotch [166]. In this thesis Metis will be utilised as it is has been used within the research group by Grandison [167] and Mohedeen [168] successfully, is open-source, fast and stable. Figure 4.2 shows an example of how a 125 cell cube has been partitioned into three sub domains using Metis.

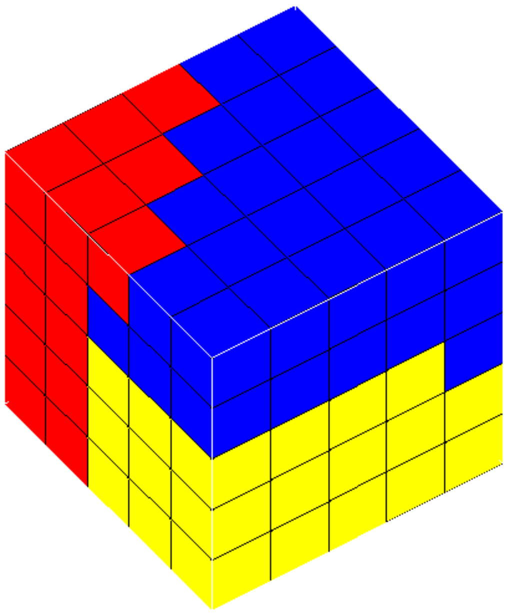


Figure 4.2: Example three dimensional mesh partitioning of a 5x5x5 cell domain.

A detailed description of the required Metis input format can be seen in Section A.4 in the Appendix.

The CUDA implementation of SMARTFIRE, as mentioned in Section 2.4.2.3, has the limitation that domain sizes are limited to a maximum of 1024 cells. Unfortunately Metis does not offer the option to specify a maximum domain size instead it only accepts the number of partitions. But as mentioned above, the goal of Metis is to minimize the surface between the adjacent partitions, the final number of domains often slightly differs from the input number. To ensure that the number of cells does not exceed 1024 Metis is repeatedly tested with an increasing number of j sub-domains until all sub-domains are less than 1024 cells in size, see Listing 4.1. As Metis runs very fast and normally only 1 to 5 iterations are necessary this was a sufficient work-around.

Listing 4.1: Metis iterative implementation.

```
        j = # cells / 1024

        fulfilled = false

        while( fulfilled = false )

                ...Metis run

                if (largest_domain < 1024) fulfilled = true
                else initial_guess++
```

Where j is the initial guess, obtained by simply using $N/block\_size$. If iterations return block sizes above the one specified j gets increased and Metis runs again.

4.2 Halo Cells

As mentioned in Chapter 3.3.2 in the discretisation process the solution value ϕ of the corresponding variable depends on the ϕ-values of the neighbouring cells. Which is shown in the matrix equation $\underline{\underline{A}}\underline{\phi} = \underline{b}$ from Section 3.2. However as the domains are now physically separated there is no information exchange from one domain to the other any more. This simply means the cells on the boundaries of each domain will not get any values passed in any more from their neighbours. To solve this issue halo cells will be added to each domain. In the example in Figure 4.3 we have a domain consisting of 9 by 5 cells. These cells are split up into three domains as shown in Figure 4.4. The lightly shaded grey cells represent the halo values. The black and white arrows indicate that the last row of calculated values from one domain represents the halo region for the adjacent domain. It also shows that halo values will only be read in and not calculated by the domain, this means there is no additional calculation effort but additional storage and mapping arrays for the halo cells are necessary.

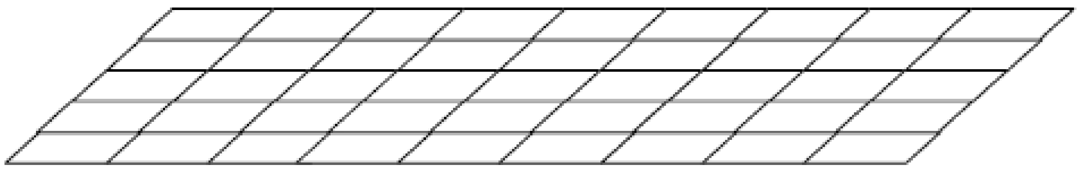


Figure 4.3: 9x5 domains [167]

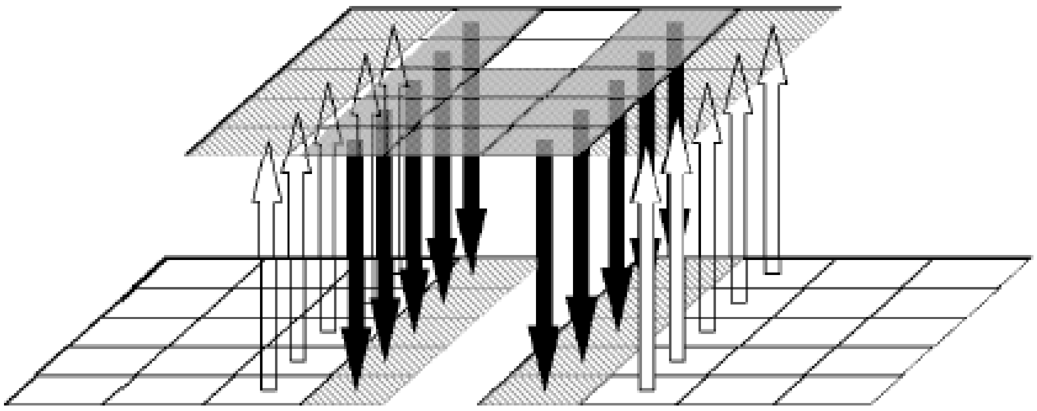


Figure 4.4: Domain subdivided with halo cells added for communication at the domain partitions [167]

The fact mentioned in Section 4.1 that the surface optimisation done by Metis minimizes the number of halo cells and therefore communications which only occupies resources is important, is illustrated in Figure 4.5 showing the difference in halo cells required for different partitioning styles.

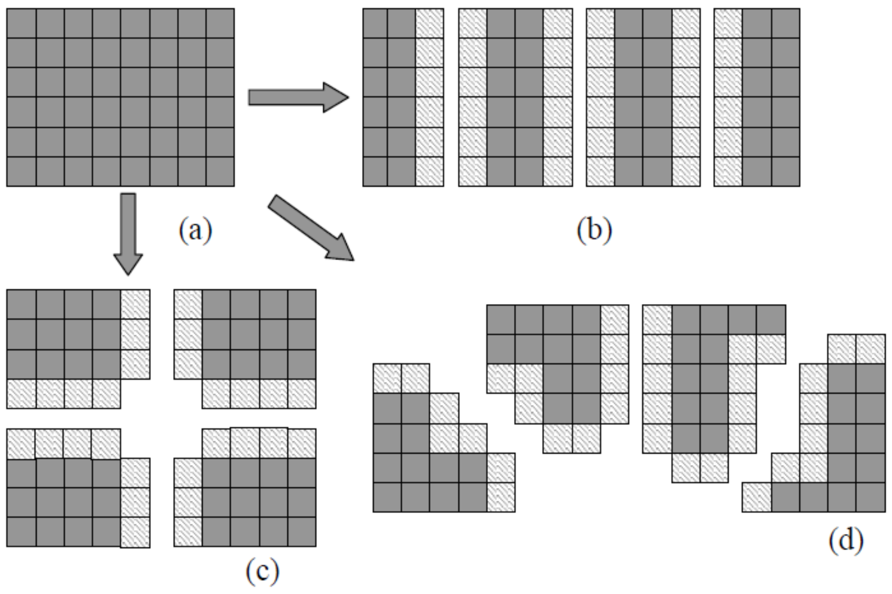


Figure 4.5: Different partitioning approaches requiring different halo cells [167]

It can be seen, that depending on the number of domains and the shape chosen the number of halo cells varies, Figure 4.5 (b) has 36 halo cells, (c) has 28 halo cells and (d) has 36 halo cells as well.

The halo communication in this thesis is implemented as a separate kernel function which is called after each solver iteration. It had to be implemented in this way to ensure that all the solver calculations have finished before the halos get updated. As said above the halos do not have to be calculated separately they are essentially taken from one domain (where they are calculated values, non-halos) and updated in the adjacent domains.

The halo update in the code, has been benchmarked to work out how much time is spent doing the update compared to the actual solving. On average the update takes less than 12 per cent per iteration (0.0004 seconds vs. 0.0031 seconds) and is therefore almost negligible. This concept has no effect on the solution process of the CFD code as every update finishes before the values are used again in the next sweep.

4.3 JOR Solver

After all the parallel specific concepts were discussed in previous sections they will now be used to explain the actual implementation of the numerical solver of the SMARTFIRE CUDA CFD engine. Code Listing 4.2 shows the serial version of the JOR solver used in SMARTFIRE. It can be seen in line 1 (for loop) of the code that the solver works sequentially over the number of cells in the domain. Therefore the described solving step will be repeated until all cells have been updated. First of all the source term (b_vector entry) gets stored in "curval". Followed by two simplifications to resolve multi-dimensional structures in the code shown in line 3 and 4 which are not essential for the code. Line 5 loops over the number of neighbouring cells present for a cell. Now in line 7 and 8 is where the actual solving happens, which follows the formula shown in Equation (3.9.1). The cumulative sum of the previous cell value * the corresponding system matrix entry is added up and subtracted from the source term entry. Followed by a division by the ap coefficient, which in SMARTFIRES' case is an unnecessary step as the system matrix is normalized and therefore always one. The final step is updating the old cell value with the newly calculated one, which can bee seen in Line 11.

Listing 4.2: Serial solver code.

```
 1  for ( cell_num = start_cell; cell_num <= end_cell; cell_num++ ){
 2                  curval = source[cell_num];
 3                  l_s_coeff = s_coeff[cell_num];
 4                  l_c_index = c_index[cell_num];
 5                  for ( indx = 2; indx <= c_index; indx++)
 6                  {
 7                      curval -= l_s_coeff[indx] *
 8                       previous_value[l_c_index[indx]];
 9                  }
10                  curval /= l_s_coeff[1];
11                  local_cell_value[cell_num] = curval;}
```

The one important thing to highlight in code Listing 4.2 is the sequential way of executing the solving steps for each cell. While the product in line 7 and 8 is calculated, the other calculations over the remaining cells have to wait for the current cell to finish. These sequential executions are where the CUDA programming model is used to make full use of the available hardware resources to reduce the time needed to solve each computational stencil. Figure 4.6 shows how two cells would be calculated, in serial and in CUDA. Looking at the time line, it can be seen that the serial version calculates the new cell values (Cell1 and Cell2) one after the other, the CUDA version performs the same mathematical operation but for the two cells simultaneously. The calculation uses the sum over the product of previous value times coefficient for each neighbour. Looking at the serial code in Listing 4.2 this can be seen in the for-loops in line 1 and line 5, CUDA does not need the outer loop any more and executes the inner loop for each cell simultaneously.

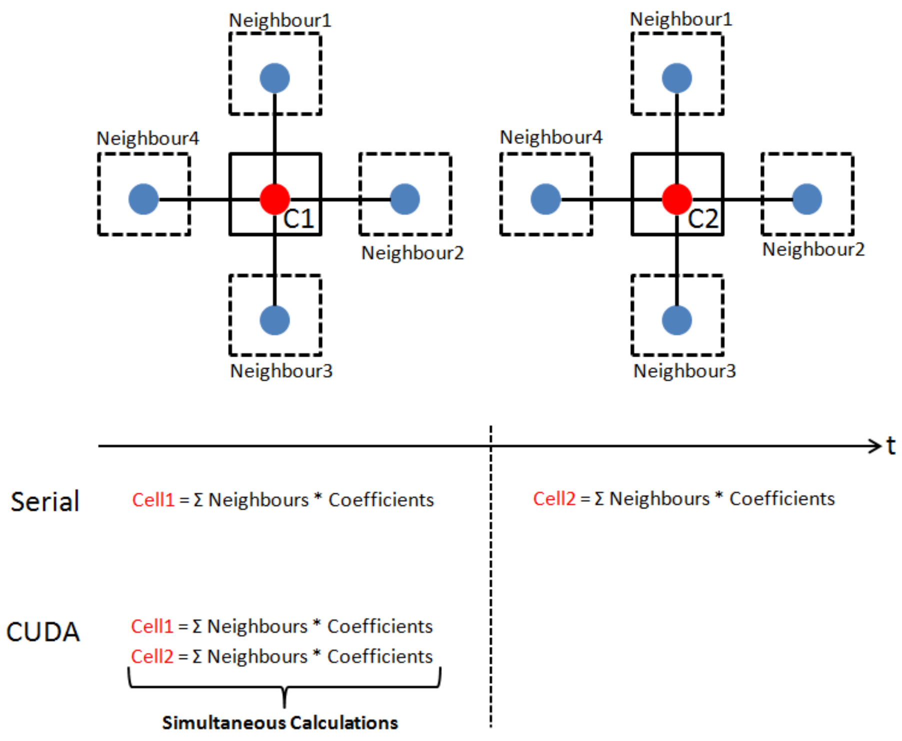

Neighbour1
Neighbour4
C1
Neighbour2
Neighbour3
Neighbour1
Neighbour4
C2
Neighbour2
Neighbour3
t
Serial
Cell1 = Σ Neighbours * Coefficients
Cell2 = Σ Neighbours * Coefficients
CUDA
Cell1 = Σ Neighbours * Coefficients
Cell2 = Σ Neighbours * Coefficients
Simultaneous Calculations

Figure 4.6: Serial and parallel calculation of a JOR solver step, for cells C1 and C2 using neighbouring values Neighbour1 to Neighbour4.

The adjusted solver code can be seen in Listing 4.3. Lines 1 to 11 are performance improvements and will be discussed in detail in Section 4.3.1. The first line which resembles the actual solver logic is line 13, the if statement ensures that each domain stays within its limits, this is necessary as the Metis domains mentioned before can have different sizes but the CUDA block_ size (as mentioned in Section 2.4.2.3) stays the same over all blocks. In general the logic remains the same as in code Listing 4.2, so the first step is to get the source term value (line 15) and subtract the previous neighbouring values times the corresponding system matrix entry (line 16 to 19). Followed by the division using the ap coefficient (line 20). The ”id” index is a global CUDA identifier (an unique identifier for each element), which is calculated by using block_id and thread_id ($block\_id * block\_size + thread\_id$), Figure 4.7 shows the ids for blocks and threads. Thread ids are unique within each block and go from 0 to block_size -1 and each block has a block_id from 0 to number of domains. These global identifiers are used to access/ write to the correct memory locations, as each thread executes simultaneously. To summarize the differences in the two implementation methods, the CUDA model executes all the individual cell updates in parallel with all its available hardware resources in difference to the cell by cell execution of the serial model. Therefore eliminating the idle times of the serial version and as a result reducing the time required for the execution.

Listing 4.3: CUDA solver code.

```
1  __shared__ Float_Type u_sh[1026];
2
3  u_sh[id]          = u_d[id];
4  u_sh[id+offset] = u_d[id+offset];
5
6  if(id == 0)
7  {
8           u_sh[1024] = 1;
9           u_sh[1025] = 0;
10 }
11 __syncthreads();
12
```

```
if(id <domain_sizes_d[bx])
{
        u_d[id]  = b_vector_d[id];
        for(int index =0;index<6;index++){
        u_d[id] -=
                u_sh[map_d[(id+index)]]  * sys_mat_d[id+index]
        }
        u_d[id] /= sys_mat_d[id+6];
}
```

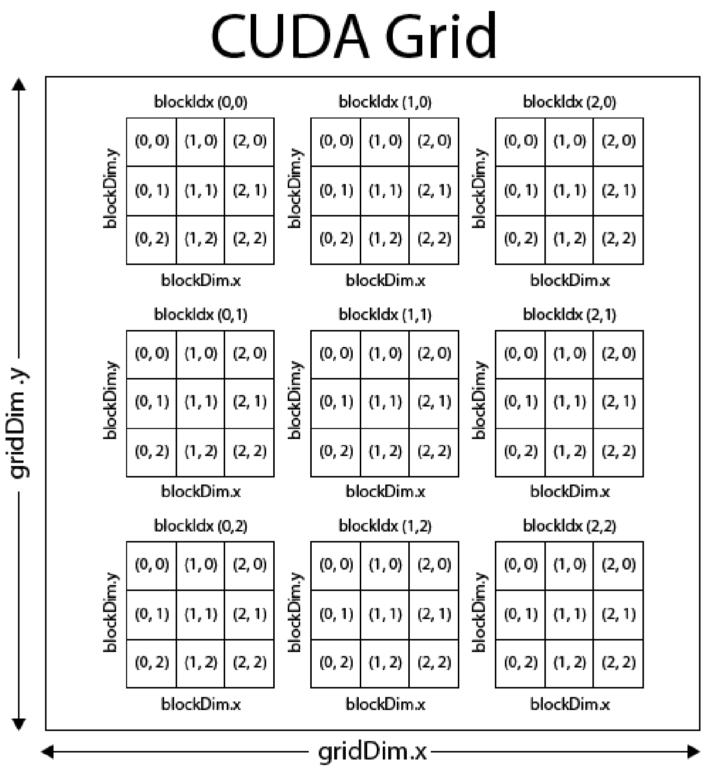


Figure 4.7: Block and thread numbering schemes used by CUDA [169]

4.3.1 Shared Memory Concept

The solver implementation shown above theoretically could be up to (number of cells) times faster than the serial code, if executed in parallel. This would mean that every cell can be calculated simultaneously compared to the sequential execution in serial. Unfortunately this is not the case as hardware limitations i.e number of processors, number of concurrently running threads, memory limitations, etc. will reduce the speed-up drastically. Therefore speed-ups can be achieved up until the point where all calculation resources (cores) are occupied and tasks will be queued up.

This section will show how the solver code can be optimised to further reduce the execution time inside the solver. Figure 4.8 shows the different kinds of memory i.e global/constant, shared, registers, accessible to different groups of data on the GPU. So far the data was kept in global memory meaning all the calculations are executed on the slowest kind of memory available. Unfortunately the faster types of memory e.g constant memory and shared memory (registers) are very limited and will be far too small to store all the data. As the slowest type of memory is also the biggest part available it will have to be used to store the multi GBs of data. To make use of the so far unused, fast types of memories a combination of global and shared memory has been developed. Shared memory is about 10 times faster (~1.7TB/s, global memory ~150GB/s).

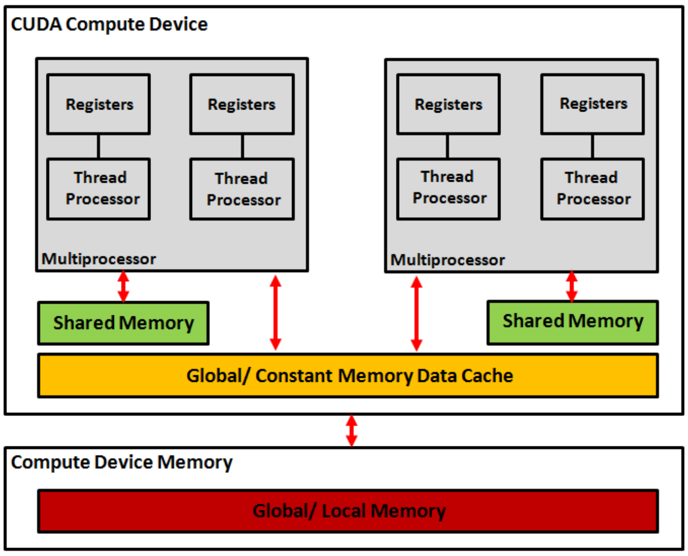
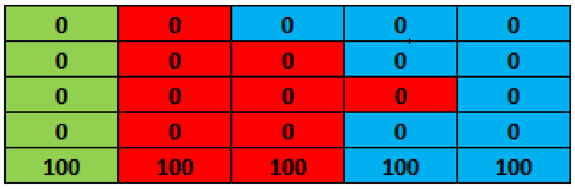


Figure 4.8: Different types of memory available on a CUDA device.

The downside of shared memory is that it is only available on a per block basis (not visible outside the corresponding block), limited in size(50kb per block), and only has the lifetime of a kernel call. The size limitation is not that much of a problem as with the maximum block_size of 1024 elements (Fermi and later generations) and we are using float numbers only 1024*4byte = 4.096kb of memory are occupied. To avoid unnecessary time to populate the shared memory for every solver kernel call the number of values has been further reduced. As multiple cells per block share the same neighbouring values they only get stored once in shared memory and get accessed by using a cell to shared memory mapping (map_d in Listing 4.3) which stays constant throughout the whole simulation,(see Figure 4.10 and Figure 4.11) more detailed mappings can be found in Appendix A.5. Unfortunately the fact that shared memory has to be populated and copied back every solver kernel call (for every variable) cannot be avoided.

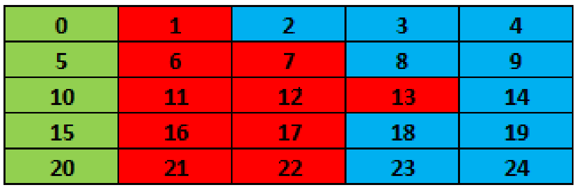
Going back to Listing 4.3 lines 1 to 11 are responsible for the shared memory handling. First the size of shared memory has to be defined (line 1), then the data (u_d) in line 3 plus the halos in line 4 have to be loaded in. Followed by the allocation of two constants 0 and 1, which will be explained further below. The last step is calling a barrier command in line 11 which ensures that loading in the data is finished before the processing of the data starts.

The whole process is illustrated with an example of a 25 (5x5) cell heat transfer problem in Figure 4.9 which was partitioned in 3 domains. Figure 4.9a shows the initial temperature distribution, Figure 4.9b the partitioned and re-numbered case. Tables 4.1 and Table 4.2 show the cell numbers included in the individual domains and the domains including the halo cell.



0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
100	100	100	100	100

(a) Initial temperatures in degree Celsius.



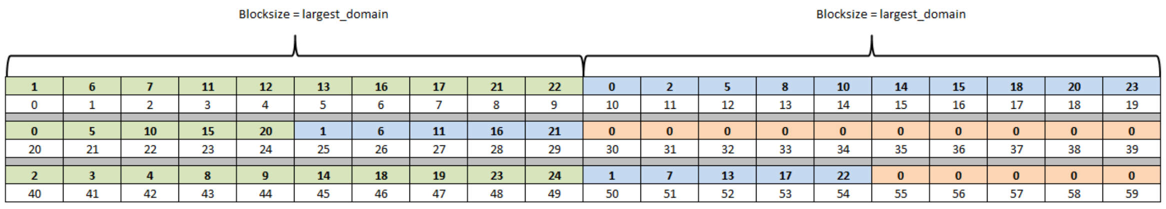
0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

(b) Cell numbering according to the partitioning.

Figure 4.9: Two dimensional mesh partitioning of a 5x5 cell domain.

Table 4.1: Cells excluding halo-cells as in Figure 4.9

Domain 1	1	6	7	11	12	13	16	17	21	22
Domain 2				0	5	10	15	20		
Domain 3	2	3	4	8	9	14	18	19	23	24

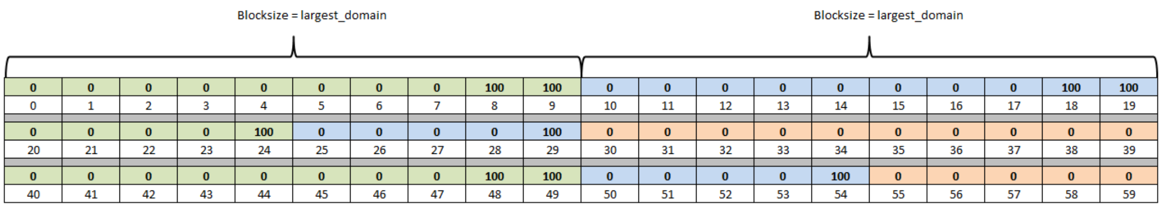


1	6	7	11	12	13	16	17	21	22	0	2	5	8	10	14	15	18	20	23
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	5	10	15	20	1	6	11	16	21	0	0	0	0	0	0	0	0	0	0
20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
2	3	4	8	9	14	18	19	23	24	1	7	13	17	22	0	0	0	0	0
40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59

Figure 4.10: Shared memory cell number and halo mapping.

Table 4.2: Cells including halo-cells as in Figure 4.9

Domain 1	1	6	7	11	12	13	16	17	21	22
(red)	0	2	5	8	10	14	15	18	20	23
Domain 2				0	5	10	15	20		
(green)				1	6	11	16	21		
Domain 3	2	3	4	8	9	14	18	19	23	24
(blue)			1	7	13	17	22			



0	0	0	0	0	0	0	0	100	100	0	0	0	0	0	0	0	0	100	100
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	0	0	100	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0
20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
0	0	0	0	0	0	0	0	100	100	0	0	0	0	100	0	0	0	0	0
40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59

Figure 4.11: Shared memory populated with corresponding temps as in Figure (4.9).

Figure 4.10 and Figure 4.11 show the the cell list for u_d, in cell numbers and in temperature values, which is loaded into shared memory. This way of populating the memory makes renumbering schemes as used by Grandison [167] and McManus [170] unnecessary to implement as the halo values automatically get stored concurrently at the end of the each shared memory block and map on the actual cell values at the beginning of the next block. However renumbering still brings advantages for the CUDA implementation regarding memory access patterns and therefore performance which will be described in the next paragraph.

Looking at Figure 4.12 which uses a thin plate heat conduction problem to demonstrate that using a fixed numbering scheme is limiting the execution performance. The inner three columns consist of boundary (blue) - non boundary (red) - boundary (blue) cells. By calculating cells 4,5,6 in this order the calculation will take a different path (represented by if statements) within the CFD code for each type of cell. Different paths in the code equals longer memory access times due to physical gaps between the storage positions in memory. If this happens in CUDA coalesced memory accesses cannot be guaranteed any more

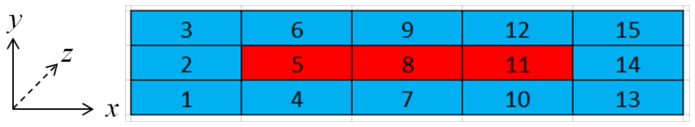


Figure 4.12: 5 x 3 domain no renumbering scheme applied.

which serializes the code. Serializing the code means that parallel execution is no longer possible and these parts of the code will get executed in a for loop fashion element by element. To avoid this the implementation renumbers the geometry as shown in Figure 4.13, this results in an execution flow which first executes all the non boundary cells followed by the boundary cells. The basic idea of the CUDA memory allocations is to group as many similar elements as possible to fully utilise the fact that the smallest unit, a warp, can access 32 elements (of the same kind) which lie back-to-back in memory in one execution cycle.

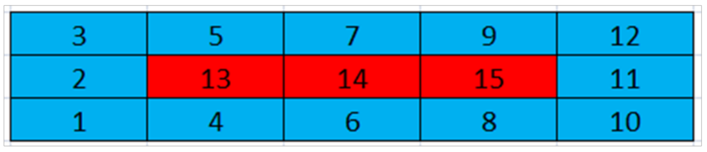
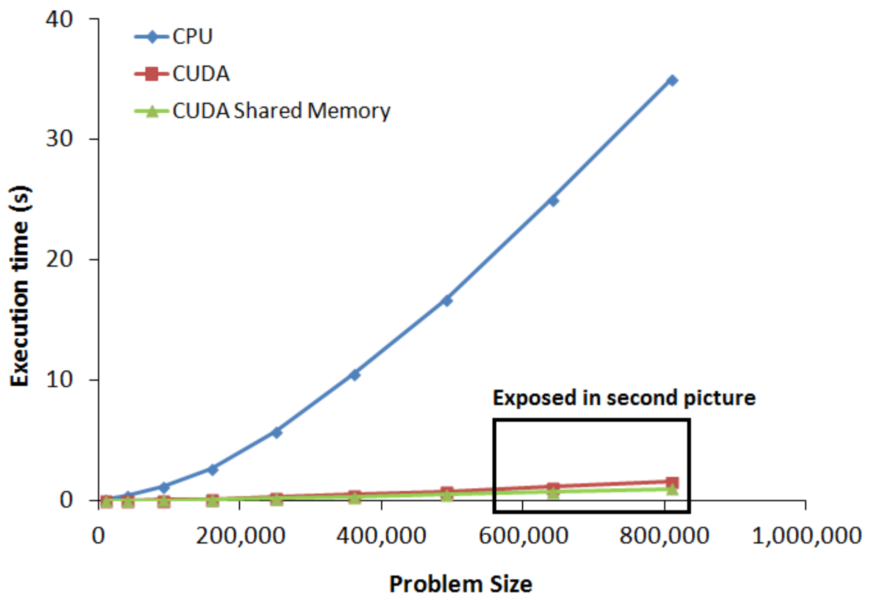


Figure 4.13: 5 x 3 domain renumbering according to boundaries applied.

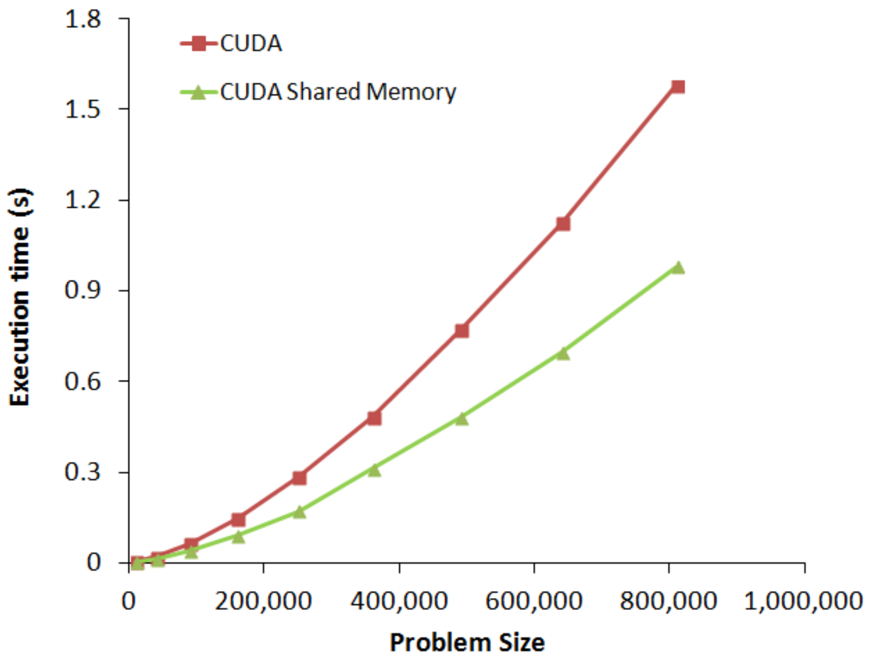
After we have partitioned Domain, identified the halo cells, implemented the solver and taken care of the shared memory, the solution can now be obtained. Figure in Appendix A.4 shows the complete case to Figure 4.10 with individual thread numbering block by block, where the largest domain dictates the size for all domains. Partially filled blocks/ domains get populated with zeros to ensure a unified block/ thread layout (see 1024s and 1025 in Figure in Appendix A.4). As it describes a two dimensional case a maximum number of four neighbours can be assumed. A group of 4 shared memory coordinates plus the number of neighbouring values (ap coefficient in this case). So for example domain 0 first cell is 0 which has the mapping coordinates (11;10;1;1025), which stand for cell numbers (2;0;6;blank). The blank entry is necessary as a consistent offset of max_number_neighbours (5) is needed to access each group of neighbours, therefore a zero has been set in shared memory to eliminate the entry in the summation. The 5th entry is the ap coefficient in the actual implementation of the code.

A two dimensional JOR solver applied to a heat transfer problem (as shown in Figure 4.9) has been benchmarked to demonstrate the difference in shared memory and non shared memory implementations. The results in Figure 4.14a show a speed-up of 17x for the non shared memory implementation compared to the serial version, a 37x to 45x speed-up (depending on problem size) for the shared memory implementation compared to the serial version. Figure 4.14b shows the direct comparison of the (non) shared memory versions, where it can be seen that the shared memory version outperforms the other version by a factor of 2.

The main advantages of the mentioned solver implementation are the fact that it uses a static mapping to access the corresponding values in shared memory. This mapping will not change during the runtime of the simulation. This makes use of the fact that shared memory accesses are a lot quicker than accesses to global memory by applying the same mapping for variable to solve. The second advantage is the fact that the shared memory implementation eliminates the CUDA constraints of being limited by its block_size definition. This means more values can be stored in shared memory even so the identifiers , determined by the block_size, cannot directly access them. This is particularly useful for the present halo concept, as the halo cells do not add any additional benefit to the domain, they are sole used to exchange data with the adjacent domains. The required number of halo cells can quickly add up and occupy a large number of resources per block, that can in turn not be used for actual cell calculations any more. For a 125 (5x5x5) cell domain up to 68% of cells can be halo cells. The loading of the cells that exceed the block_size can be seen in line 4 in Listing 4.3. The layout of the data in memory plus the mapping mentioned previously take care of copying and accessing these values. This concept makes use of the maximum number of resources to calculate its new values during a solver step.



(a) CUDA non-shared memory, CUDA shared memory and CPU solver.



(b) CUDA non-shared memory and CUDA shared memory solver.

Figure 4.14: 2D heat transfer problem solver benchmark.

Chapter 5

TESTCASES CFD

A CFD fire model must fulfil various criteria to be considered accurate. It has to be able to model all the products that occur during a fire as well as keeping the overall execution time at an expectable level. As there will always be a trade off between maximum accuracy and the computational effort necessary to complete the simulation it is important not to over-engineer simulation cases and still stay within expected ranges of result.

In order to validate that the model produces results within an acceptable tolerance level, results have to be benchmarked against verified data sets. Unfortunately the nature of fire sciences/ fire safety engineering makes it difficult to obtain accurate results.

As part of testing the CUDA CFD model, two cases have been simulated and benchmarked. The first case is based on one of the experiments performed by Steckler et al. in 1982 [171] and later in 2001 from Grandison [172]. This case is well suited to validation studies due to the relatively simple nature of the experiment together with a high degree of instrumentation. The other case resembles a two storey care home to demonstrate how the CUDA version performs in more complex scenarios.

5.1 Steckler Room

The geometry and dimensions of the Steckler room case can be seen in Figure 5.1 and consisted of a compartment measuring 2.8m x 2.8m x 2.18m with a centrally located doorway with the dimensions 0.74m x 1.83m. Furthermore it included a heat source (non-spreading fire) in the form of a centrally located methane gas burner. The dimension of the burner were 0.3m in diameter and 62.9kW output. The room consisted of 0.1m thick ceramic fibre insulated boards to establish near steady state conditions within 30 minutes. The original room had moveable bidirectional velocity probes and bare-wire thermocouples within the door to measure velocities and temps in the doorway (see red line, door frame in Figure 5.1).

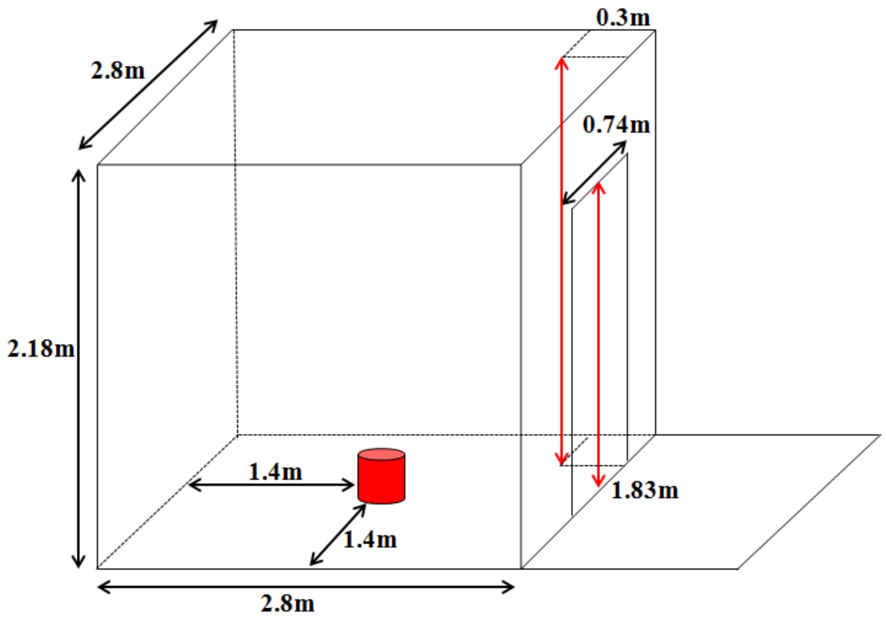


Figure 5.1: Steckler room setup, probes in red [171]

The Steckler room model has become one of the standard fire model benchmark cases, used by a number of field and zone model developers. It is mainly used to test temperature and flow distributions in small compartments subjected to a steady non-spreading fire where several predictions of parameters are made and cross compared [172].

CUDA SMARTFIRE results and serial SMARTFIRE results have been obtained using the same simulation set ups, over a simulation time of 200 seconds. Measurements were taken using a vertical measuring line probe, exactly in the middle of the door width. It can be seen that results (see Figure 5.2 and Figure 5.3) are in good agreement with some minor differences in the outputs. The differences compared to the serial SMARTFIRE version can be explained by the fact that the CUDA model uses a different kind of floating point arithmetic. The differences compared to the experimental results can be explained by the fact that both, the serial and the CUDA SMARTFIRE model had to be run without the combustion model, as the CUDA model currently does not have the ability to model combustion. To further demonstrate the accuracy the CUDA u-velocity and temperature results have been evaluated against Steckler's experimental data and the serial SMARTFIRE results and can be seen in Figure 5.2 and Figure 5.3. It can be seen that the results obtained by the CUDA SMARTFIRE version fall within the range of results produced by the other model.

The main differences in CUDA's floating point arithmetic compared to the serial version are:

- Floating point operations are not associative, so seemingly-benign reorderings (such as the race conditions from multi threading) can change results
- Different architectures support different levels of precision and rounding under different conditions (i.e. compiler flags, control word versus per instruction)
- different compilers interpret the language standards differently
- some architectures support Fused Multiply-Add (FMAD) and some do not

GPUs have a much better-designed architecture for floating point arithmetic than any contemporary CPU. GPUs include native IEEE standard support for 16-bit floats and FMAD, have full-speed support for denormals, and enable rounding control on a per-instruction basis rather than control words whose settings have side effects on all floating point instructions and are expensive to change [173].

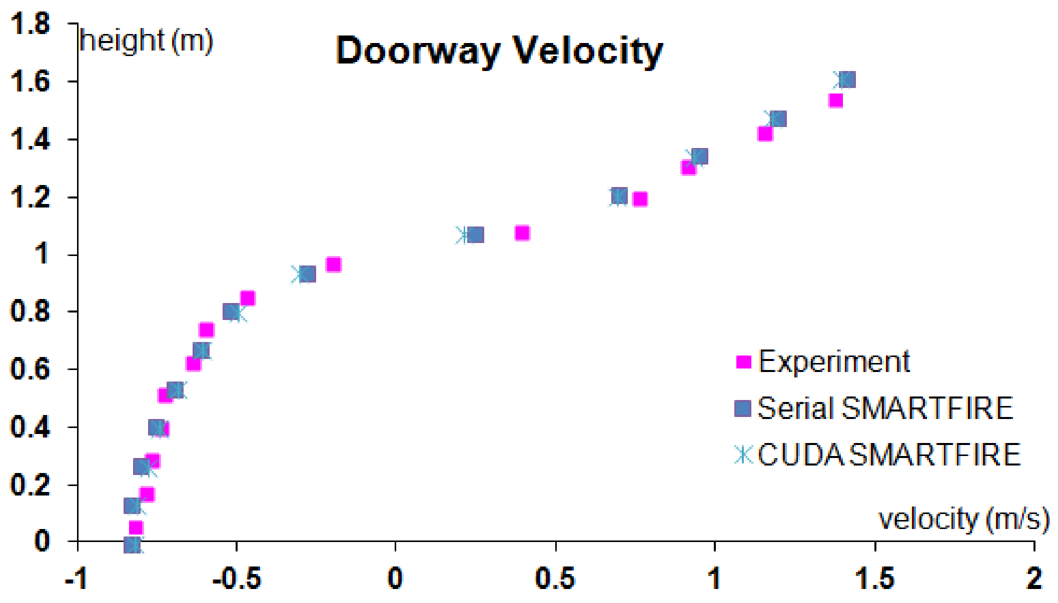


Figure 5.2: Steckler room simulation u-velocity results for various models.

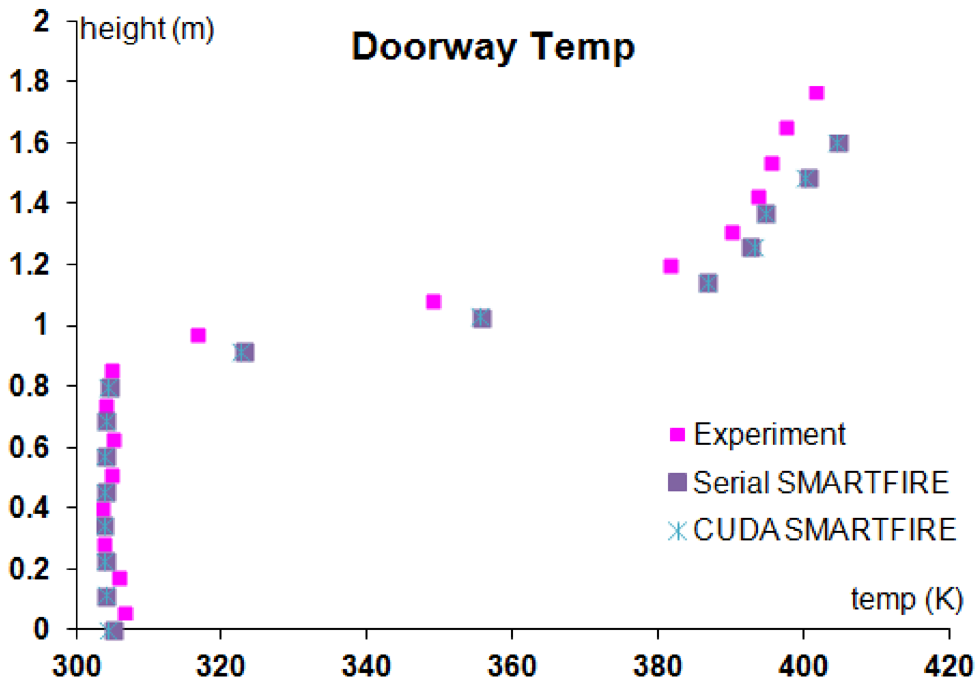


Figure 5.3: Steckler room simulation temperature results for various models.

In addition to the benchmarking, Steckler's room was used to create further scenarios to demonstrate the effects of dynamically changing the environment. The original Steckler room was adapted by adding a door to close the room from its surrounding area. To demonstrate the door capabilities, the following 4 scenarios have been simulated:

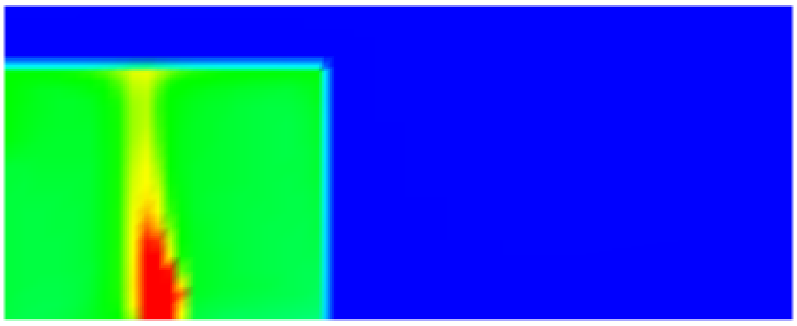
- Door always closed
- Door always open
- Door opening after 50 seconds
- Door opening after 90 seconds

Figures 5.4 shows these 4 scenarios where it can be seen that the further into the simulation the change happens the bigger the effect on the temperature distribution becomes. In addition to the benchmarking, performance timings of the parallel code compared to the serial have been taken, see Table 5.1.

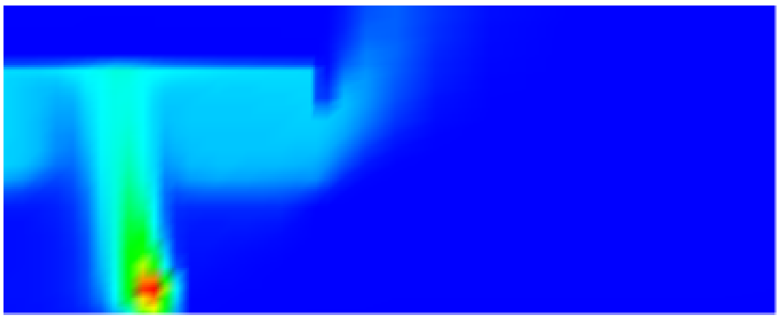
Table 5.1: CUDA execution speeds vs serial execution speeds

Number of cells	CUDA (t in s)	Serial (t in s)	Speed Up
18K	390	4806	12.32x
47K	961.3	14721	15.31x

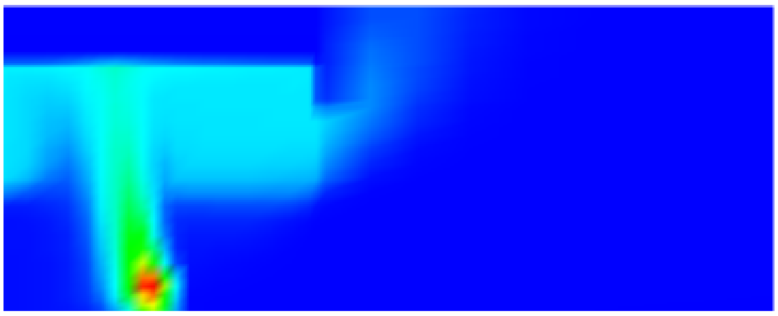
It can be seen that the speed up increases when the problem size increases. More details about expected speed ups and measurement results from much larger cases will be presented in the next Section 5.2. In general the performance of the CUDA code will go up until all the available hardware resources are in use simultaneously.



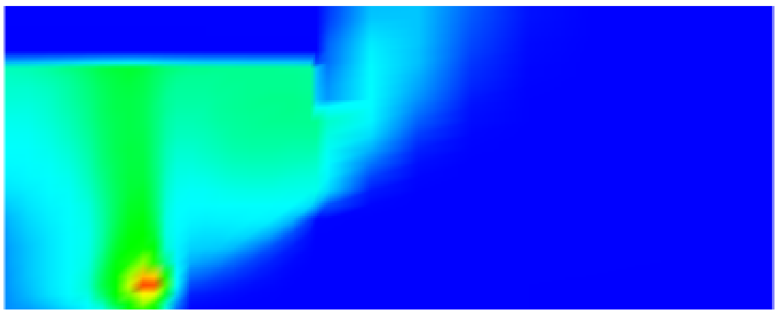
(a) Steckler room simulation door closed for 100 seconds.



(b) Steckler room simulation door open for 100 seconds.



(c) Steckler room simulation door opens after 50 seconds.



(d) Steckler room simulation door opens after 90 seconds.

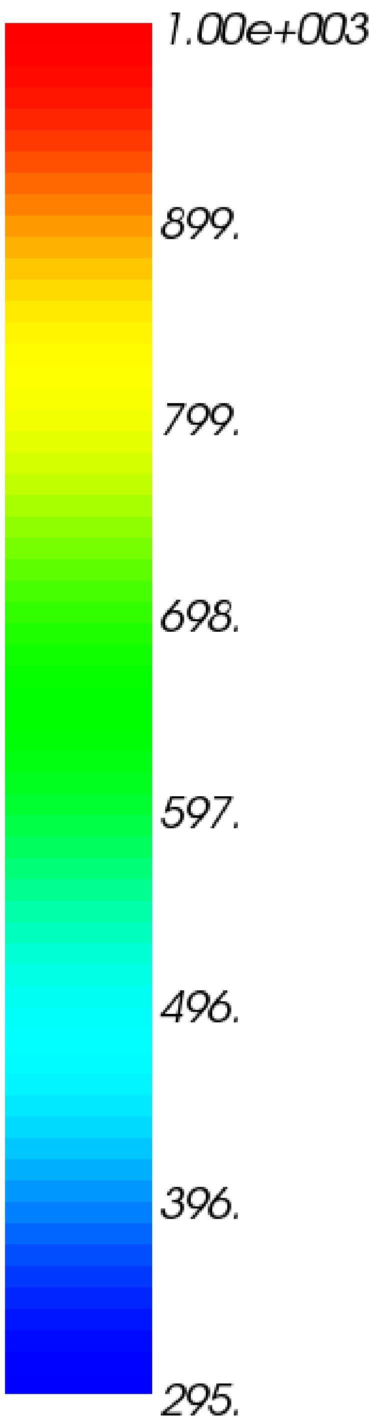


Figure 5.4: Cut plane view of the temperature distribution for different opening times of the door.

5.2 Care Facility

The second code comparison case, represents a two-storey day care facility. The geometry consists of 2 floors, 2 staircases, one main exit and three emergency exits on the ground floor. The building includes several nursing rooms, communal areas and offices for staff members. The fire is located in the left staircase, starting from the ground floor, see red dot in Figure 5.5.

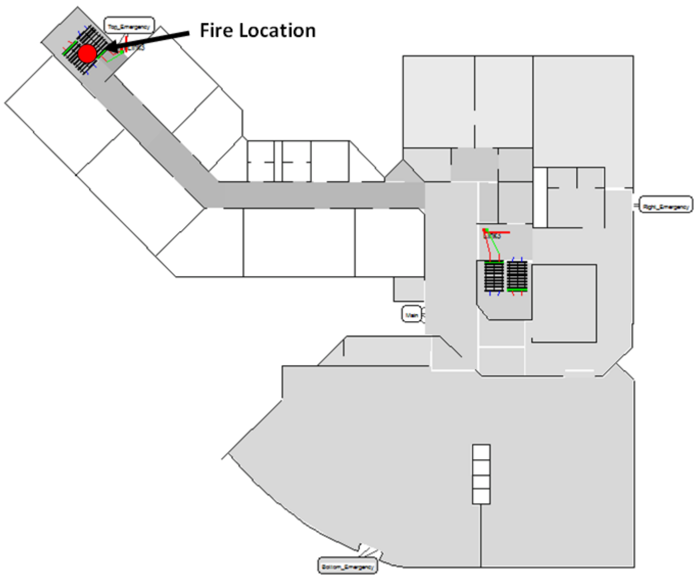


Figure 5.5: Day care facility, identical layout for both floors.

The fire has its origin in the left staircase, causing smoke and temperature to spread through the rest of the ground floor and via the staircases to the first floor. Some rooms are fitted with fire doors, therefore fire products do not spread within those areas.

Staircase details:

- Style: Dog Leg i.e. Two runs with landing
- Width: 1.2m
- Height: 3.0m combined (1.5m for each leg)
- Length: 1.89m
- Lanes: 2
- Riser number: 10 (i.e. 9 treads)
- Hand rail: enabled with 0.1m handrail diameter
- Land width: 0.5m

Population details:

- 100 occupants on ground floor
- 50 occupants on first floor
- 2 staff per floor

Fire specifications:

- Total source 500KW
- Smoke total source 0.01 kg/s
- Location on ground floor in the left staircase
- The fire volume for fuel release is 1.2m x 1.2m x 1.2m
- Radiation model enabled (multi-ray)

Further geometry specifications:

- Building dimensions are 48m x 36.4m x 6.2m

- Considering the structure ventilation, the door immediately adjacent to the fire is assumed to be closed throughout the simulation beside little gaps above and below the door. The main entrance door is also assumed to be closed, but the left- and right-emergency doors, on the ground floor, are open throughout the simulation. There are no open external doors or windows on the upper floor.
- Heat conduction through solid walls is not enabled. This is a conservative approximation as all heat will be available for the occupants to experience, within the building environment.
- All upper floor rooms are assumed to be open to internal corridors, however the lower long corridor (towards the fire) is assumed to have rooms with their doors closed.
- The geometry is partitioned into 93 separate Hazard Sub-Volumes for exporting the fire and toxicity environment

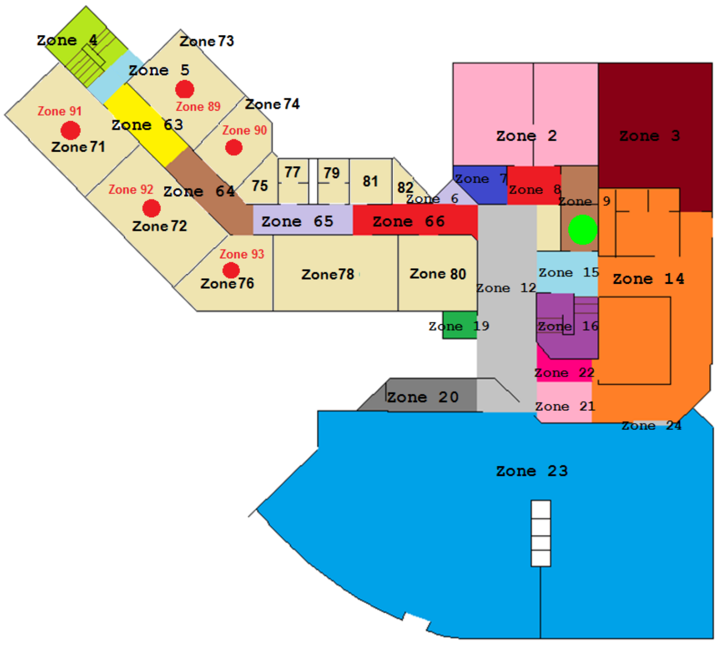


Figure 5.6: Smoke zone layout, ground floor.

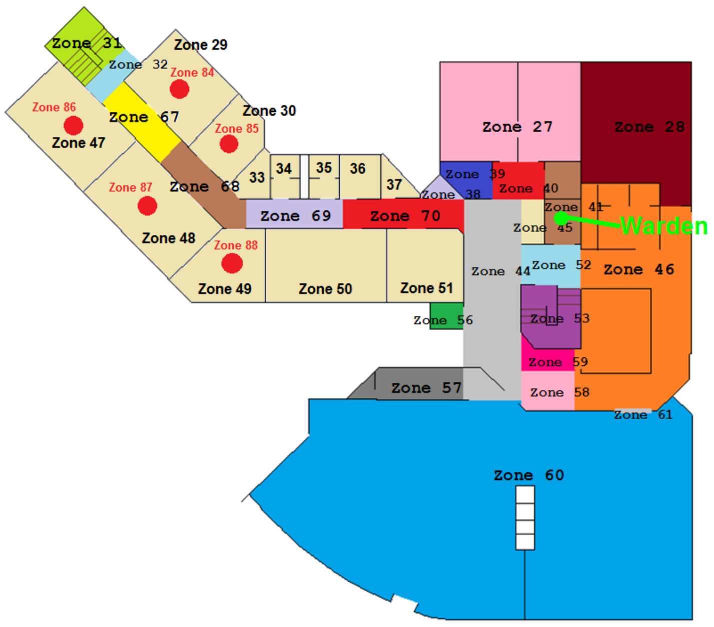


Figure 5.7: Smoke zone layout, first floor.

The benchmark case uses the same geometry as shown in Figure 5.5, where a measuring probe has been placed in the corridor on floor one in y direction (floor to ceiling). Figures 5.8 to Figure 5.13 show the measuring probe height, in equal sub units (x axis) vs. measured quantity (y axis).

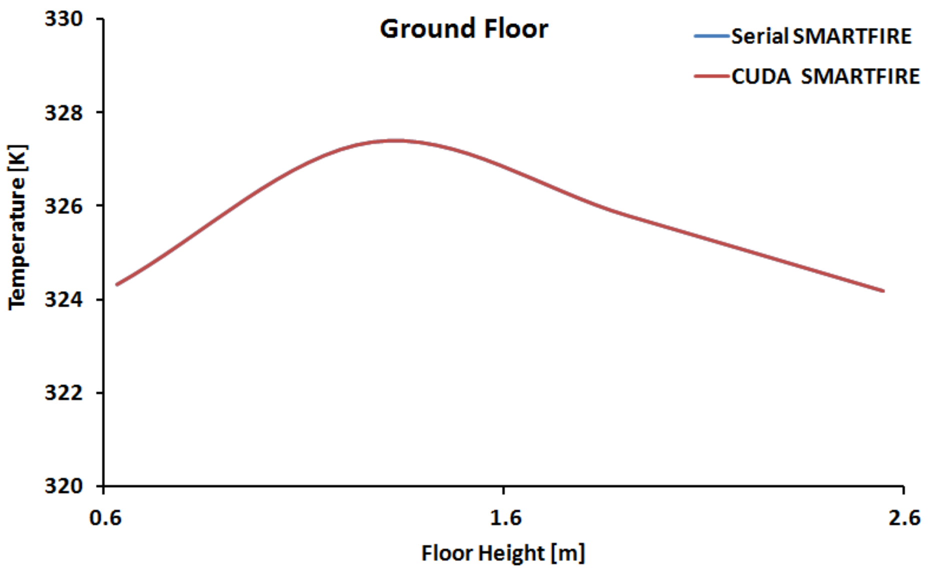


Figure 5.8: Serial and CUDA temperatures vs. height, Ground Floor.

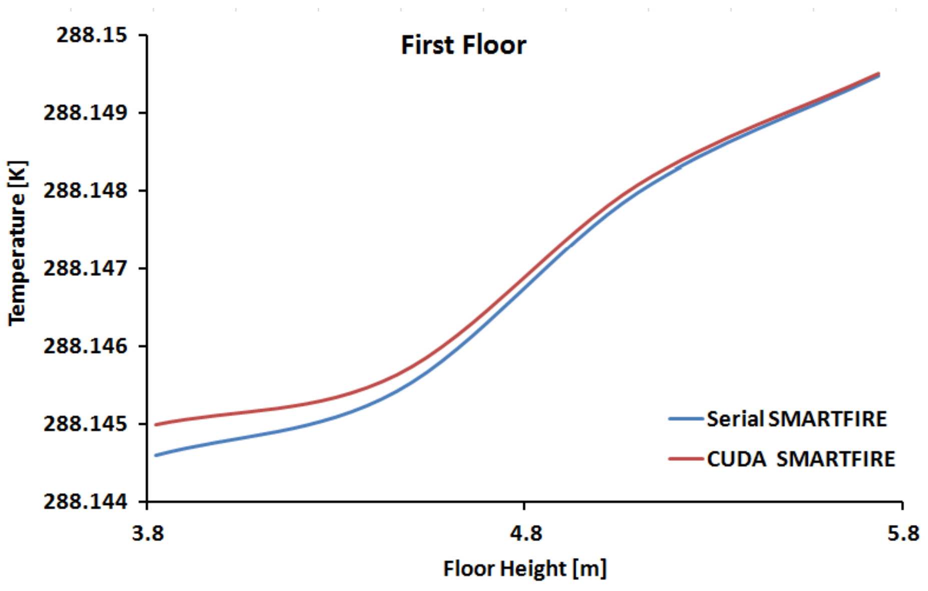


Figure 5.9: Serial and CUDA temperatures vs. height, First Floor.

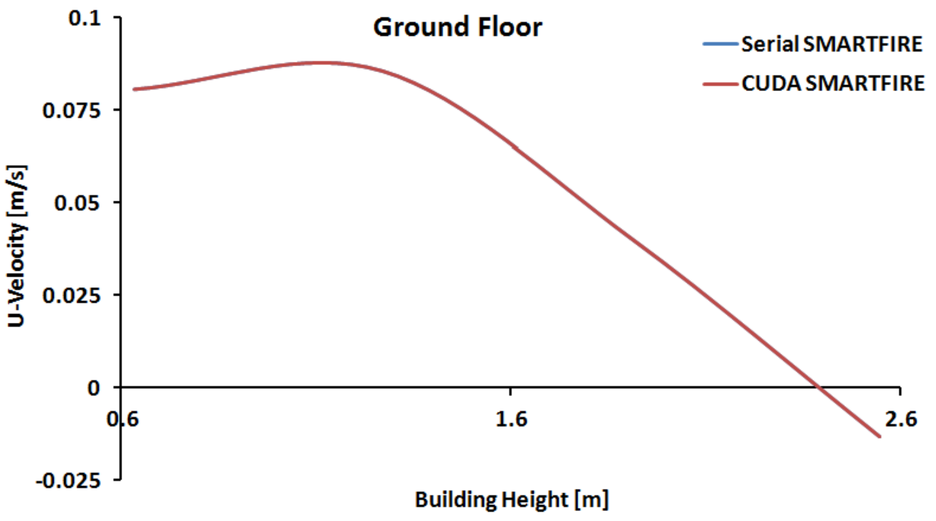


Figure 5.10: Serial and CUDA velocity vs. height, Ground Floor.

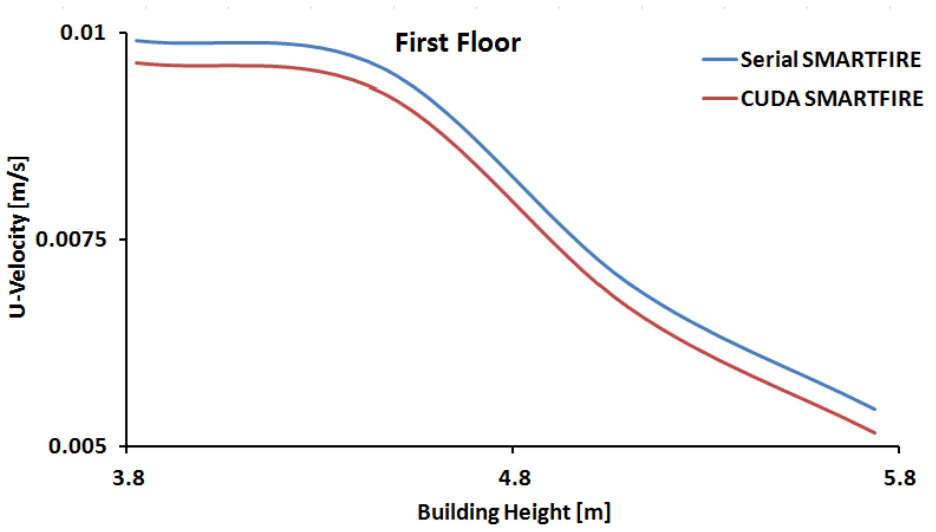


Figure 5.11: Serial and CUDA velocity vs. height, First Floor.

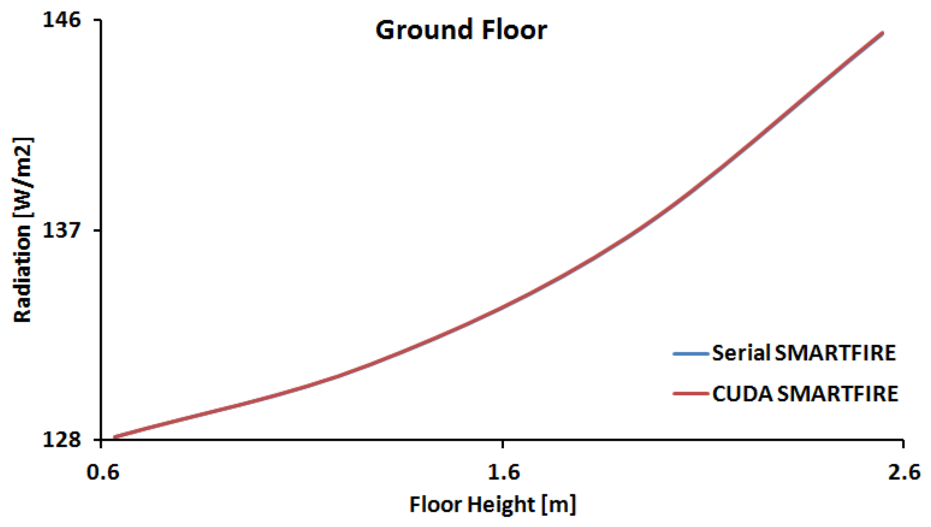


Figure 5.12: Serial and CUDA radiation vs. height, Ground Floor.

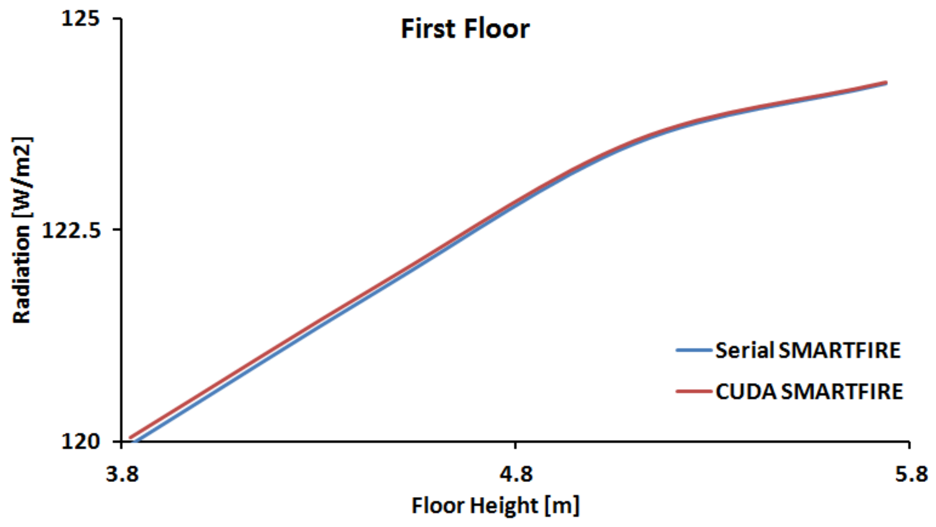


Figure 5.13: Serial and CUDA radiation vs. height, First Floor.

It can be seen in Figures 5.8 to 5.13 that the values calculated by the CUDA version match the serial results with very little differences.

The results in Table 5.2 show the full scale simulation runs for the day care facility, all cases were applied to the same problem and were using a heat model (including 24 ray radiation), flow model, smoke model and turbulence model. The mesh size has been varied for each

Table 5.2: CUDA execution speeds vs serial execution speeds

Case #	Number of Cells	Serial (t in s)	CUDA (t in s)	Speed Up
1	317208	467.00	22.50	20.75x
2	349676	517.00	24.58	21.03x
3	371420	553.00	26.36	20.98x
4	543376	844.00	38.87	21.71x

case, from 317208 in case 1 to 543376 cells in case 4. It can be seen that the CUDA model performs much better compared to the previous benchmark case in Table 5.1 in terms of speed ups. As mentioned before in order to run efficiently, the GPU needs a large workload to exploit its strength. Figure 5.14 shows the performance graphs (speed up vs. problem size), with full optimisation switched on and compiled up in release mode plus the theoretical possible speed up. The red line shows the theoretical hardware limit of the GPU where theoretically 100% of the available resources are occupied and used for calculations. After that point data has to be added to a queue until resources are available again. Theoretically the green line should be a straight horizontal line after this point, as no more speed up can be achieved, but as full hardware occupancy is really hard to achieve and highly case and implementation depended (branching, shared memory limits, synchronisations, etc.) this point can get shifted to the right. In the present care facility case an occupancy of 92% has been achieved. But it can clearly be seen that it follows the expected trend and that smaller problems will result in smaller speed ups as not all resources will be used to calculate results. The theoretical speed up scales linear up to the point where no additional hardware resources are available anymore and the speed up stays constant afterwards. The CUDA timing procedure is very similar to the common approach for general timing purposes, where dynamic features (file savings and outputs in parallel implementations) get disabled. These dynamic features can totally distort the actual performance and only highlight the bottlenecks, as Grandison described in [167].

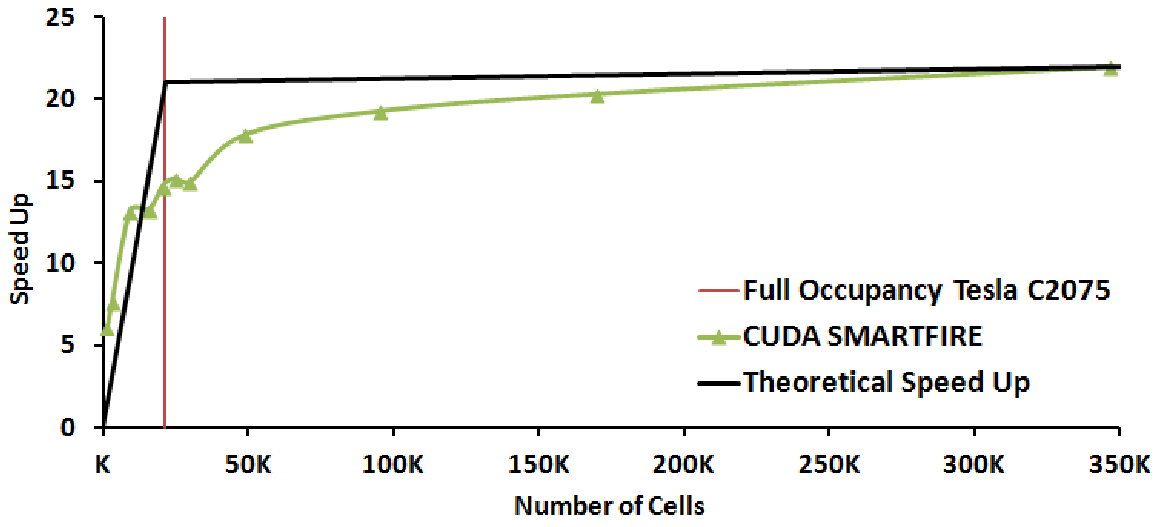
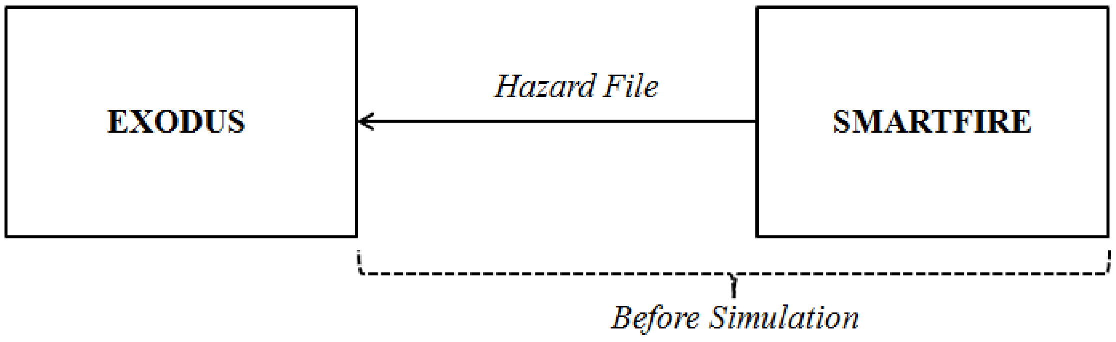


Figure 5.14: CUDA speed up vs problem size.

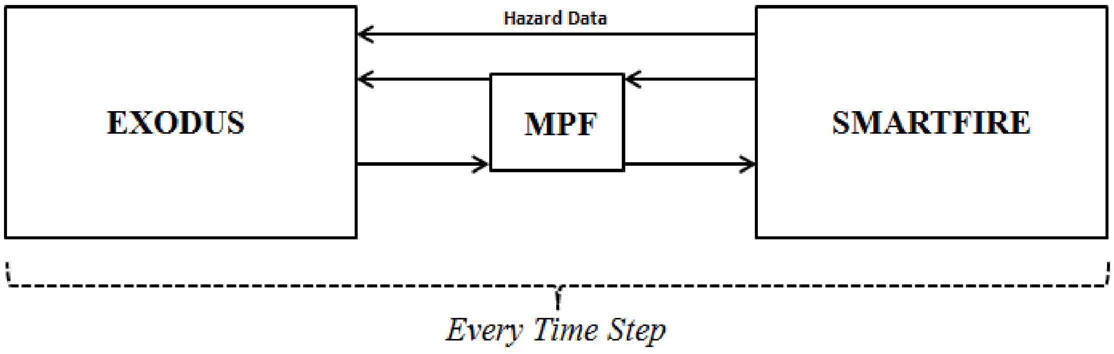
Chapter 6

MODEL COUPLING

The definition of a coupling is the degree of independence between two pieces of software, a measurement of how closely together two work flows interact with each other. In this thesis the coupling of the two software tools, SMARTFIRE and buildingEXODUS, described in Section 2.2 will be described. On a high level description the coupling enables buildingEXODUS to control features in SMARTFIRE and vice versa. Figure 6.1 shows a simplified illustration of the previously used one-way coupling, where a pre-calculated CFD files gets loaded into EXODUS prior to the simulation, compared to the newly developed two-way coupling where the CFD files are updated every time step. This feature can be used to give agents in buildingEXODUS the capability to change the geometry and therefore the CFD results. Geometry changes can be the opening/ closing of doors, smashed windows, structural changes, collapsing walls etc. The CUDA version of SMARTFIRE will focus on dynamic door changes triggered by agents in the buildingEXODUS model. The interaction is established by connecting the two models and passing control messages between them. To capture every event the synchronisation will be performed every CFD time step, in order to incorporate every geometry change in the next upcoming time step. One example of a single time step process flow is e.g an agent has a task to open Door A at his arrival, at the times step of his arrival he changes the the door status in EXODUS to ”open” and simultaneously sends the update command to SMARTFIRE. The control message gets picked up by SMARTFIRE and gets executed i.e the boundary condition for Door A gets updated, and the next time step values get calculated using the updated conditions. The final step is to send the newly calculated end results back to EXODUS. This process can repeat itself numerous times during a simulation run and can include multiple events at the same time as well.



(a) One way coupling, hazard file gets passed in before simulation.



(b) Two way coupling, Message Passing File (MPF) constantly exchanges control instructions and hazard data between the models.

Figure 6.1: Simplified illustration of One-way and Two-way coupling of EXODUS and SMARTFIRE.

Figure 6.2 gives an overview how the different EXODUS and SMARTFIRE modules, discussed in Section 2.2 (see Figure 2.2 and Figure 2.1) interact, it can be seen how the connection between the SMARTFIRE CFD engine and the EXODUS HAZARD and ENCLOSURE models are established. Where the ENCLOSURE model communicates the actions performed by the agents on the geometry back to the SMARTFIRE model which then updates the HAZARD sub-model data in EXODUS. A detailed overview of the synchronisation model can be seen in Figure 6.3 which shows the different stages of the two models during a coupled simulation run.

It can be seen that both models reside in suspended states after their time step operations complete. Unfortunately these suspended states cannot be avoided as EXODUS and SMARTFIRE both have very different execution times. EXODUS takes seconds to minutes to finish while SMARTFIRE can take minutes to hours to complete, depending

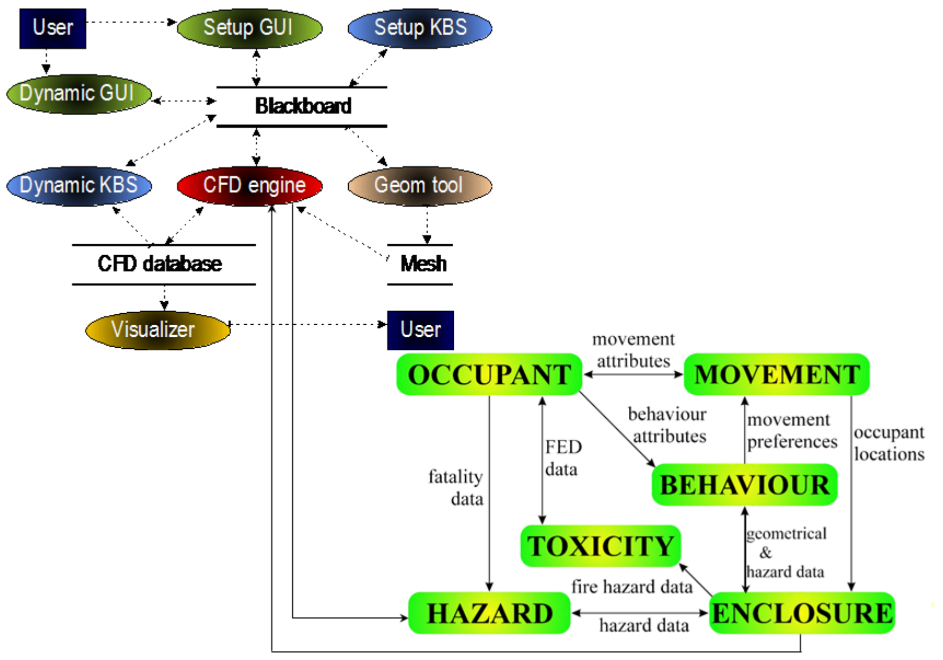


Figure 6.2: Detailed illustration of Two-way coupling of EXODUS and SMARTFIRE.

on the case simulated. In detail this means that the models are time step locked, while one tool calculates its current time step the other waits idle for a status message to start its next time step calculation. Figure 6.3 shows these idle stages in (2) and (5) the two status messages required to continue are *EXMS_SMF_ESF_READY* for SMARTFIRE and *SMF_EX_DAT_READY* for EXODUS respectively. The different control messages can be seen in Table 6.1. The synchronisation system is established by using a binary file to store the status messages from both tools. During each step the file will be blocked by the model currently using it, to ensure that no messages get lost. The execution order of the two

Table 6.1: Coupling control status messages.

EXMS_SMF_CONNECTED	Connection to .esf file established
EXMS_SMF_ESF_READY	EXODUS finished writing .esf file
SMF_EX_DAT_READY	SMARTFIRE finished writing .dat file

models is predetermined, as the CFD data has to be available to initialise EXODUS first. This can be seen in Figure 6.4, SMARTFIRE is started first, checks for the existence of the synchronisation (*.esf*) file and connects to it, next the data output file (*.dat*)is created and header entries are written, describing the hazard zone configuration and the variables used to calculate the upper and lower layer per zone. Next SMARTFIRE sets the status message to *EXMS_SMF_CONNECTED*, followed by a message containing the total CFD simulation time and the file name of the data output file (*.dat*). SMARTFIRE then performs the calculation for the initial entries of the output file. After the output file is ready, EXODUS can initialise the geometry and run its first time step. The highlighted area in blue in Figure 6.4 represents the time step calculation loop in SMARTFIRE which repeats itself from *n=1...n=total_simulation_time*.

The biggest strength of the two-way coupling is the ability to eliminate the necessitate to pre-create CFD cases based on assumptions when events might have occurred, meaning the two-way couple gives the agents the ability to execute tasks as they reach target locations. This means the agents do not have a pre-set input, when to perform a task e.g open a door. The time the tasks get executed purely depends on environmental factors present in the evacuation simulation, e.g temperature, smoke and radiative heat. These environmental factors control the time it takes the agent to reach its targets.

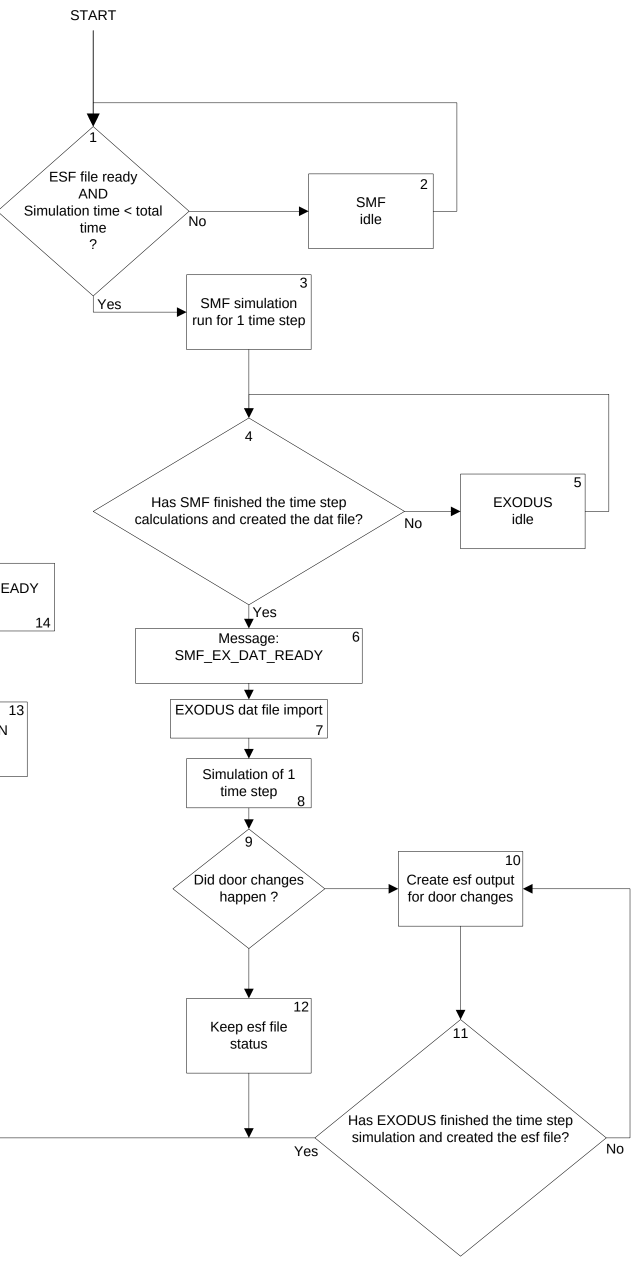

START
1
ESF file ready AND Simulation time < total time ?
No
2
SMF idle
Yes
3
SMF simulation run for 1 time step
4
Has SMF finished the time step calculations and created the dat file?
No
5
EXODUS idle
Yes
Message: SMF_EX_DAT_READY
6
EXODUS dat file import
7
Simulation of 1 time step
8
9
Did door changes happen ?
10
Create esf output for door changes
11
Has EXODUS finished the time step simulation and created the esf file?
Yes
No
12
Keep esf file status
13
SIMULATION TIME +1
Message: EX_SMF_ESF_READY
14

Figure 6.3: Communication flow between SMARTFIRE and buidlingEXODUS during coupling.

By nature, EXODUS is a stochastic model where behaviours and movements are determined probabilistically, in order to get statistically significant results, a sufficient, preferably large number of simulation runs (with different starting positions for the agents) have to be performed per scenario (see Section 2.2.2). Each change that happens during a simulation run creates a new sub-scenario. In order to stay statistically significant, a large number of simulation runs have to be performed for all permutations of the sub-scenarios (e.g changes in the populations properties, e.g starting locations). The cases discussed in this thesis, re-used the same population for each scenario as the focus is on the capabilities of the coupling, which made the results obtained deterministic. This reaches a point very quickly, where it is simply too time consuming or just impractical to pre-calculate that many different CFD sub-scenarios. This shows the other big advantage of the two-way coupling, which eliminates this necessity by constantly exchanging information to adjust to changing environments, which means there is no need any more to manually go through all the permutations.

The two-way coupling offers several novel capabilities and eliminates possible causes of problems in the modelling process. It is for example almost impossible to know a priori how the fire and evacuation will interact which makes it difficult to set up scenarios using pre-defined parameters. The two-way coupling eliminates some of these assumptions previously required to model scenario when a one way coupling was used e.g. doors staying open or closed at fixed times, these times will be determined during the simulation by the environment and the interaction of the agents with the environment. This allows FSEs the variation of the evacuation to effect the course of the fire scenario. At the time when this research was carried out this complete two-way coupling of a CFD code and an evacuation simulation hasn't been done before.
The two-way coupling will hopefully establish itself in the FSE community and make a valuable contribution to CFD fire and evacuation modelling and eventually safe lives.

The next chapter will describe different example coupling cases and highlight the differences compared to non-coupled executions.

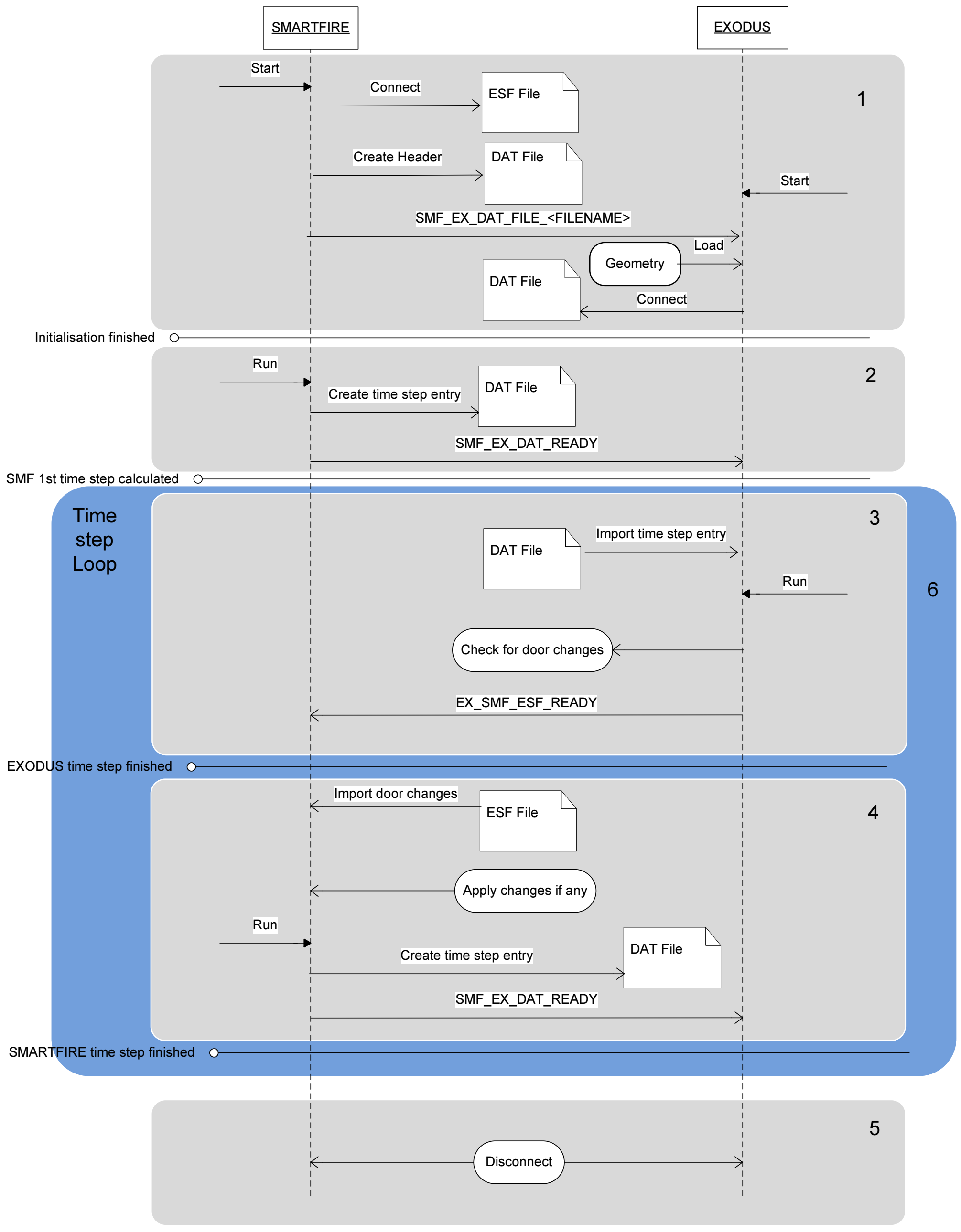


Figure 6.4: Sequential diagram between SMARTFIRE and buidlingEXODUS during coupling.

Chapter 7

TESTCASES COUPLING

In this chapter the capabilities of the two-way coupled model will be demonstrated. SMARTFIRE and buidlingEXODUS will be used on two scenarios, a simplified geometry consisting of three rooms, connected by a corridor and the more complex care facility scenario already presented in Section 5.2. The simplified scenario will be used to highlight the general concept of the coupling and demonstrate the interactions of the agents with the geometry plus the differences in evacuation outcomes this leads to. The care facility case will give an overview of how the coupling performs in complex environments, with one or more agents performing tasks.

7.1 Coupling Case 1: Simplified Geometry

Coupling case one consists of a very simplified layout and hazard set up. Figure 7.1 shows a geometry with three rooms, two doors and a total population of 10 people plus one warden. The three agents, Agent A in Figure 7.1 green circle, Agent B, blue circle and a warden, red circle . A fire is located in the room on the right (red square) which is initially connected to the corridor by an open door. The warden has the task to inform the people in both rooms and lead them to the main exit in the bottom right hand corner. The door to the room containing the fire is labelled Door_1 the door to the rooms with the ten people as Door_2.

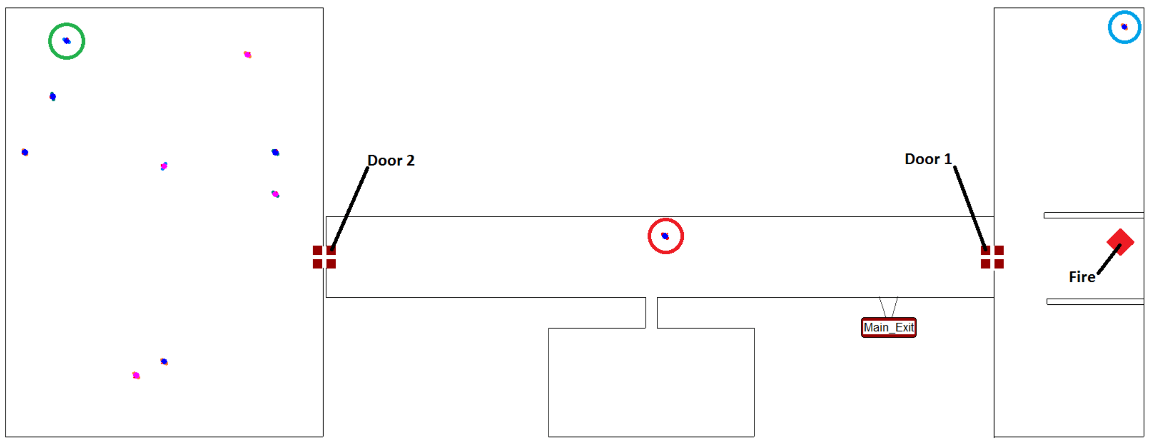


Figure 7.1: Geometry and population set up for test case 1.

Three scenarios were simulated using this simplified geometry:

1. Warden checks the room on the right for people, closes Door_1, opens Door_2, informs the people to start evacuating and guides them to the main exit where all of them exit the building.

2. Warden checks the room on the right for people, but does not close Door_1 and goes directly to open Door_2 to get the people to guide them to the main exit where all of them exit the building.

3. Warden goes directly to the room on the left, opens Door_2, informs the people and guides them towards the main exit before he checks the room on the right before he closes Door_1.

The outcomes of these scenarios are evaluated by comparing the narcotic properties for specific agents and by comparing the overall evacuation times required. The different kind of exposures an agent faces, have been described by Galea et al. [174]. The ones important for this thesis are:

- FIH: FED (fractional effective dose) model measuring the individuals cumulative exposure to radiative and convective heat.

- FIHc: FED model measuring an individuals cumulative exposure to convective heat.

- FIHr: FED model measuring an individuals cumulative exposure to radiative heat

Coupled versus Non-Coupled results

As previously mentioned the strength of the coupled version is that a fire safety engineer does not have to make as many assumptions and can safe time. In the classic approach, by using the one way coupled version of Exodus and SMARTFIRE, from point a to point b (point a being a starting point and point b being a point of interest, e.g a door, a person's location , etc.). In the previously discussed test cases point a would be the starting position of the warden and point b one of the two doors as they will have an effect on the CFD outputs. After obtaining times t1 and t2, for the times it takes the warden to reach door_1 and door_2 respectively the static CFD fire case will be simulated in SMARTFIRE, using door event times t1 and t2. Afterwards the results of the fire simulation will be used to simulate the Exodus case to obtain individual exit times and narcotic properties. The fundamental problem with this approach is that one has to set event times purely based on distance and static travel speed (RSET/ASET). Which represents a difficult problem as environmental factors that might delay the agents event times t1 and t2 have to be an assumption. The coupled version does not rely on any predefined settings or assumptions, it constantly updates the environmental influenced attributes in Exodus, adjusting health and speed levels. The result is a more realistic representation of individual times, overall time and health statistics for each agent.

A more accurate, non-coupled simulation could be achieved by manually feeding back EXODUS times into SMARTFIRE. Figure 7.2 shows the simple case where a door gets opened during the simulation run. An initial CFD hazard case will be created, EXODUS uses this data to determine the arrival time of an agent at the door to be changed, this time will be used to create a second CFD hazard file which includes the geometry change and finally this CFD hazard data can be used to run the rest of the simulation. This process can be extended for more than one event but the modelling work becomes quite time consuming pretty quickly.

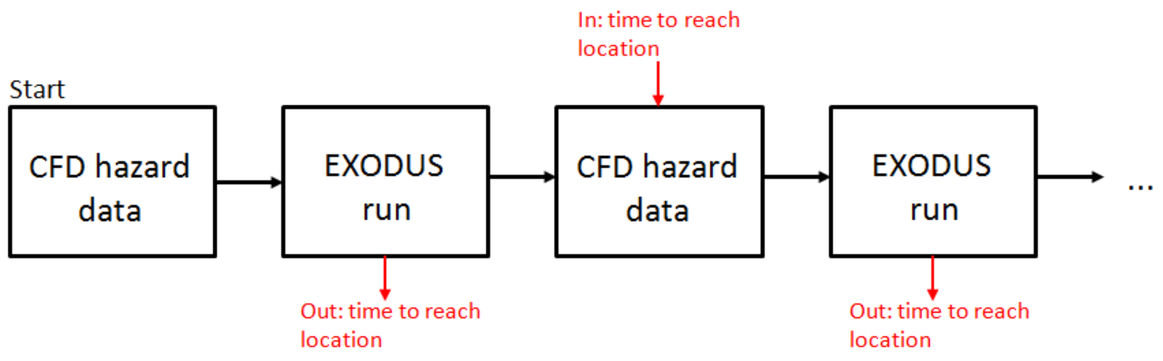

In: time to reach location
Start
CFD hazard data
EXODUS run
CFD hazard data
EXODUS run
...
Out: time to reach location
Out: time to reach location

Figure 7.2: Manuel execution process for two-coupling.

The number of cases that have to be simulated for more complex scenarios and the cumulative errors caused by introducing more assumptions per CFD stage, quickly makes it an impracticable approach.

The following results in Table 7.1 and 7.2 show the comparison of the results for scenario 1 with a 60s delay for the warden, coupled and non-coupled (with action times derived from a basic EXODUS run). The fire strength has been slightly lowered (700KW to 500KW) and the smoke yield has been slightly upped (0.000125 to 0.01 Kg/s) to clearly show the effect the environment has on the agents travel speeds without instantly killing them. Table 7.3 shows the differences in action times, exit times and total evacuation time. It can be seen that action times for door_1 and door_ 2 differ by 72 seconds from 81 seconds non-coupled to 153 seconds coupled and by 129 seconds from 101 seconds non-coupled to 230 seconds coupled. This is caused by the effect the smoke has on the agents travel speeds, furthermore in the non-coupled version door_1 closes 72 seconds earlier, increasing the concentration of radiation and heat inside the room

Table 7.1: Narcotic properties for Agent A, Agent B and the warden, Scenario 1 - 60 seconds delay non-coupled.

Agent	FIN	FIH	FIHc	FIHr
Agent A	0.00124	0.0	0.0	0.0
Agent B	0.00040745	0.0	0.0	0.0
Warden	0.00118	0.93988	0.04525	0.89463

Table 7.2: Narcotic properties for Agent A, Agent B and the warden, Scenario 1 - 60 seconds delay coupled.

Agent	FIN	FIH	FIHc	FIHr
Agent A	0.00145	0.0	0.0	0.0
Agent B	0.00040745	0.0	0.0	0.0
Warden	0.00138	0.01105	0.01105	0.0

Table 7.3: Action, exit and total evacuation times for scenario 1.

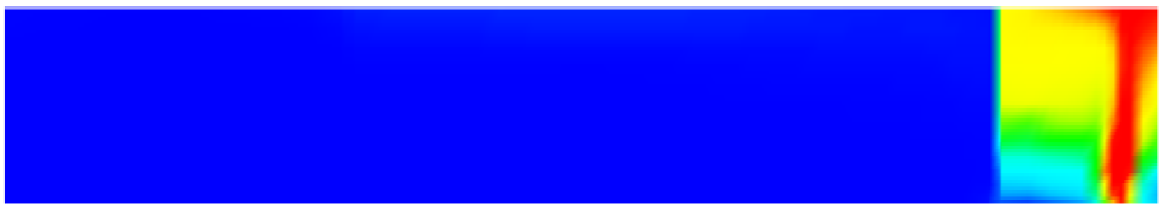
	Non-coupled	Coupled
Action time door 1	81s	153s
Action time door 2	101s	230s
Exit time Agent A	266s	311s
Exit time Agent B	88s	88s
Exit time warden	256s	296s
Total evacuation time	285s	329s

on the right a lot. The assumed pre-calculated closing time of door_1 did not take into account that due to the presence of smoke it will take the warden much longer to reach the door before he can close it. As EXODUS in the non-coupled version does not feed back the slower pace to the CFD results, door_1 closes purely based on the pre-calculated time, while the warden is still in the room, causing the conditions to be a lot worse.

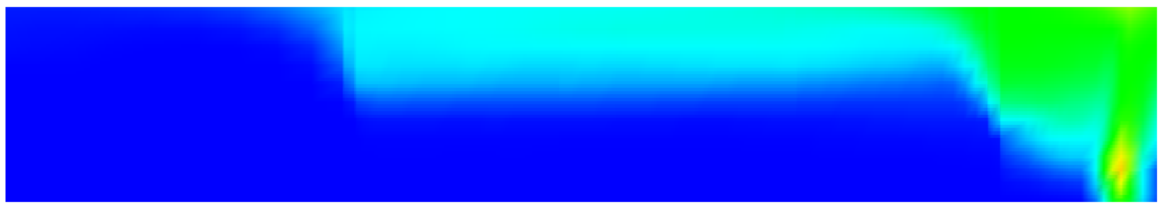
This can bee seen in the wardens FIH readings in Table 7.1 and 7.2, a reading of 0.93988 (1.0 being the fatal dose) means he would have been nearly incapacitated in the non-coupled version. Furthermore it can be seen that the smoke increases individual exit times and the overall evacuation time by 15% (44s). The delay in the closing of door_1 allowed a lot more smoke to fill up the corridor and also effecting the populations travel speeds.

Further Scenarios

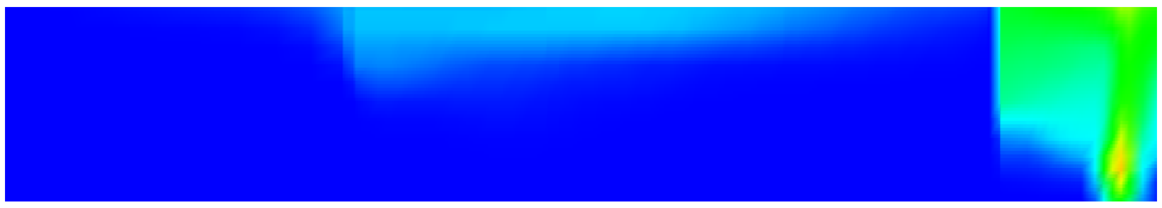
In addition to the under or over-estimation of factors as described in the coupled versus non-coupled comparison, this section will focus on the variety of sub-scenarios that can be directly simulated to show how the coupling can be also used to determine the range of outcomes for a given geometry. It can be seen in Figure 7.3, that the closing of door_1 has a large effect on the environment and therefore the outcomes. The closed door_1 prevents the temperature, smoke and radiation to enter the corridor. Table 7.4, 7.5 and 7.6 show the properties of three agents, Agent A , Agent B and the warden.



(a) Scenario 1 with closed Door 1.



(b) Scenario 2 with open Door 1.



(c) Scenario 3 with closed Door 1.

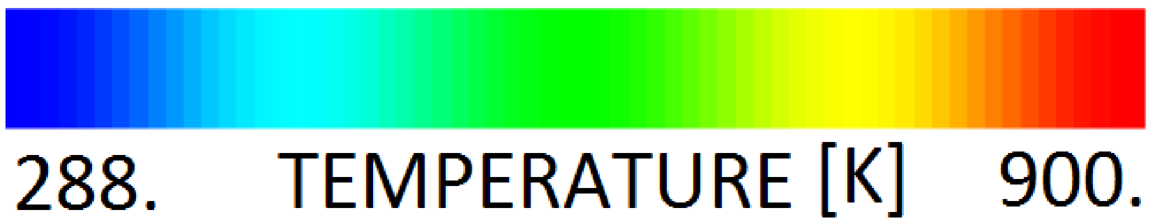


Figure 7.3: Cut plane temperature distribution inside the middle of the corridor for scenario one to three, 47 seconds into simulation.

The outcomes in Table 7.4 and Table 7.5 for Agent B, who was located in the upper right corner of the geometry, show that this agent is not effected by the door change in scenario 1 and 2 as he goes straight down towards the main exit after the door was closed. Meaning he faces the same situation in both scenarios and is only exposed to a different environment during the short time it takes him to travel from door_1 to the exit. In contrast to in scenario

Table 7.4: Narcotic properties for Agent A, Agent B and the warden, Scenario 1, coupled.

Agent	FIN	FIH	FIHc	FIHr
Agent A	0.00038022	0.0	0.0	0.0
Agent B	0.00014318	0.06364	0.06364	0.0
Warden	0.00026514	0.05156	0.05156	0.0

Table 7.5: Narcotic properties for Agent A, Agent B and the warden, Scenario 2, coupled.

Agent	FIN	FIH	FIHc	FIHr
Agent A	0.00037717	0.03867	0.03867	0.0
Agent B	0.00014443	0.06364	0.06364	0.0
Warden	0.00025738	0.06020	0.06020	0.0

3 where he gets notified at last and had a much longer exposure time to a much more developed fire environment.

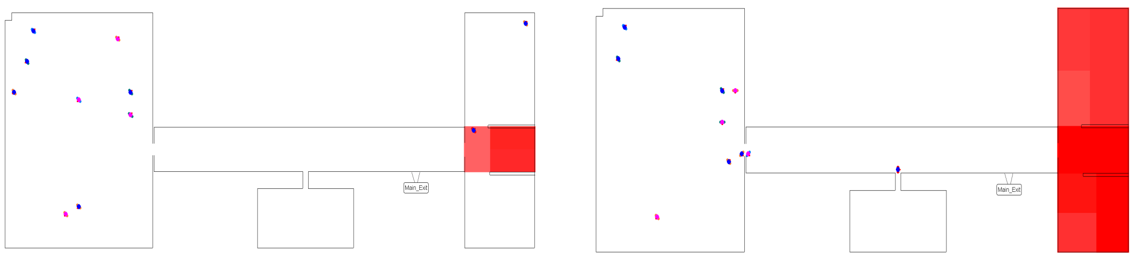
The warden on the other hand shows a 17% higher FIH reading in scenario 2 compared to scenario 1. In both cases he is exposed to heat and radiation while he enters the room on the right to check for people. The higher FIH reading for scenario 2 can be explained as the exposure to heat and radiation in scenario 1 stops right after he closes door_1 whereas in scenario 2 the doors stays open and he faces more heat and radiation on his way back to the main exit. Where the fire had more time to increase the temperature and radiation inside the corridor.

The most obvious effect can be seen for Agent A, who was not exposed to any heat or radiation in scenario 1, as he is the last one to exit the building. In scenario 2 Agent A is the person who has the longest exposure to the most evolved fire conditions.

The closing time of the door also has a direct effect on the amount of smoke present in the corridor, which decreases the visibility and therefore slows down the agents. This can be seen in total simulation times which are 82 seconds in scenario 1 and 83 seconds in scenario 2. The smoke concentration only reaches a critical level within the room on the right, in scenario 1 the warden and Agent B leave the room at an early stage where the fire didn't have enough time to develop. After door_1 gets shut no more smoke or temperature can penetrate into the corridor. In scenario 2 the smoke and temperature can spread through the corridor during the whole time of the simulation as door_1 stays open.

Scenario 3 describes a different evacuation strategy, the warden decides to first notify the people in the room on the left before he decides checking the room on the right, notifying the person in there and closing door_ 1.

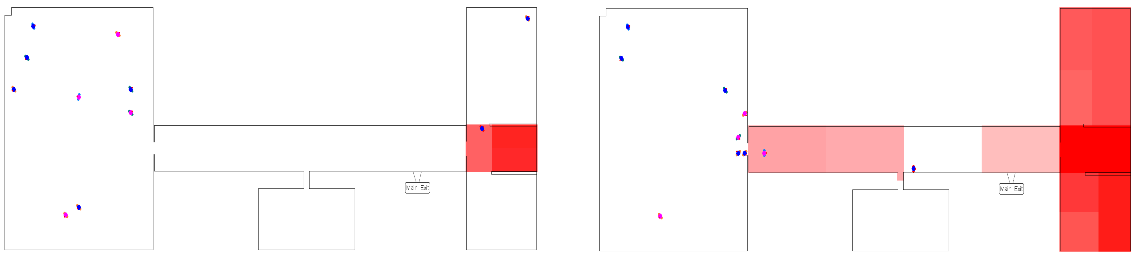
Figure 7.4 to Figure 7.6 show the different temperature distributions in EXODUS for scenarios 1 to 3. Figure 7.4a, Figure 7.5a and Figure 7.6a show the temperature conditions 10 seconds into the simulation run. It can be seen that the same condition are present in all three scenarios as no changes were applied at that stage. In comparison Figure 7.4b, Figure 7.5b and Figure 7.6b show the different outcomes for scenario 1 to 3, 47 seconds into the simulation, which represents the point where the fastest scenario (scenario 3) finishes. It can be seen that by not closing door_1 the temperature inside the corridor increases and agents get exposed to heat and radiation on their way towards the main exit. Scenario three also shows that closing door_1 again after leaving the room on the right, lowers the temperature inside the corridor again.



(a) Warden checking room, walking through high temperature area, 10 seconds into simulation.

(b) Warden leading the agents towards the main exit, 47 seconds into simulation, door_1 closed.

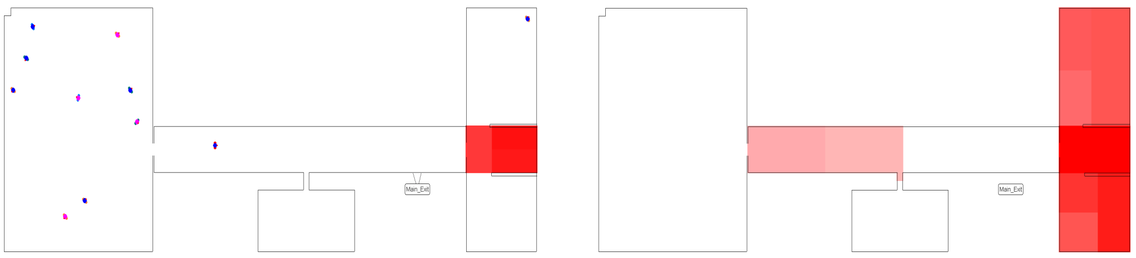
Figure 7.4: Scenario one, warden does close door_1 before evacuating the rest of the building.



(a) Warden checking room, walking through high temperature area, 10 seconds into simulation.

(b) Warden leading the agents towards the main exit 47 seconds into simulation, door_1 stays open.

Figure 7.5: Scenario two, warden does not close door_1 before evacuating the rest of the building.



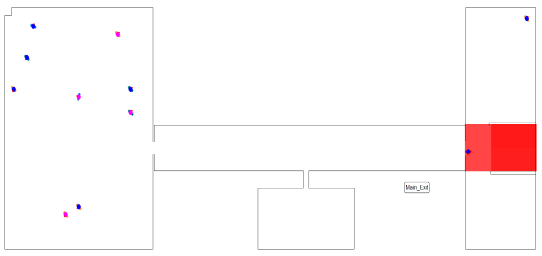
(a) Warden on his way to door_1 after opening door_2 first, 10 seconds into simulation.

(b) Last person just left the building, door_1 closed, 47 seconds into simulation.

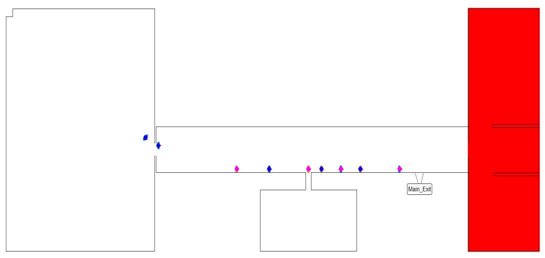
Figure 7.6: Scenario three, warden does close door_1 after he evacuated the last person.

To highlight the differences between scenario 1 and scenario 2, where scenario 2 could almost be modelled in a non-coupled way (by neglecting the closing event of door_2) Figure 7.7 and Figure 7.8 show the outcomes after 60 seconds into the simulation. It can be seen in Figure 7.8b that the longer the evacuation takes place with door_1 open, the higher the temperature levels inside the corridor get and the higher the exposure to heat are for the remaining agents. Despite the different strategies, Figure 7.6 shows similarities to scenario 1 in Figure 7.4, where very little heat spreads into the corridor. This is due to the fact that the whole evacuation only took 47 seconds as all the people in the room on the left get notified at a much earlier stage. Figure 7.6b was taken at 47 seconds into the simulation right after the last person exited the building.

The coupling provides the user with the possibility to test various different strategies to see the effects of human interaction with the environment.

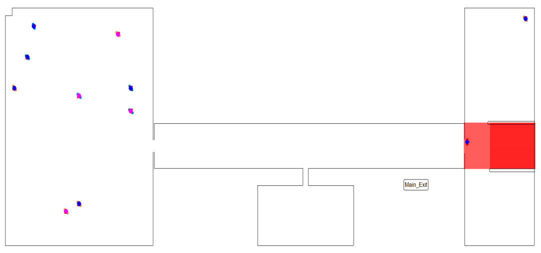


(a) Warden checking room, walking through high temperature area.

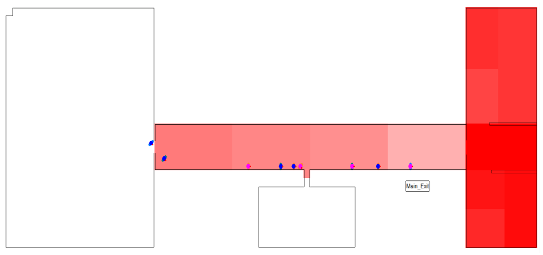


(b) Warden leading the agents towards the main exit at 60s, door_1 closed.

Figure 7.7: Scenario one, warden does close door_1 before evacuating the building.



(a) Warden checking room, walking through high temperature area.



(b) Warden leading the agents towards the main exit at 60s, Door 1 open.

Figure 7.8: Scenario two, warden does not close door_1 before evacuating the building.

The narcotic properties in Table 7.6 show that Agent A who is the last one to leave the building gets no measurable exposure to heat (FIH) but it can be seen that that Agent B and the warden experience much higher effect of the radiation and temperature inside the right room. This is due to the fact that they have to cross the area in front of the fire source at a much later stage as in scenarios 1 and scenario 2. Looking at Figure 7.3c it can be seen that the additional time before shutting door_1 raises the temperature inside the corridor, not as much as in scenario 2 where the door was open the whole simulation, but more than in scenario 1 where the door was shut at an early stage.

Table 7.6: Narcotic properties for Agent A, Agent B and the warden, Scenario 3, coupled.

Agent	FIN	FIH	FIHc	FIHr
Agent A	0.00021956	0.0	0.0	0.0
Agent B	0.00021500	0.13639	0.08695	0.04945
Warden	0.00020582	0.13723	0.09557	0.04166

To further demonstrate the differences caused by varying the scenarios, three more sub-cases have been set up where the wardens initial response time was increased:

- Warden's response time was set to 10 seconds.
- Warden's response time was set to 20 seconds.
- Warden's response time was set to 30 seconds.

It can be seen that the delay of the warden exposes the agents to fire conditions which had an additional 10, 20 and 30 seconds to develop and therefore are more harmful. Tables 7.7, 7.8 and 7.9 show the narcotic property results for Agent B, the warden and Agent A. It also shows that Agent B's narcotic properties, see Table 7.7 between the individual cases, e.g scenario 1 with 10 seconds delay, scenario 2 with 10 seconds delay and scenario 3 with 10 seconds delay only depend on the fire conditions up to the point where he leaves the room. Therefore scenario 1 and scenario 2 are expected to produce similar results. The rest of the population is effected by the delay in events happening, therefore the longer the delay the higher the narcotic property readings. Total evacuation times vary from {83 seconds, 90 seconds, 101 seconds} for scenario 1, {82 seconds, 90 seconds, 101 seconds} for scenario 2 and {77 seconds, 77 seconds, 78 seconds} for scenario 3. The shorter times for scenario 3 are caused by the fact that the warden first gets to inform the agents in the rooms on the left, Agent A in the upper left corner (green circle in Figure 7.1) otherwise is always the last one to exit the building, experiencing the most developed fire conditions if present.

A full list of property outputs can be found in Section A.6 of the Appendix.

Table 7.7: Narcotic properties for Agent B.

Scenario	FIH	FIHc	FIHr
Scenario1 10s delay	0.10770	0.05733	0.05036
Scenario2 10s delay	0.10877	0.05837	0.05041
Scenario3 10s delay	0.22034	0.16325	0.05709
Scenario1 20s delay	0.13547	0.08326	0.05222
Scenario2 20s delay	0.13997	0.08931	0.05066
Scenario3 20s delay	0.38924	0.25619	0.13305
Scenario1 30s delay	0.19858	0.14163	0.05695
Scenario2 30s delay	0.20473	0.14843	0.05630
Scenario3 30s delay	0.64146	0.34209	0.29937

Table 7.8: Narcotic properties for warden.

Scenario	FIH	FIHc	FIHr
Scenario1 10s delay	0.09717	0.06504	0.03212
Scenario2 10s delay	0.10099	0.07332	0.02767
Scenario3 10s delay	0.23008	0.16027	0.06981
Scenario1 20s delay	0.12678	0.08833	0.03845
Scenario2 20s delay	0.15774	0.11751	0.04023
Scenario3 20s delay	0.32482	0.22229	0.10253
Scenario1 30s delay	0.20998	0.14199	0.06799
Scenario2 30s delay	0.24752	0.18093	0.06659
Scenario3 30s delay	0.45576	0.24429	0.21147

Table 7.9: Narcotic properties for Agent A.

Scenario	FIH	FIHc	FIHr
Scenario1 10s delay	0.0	0.0	0.0
Scenario2 10s delay	0.03444	0.03444	0.0
Scenario3 10s delay	0.00891	0.00891	0.0
Scenario1 20s delay	0.00372	0.00372	0.0
Scenario2 20s delay	0.04485	0.04485	0.0
Scenario3 20s delay	0.02053	0.02053	0.0
Scenario1 30s delay	0.00649	0.00649	0.0
Scenario2 30s delay	0.13803	0.09142	0.04661
Scenario3 30s delay	0.02068	0.02068	0.0

From the narcotic property readings it can be seen that scenario 1 has the most favourable outcome for the population. This goes against the assumption that it is most beneficial that the warden always should rescue the majority of the people first (scenario 3). To further investigate this situation, scenario 1 and scenario 3 have been modelled as another case with an initial 75 seconds delay time for the warden. It can be seen in the outcomes in Table 7.10, that scenario 1 now results in the worst case, where everyone would have eventually died as the warden would have never made it to the second room. The warden enters the room on the right after about 80 seconds, the 75 seconds delay results in fire conditions that are much worse than in previous scenarios, Agent B dies right after the warden enters, at around 85 seconds (see FIH readings for Agent B in Table 7.10, FIH values greater than 1 are fatal). The warden also dies on his way back towards door_1 because of his high exposure to heat and radiation right in front of the fire source. The warden's death directly results in the fact that the other people won't get notified any more, giving the simulation basically an infinite total evacuation time.

Scenario 3 with the 75 seconds delay applied results in similar conditions for the warden and Agent B, 2 fatalities, but in contrast to scenario 1 the other agents can escape the building, see Table 7.11. Agent B dies at around 85 seconds as he is facing the same conditions as in

Table 7.10: Narcotic properties for Agent A, Agent B and the warden, additional scenario 1, 75 seconds delay, coupled.

Agent	FIN	FIH	FIHc	FIHr
Agent A	0.00046707	0.0	0.0	0.0
Agent B	0.00038723	1.00655	0.42340	0.58315
Warden	0.00047099	1.00704	0.56736	0.43969

Table 7.11: Narcotic properties for Agent A, Agent B and the warden, additional scenario 3, 75 seconds delay, coupled.

Agent	FIN	FIH	FIHc	FIHr
Agent A	0.00050663	0.31945	0.13721	0.18225
Agent B	0.00038723	1.00655	0.42340	0.58315
Warden	0.00050571	1.00501	0.53508	0.46993

the previous 75 seconds scenario 1 case, which is only determined by the time the fire had to develop, the warden dies at 109 seconds, just after he reaches Agent B's location. The last person to leave the building is Agent A at 113 seconds. Although this resulted in 2 fatalities, still all the other agents would have survived compared to 100% fatalities in the other case. The scenarios show a broad variety of outcomes for different starting conditions and behaviours. The scheduling of those scenarios would have been almost impossible because of the various unknowns which are determined during runtime, e.g when will the warden execute his tasks while he is slowed down to different environmental conditions like smoke or heat. The coupling offers the possibility to simulate a broad range of scenarios which are determined by the interaction of the two models removing the uncertainties of making too many assumption by setting up the models. A fire engineer can now apply various concepts and ideas in a very reasonable amount if time, in the minutes to tenth of minutes.

7.2 Coupling Case 2: Care Facility

Coupling case two consists of the geometry described in Section 5.2 with added agents. The population consists of 154 people, 150 aged between 65 and 90 years with mixed abilities and 4 staff members. To highlight the different environmental conditions throughout the geometry a total of 93 measuring zones have been set up (see Figure 5.6 and Figure 5.7), of which 10 zones are probes in the rooms closest to the fire source (zone 84 to zone 93). The simulated test cases cover different scenarios with the main purpose to protect and evacuate the people in building as quickly and safely as possible. The strategy in all the test cases is to shield of the fire source from the rest of the building as quickly as possible and therefore protect the residents from heat, radiation and smoke. This can only be achieved by closing off the two doors connecting the staircase to the ground and first floor. Furthermore a strategy to ensure minimal impact on the health of residents, where it is assumed many of them are bedridden and cannot be moved that easily, is implied. The doors connecting the main corridors with the rooms to the left and right get shut first, to stop smoke and heat penetrating into them, then the fire is shielded off by closing the staircase doors and afterwards the people in the rooms get evacuated safely. In order to demonstrate the capabilities of the coupled version and to highlight the complexity of interaction between changing CFD boundary conditions and adapted behaviour of the agents in the evacuation model a warden has been introduced who will lead the evacuation procedure in different scenarios.

The coupling was applied to the following three different scenarios:

- Scenario 1: A warden located on the ground floor shuts all doors on the ground floor corridor first, on the way back he evacuates the people from within these rooms followed by the same steps upstairs.
- Scenario 2: A warden located on the ground floor shuts all doors on the first floor corridor first, on the way back he evacuates the people from within these rooms followed by the same steps downstairs.
- Scenario 3: Two wardens, one located on each floor, shut all doors on their corresponding floor's corridors first, on the way back they evacuate the people from within these rooms.

Figures 7.9 and Figure 7.10 show what the output of the coupled simulation in EXODUS for the care facility looks like. Figure 7.9 shows the temperature output and Figure 7.10 the smoke output. The outputs shown are the end results for Scenario 3, discussed in Section 7.2.2.

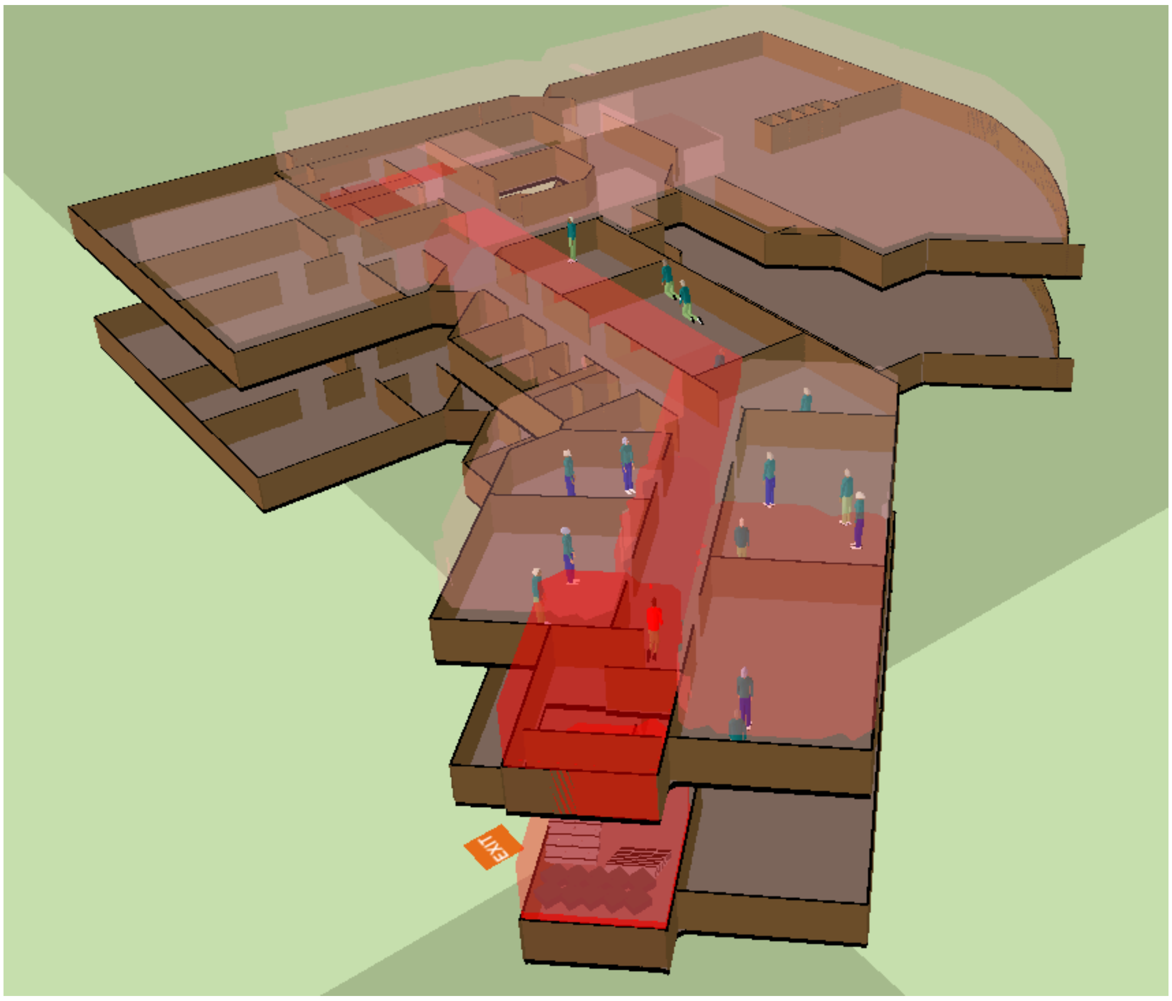


Figure 7.9: VR Exodus output for the temperature distribution 251 seconds into simulation.

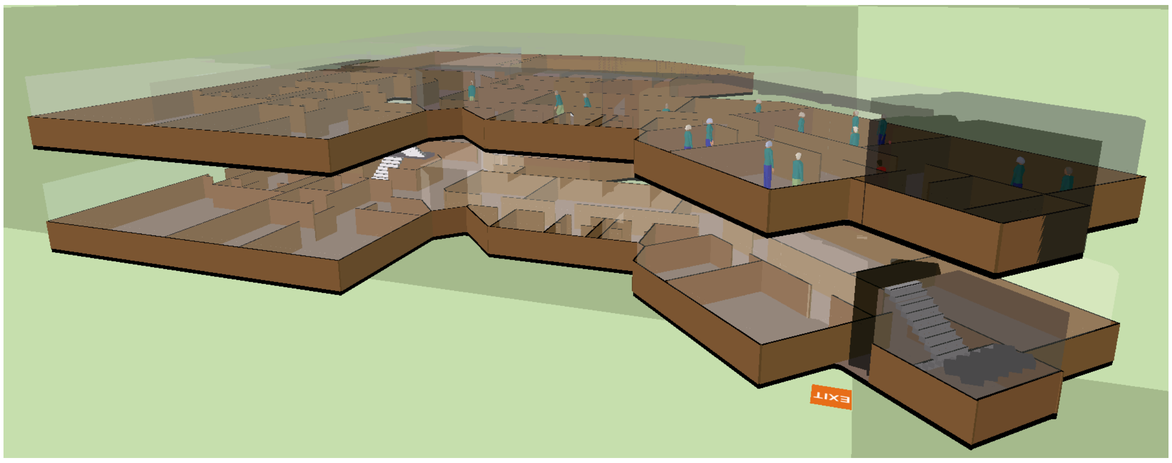


Figure 7.10: VR Exodus output for the smoke distribution 251 seconds into simulation.

The wardens starting locations can be seen in Figure 5.6 and Figure 5.7 highlighted by the green circles. The outcomes of all three scenarios will be evaluated by looking at the environmental factors which influence the results of an evacuation simulation. Five measurement probes were set up on each floor as it can been seen in the red dots it Figure 5.6 and Figure 5.7. These probes are used to output temperature, smoke density and ray intensity over the time of the simulation runs. The measurement probes have been placed inside the rooms closest to the fire, the further away of the source the smaller the readings get.

7.2.1 Coupled vs. Non-Coupled

To further highlight the differences in the outcomes of the coupled and non-coupled version of the simplified scenario in Section 7.1. This section will compare the coupled and non-coupled case for scenario 1 and scenario 3 against each other. Scenario 2 was not simulated using the non-coupled approach as it is very similar to scenario 1.

7.2.1.1 Scenario 1

As already mentioned in Section 7.1 a non coupled approach was also applied to the care facility scenario. The ASET / RSET method was used to calculate the time it will take the warden to reach the two exits, Door 25 and Door 26. These two timings are purely based on the distance and the travel speed of the warden. The fire case was calculated using these inputs, which results in the fact that the two doors close much earlier than they would do in the coupled case, as the warden doesn't get slowed down by smoke. As can be seen in Table 7.12 the door on the ground floor closes at 36 seconds and the first floor door at 125 seconds compared to 55 seconds and 255 seconds respectively in the coupled scenario shown in Table 7.17.

Table 7.12: Door events scenario 1 serial.

Door number	Time [s]	Action
Door 26	36	Closes
Door 25	125	Closes

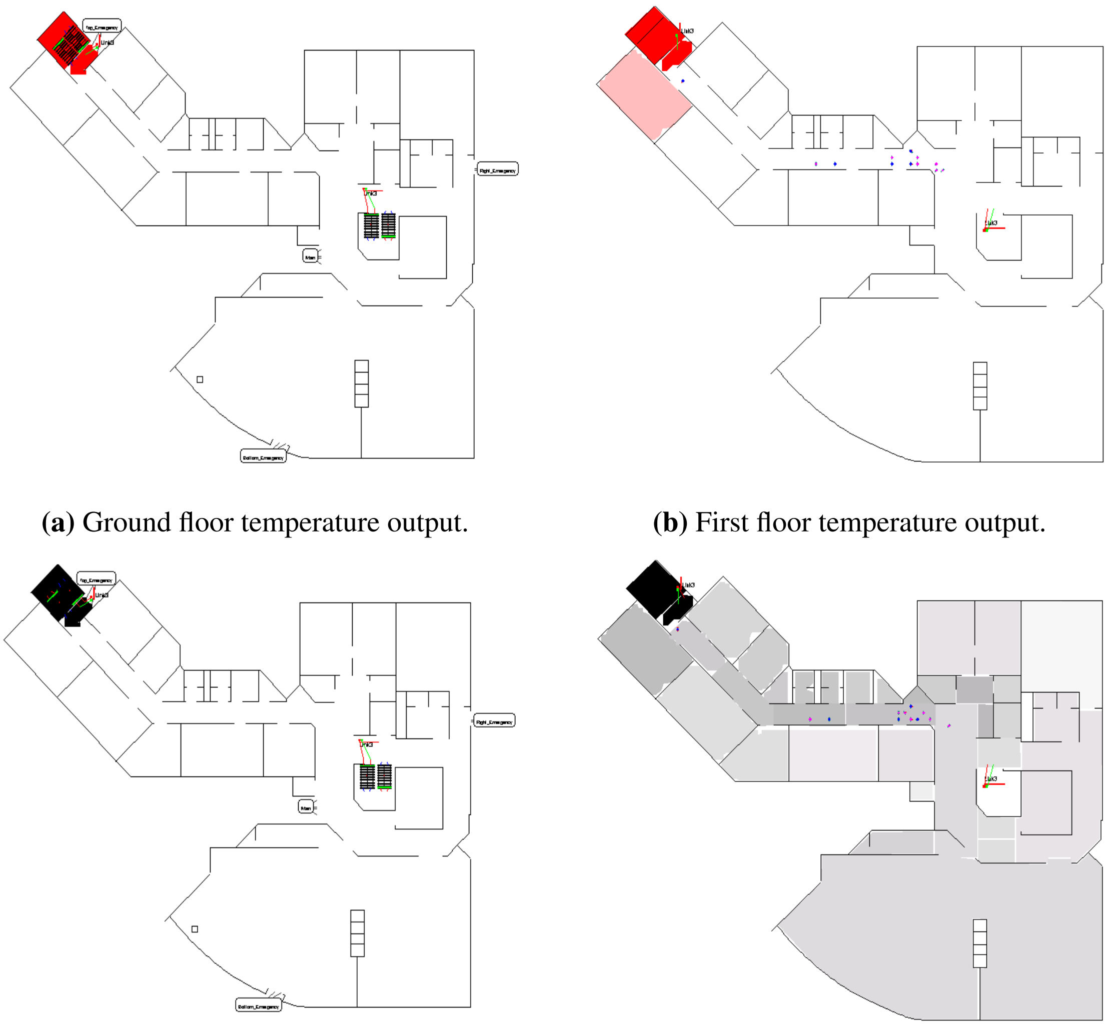
Table 7.13 shows the narcotic properties for the warden. It can be seen that FIH, FIHc and FIHr values are all zero, in comparison to the coupled outputs of scenario 1 in Table 7.18 where FIH and FIHr are equal to 0.82904 where 1 resembles the fatal threshold. The big differences in the narcotic outputs are a results of the fact that the warden in the coupled case experiences a high dose of radiative heat when he gets close to the two doors 25 and 26

Table 7.13: Property output for the warden at the end of the simulation for scenario 1 serial.

Person	FIN	FIH	FIHc	FIHr
Warden	0.00189	0.0	0.0	0.0

when closing them. In the non-coupled case, the doors close too early, by the time the warden arrives at the door locations, the CFD temperature and radiation data is already based on the closed door scenario and therefore there is no exposure to the fire source any more.

The temperature and smoke outputs for the non-coupled case can be seen in Figure 7.11, in comparison to the outputs for the coupled case in Figure 7.29 it can be seen that there is a big difference in the temperature and smoke distribution on the first floor. This can be explained by the fact that in the coupled case, the door stays open for 255 seconds (compared to 125 seconds in the non-coupled case) and the fire has 130 seconds more time to spread



(a) Ground floor temperature output.

(b) First floor temperature output.

(c) Ground floor smoke output.

(d) First floor smoke output.

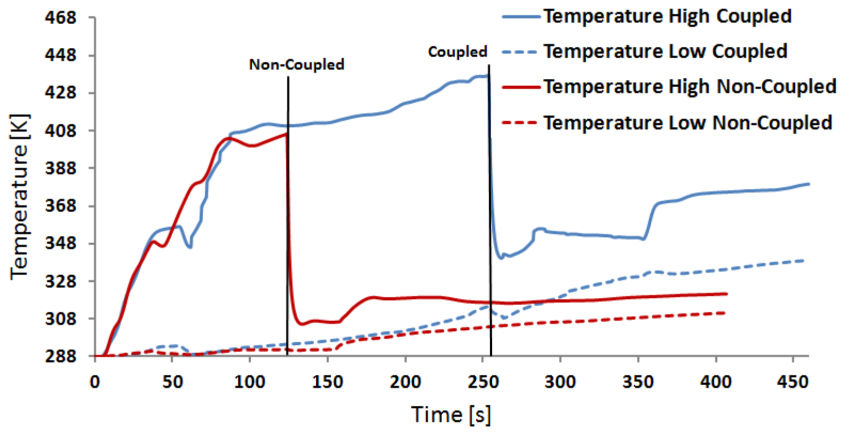
Figure 7.11: Exodus temperature and smoke outputs for both floors for scenario 1 non-coupled.

temperature and smoke on the upper floor.

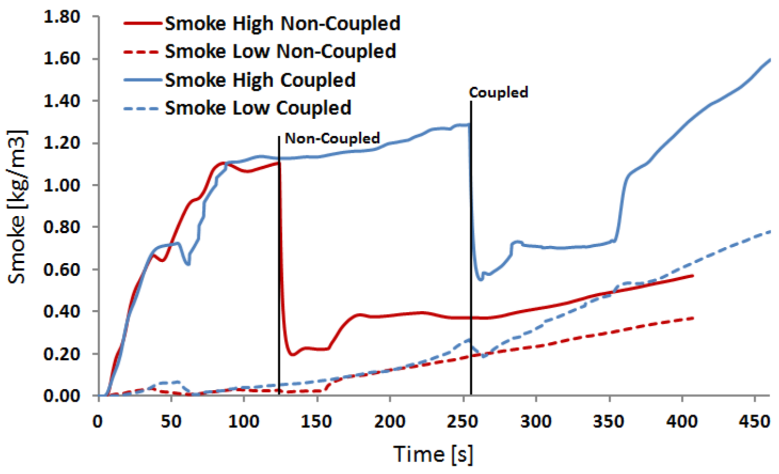
This influences the outcomes of the whole simulation the first person exits the building at 38 seconds, the last person (warden) at 406 seconds, the second to last person at 352 seconds. In the coupled version the first person leaves the building at 66 seconds, the last person at 458 seconds, the warden leaves the building at 408 seconds. The overall differences between the last persons out is 106 seconds (352 seconds to 458 seconds).

Figures A.43 to A.52 show the combined graphs for coupled versus non-coupled temperature, smoke and radiation outputs for the probes 84 to 93 shown in Figure 5.6 and Figure 5.7. The blue vertical line in the graphs are the times when door 25 and door 26 respectively open on each floor.

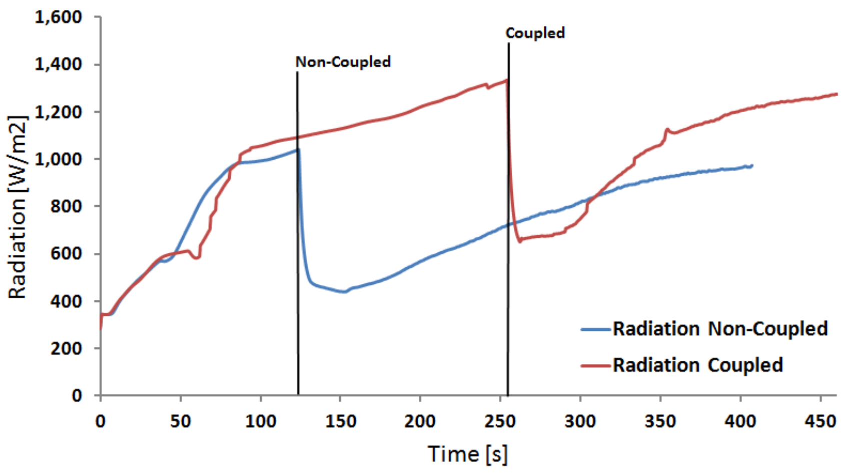
The following Figures 7.12,7.13,7.14,7.15,7.16 and 7.17 show the temperature, smoke and radiation outputs for the probes centrally located in zones 67,86,84,91,63 and 98 , see Figure 5.6 and Figure 5.7. It can be seen how the results for the coupled and serial version differ, due to the different times when the staircase doors were closed. The steep drops in values highlight how the closing of the door changes the conditions around the staircase. Figure 7.14 and Figure 7.17 are an exception as they both had another door event happening which was not covered in the serial version. This can be seen in the in the premature drop of values before the staircase door was closed. These graphs show again what a big difference in event times and therefore in environmental conditions the two way coupling makes.



(a) Temperature

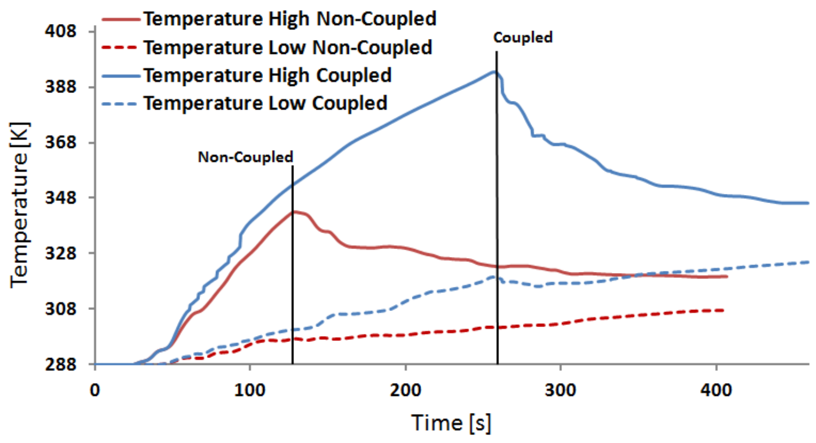


(b) Smoke

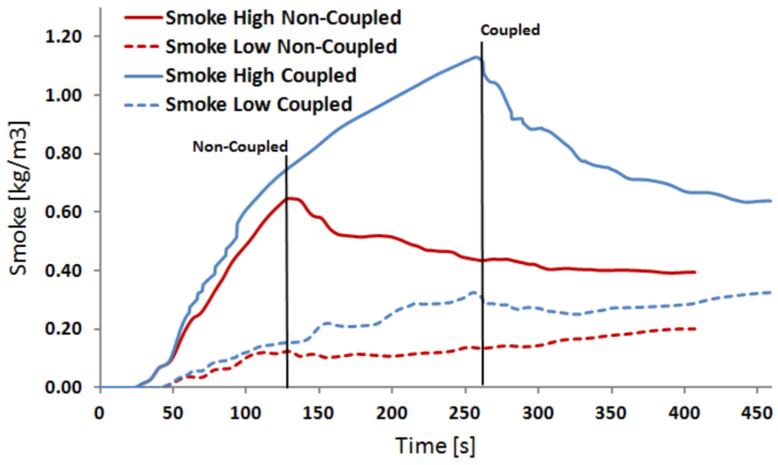


(c) Radiation

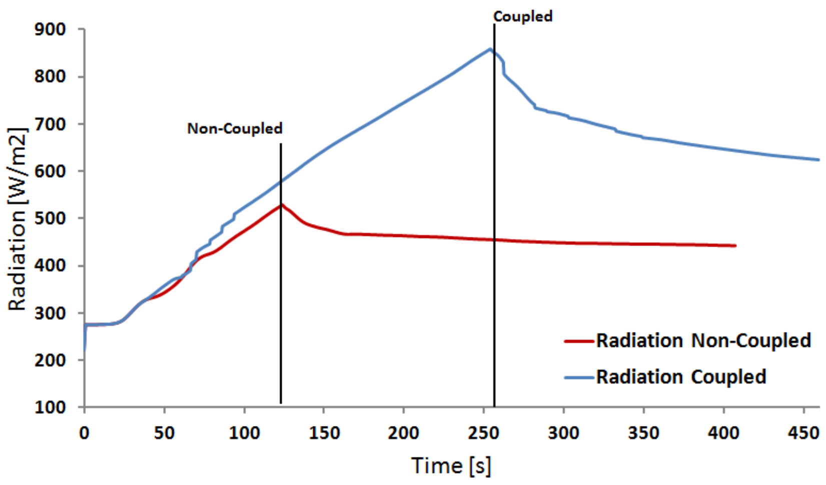
Figure 7.12: Temperature, Smoke and Radiation outputs for Zone 67



(a) Temperature

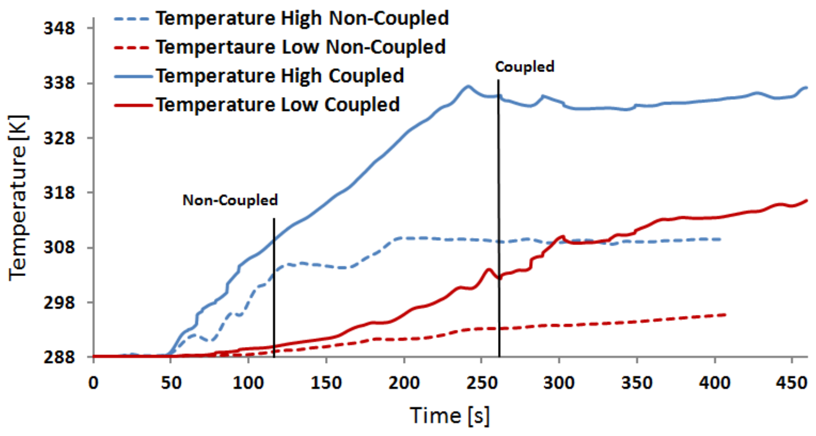


(b) Smoke

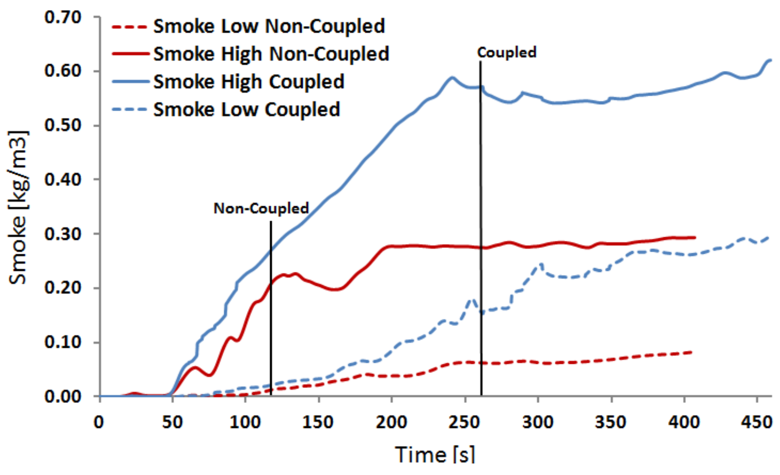


(c) Radiation

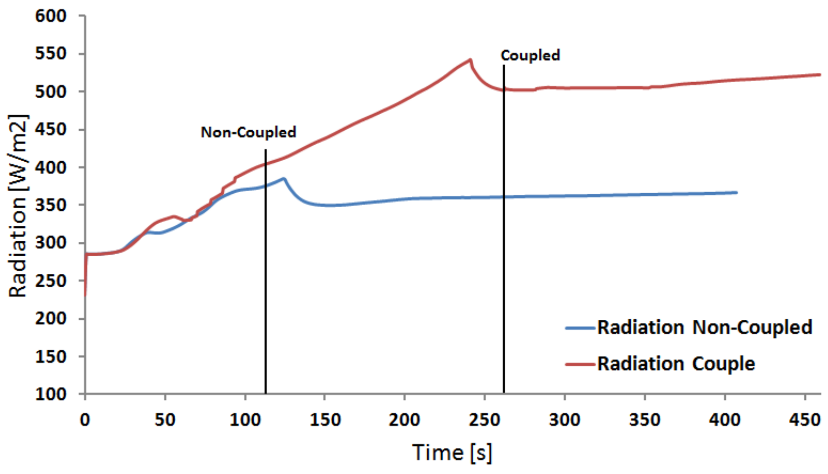
Figure 7.13: Temperature, Smoke and Radiation outputs for Zone 86



(a) Temperature

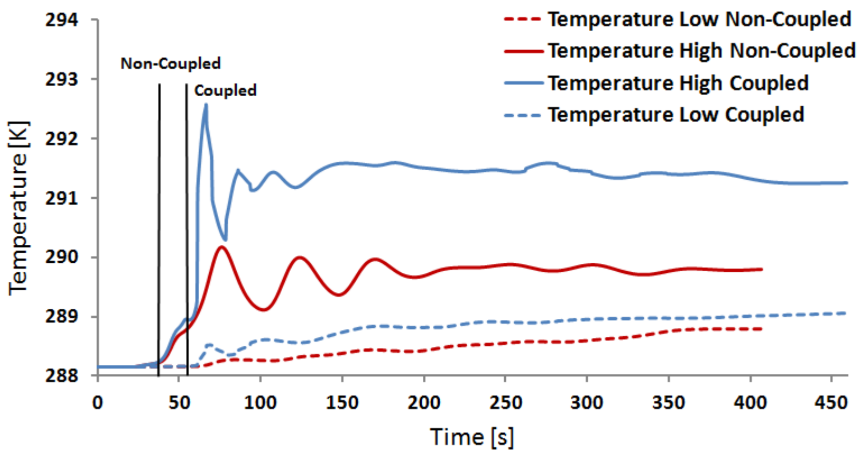


(b) Smoke

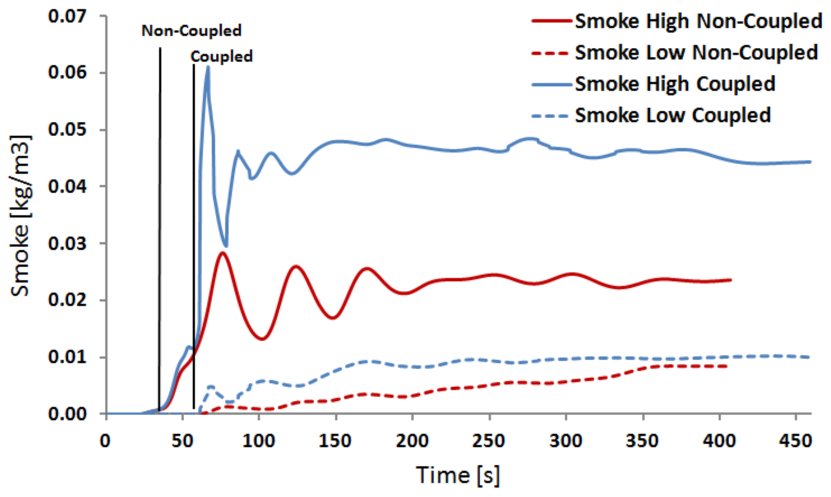


(c) Radiation

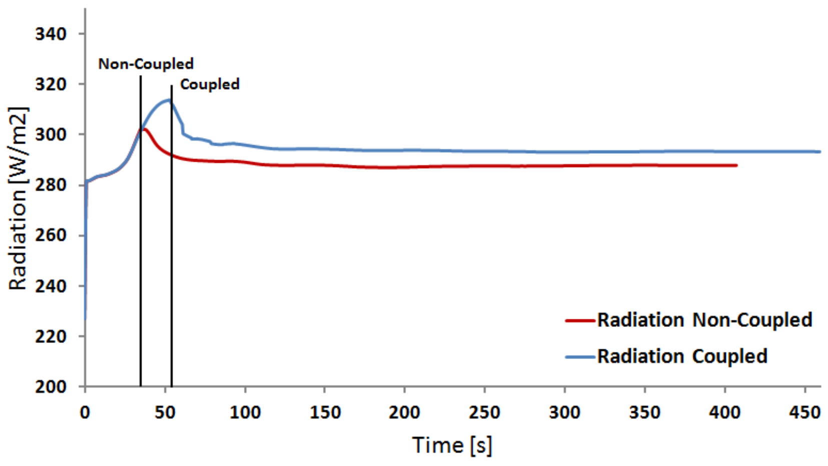
Figure 7.14: Temperature, Smoke and Radiation outputs for Zone 84



(a) Temperature

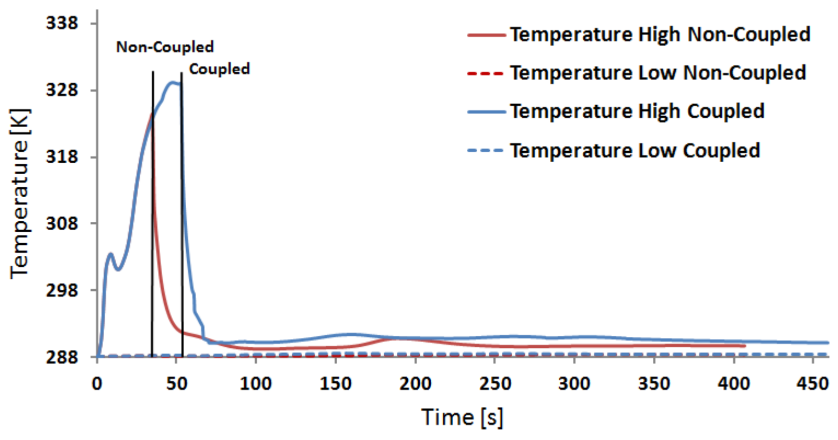


(b) Smoke

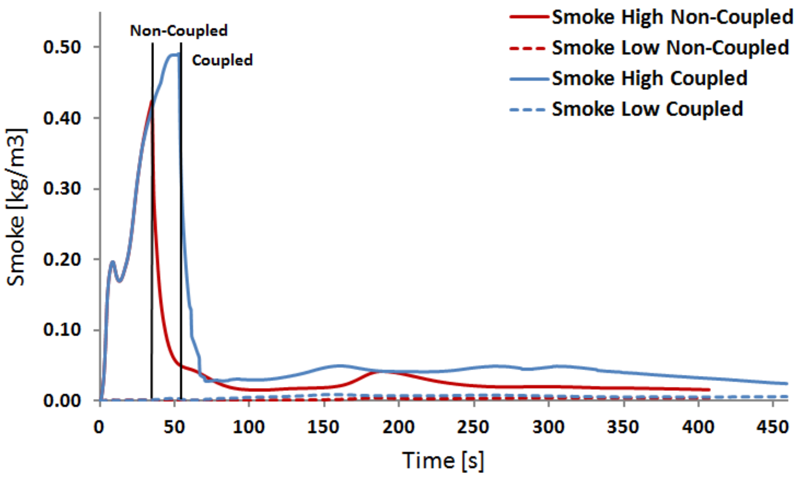


(c) Radiation

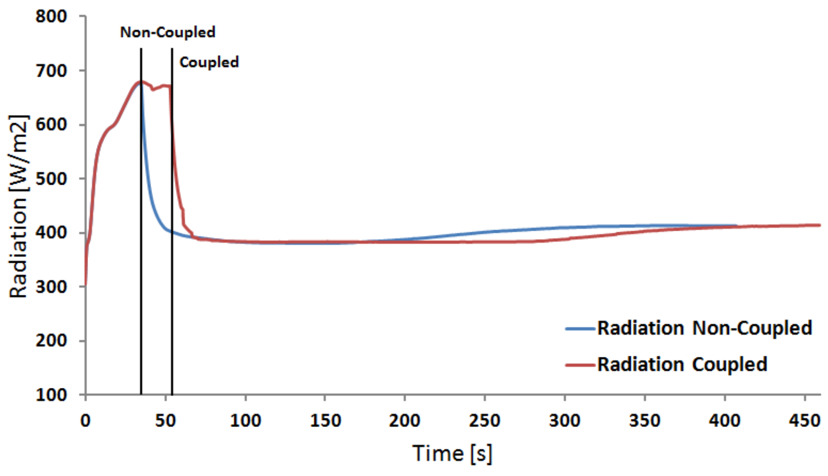
Figure 7.15: Temperature, Smoke and Radiation outputs for Zone 91



(a) Temperature

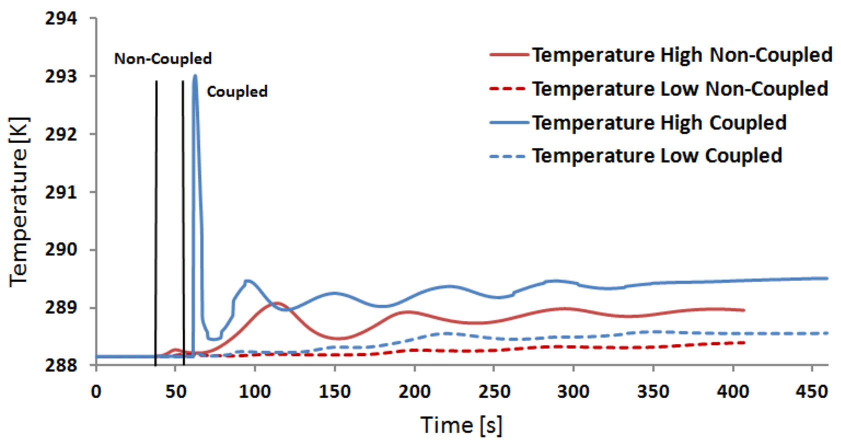


(b) Smoke

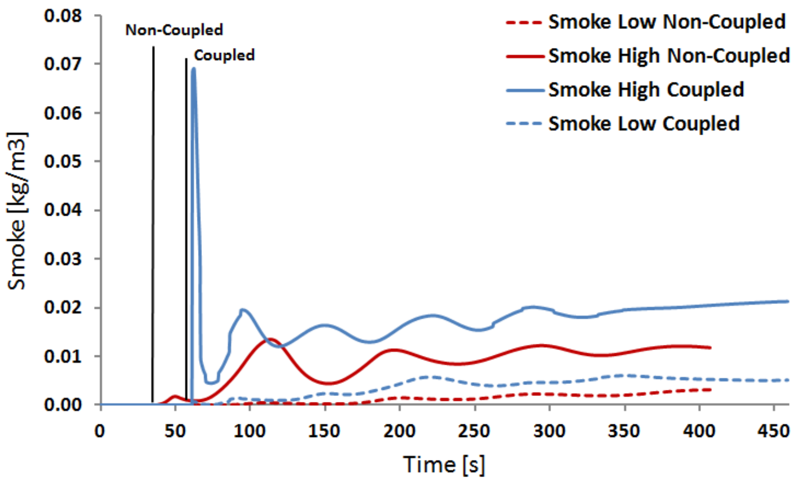


(c) Radiation

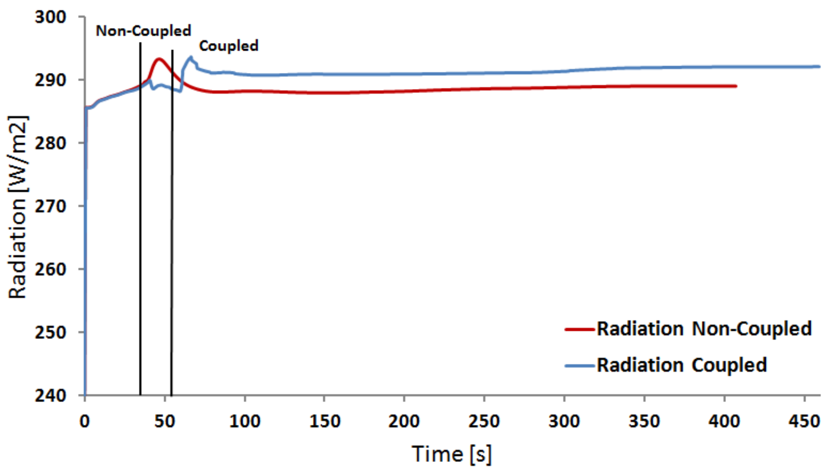
Figure 7.16: Temperature, Smoke and Radiation outputs for Zone 63



(a) Temperature



(b) Smoke



(c) Radiation

Figure 7.17: Temperature, Smoke and Radiation outputs for Zone 89

7.2.1.2 Scenario 3

To complete the comparison between coupled and non-coupled results, scenario 3 will be compared against the non-coupled execution. In contrast to the other scenarios, scenario 3 introduces another level of complexity, regarding agent interaction with the geometry. The results highlighted in Section 7.2.2 were obtained by introducing a second warden, one per floor. The additional warden reduces the simulation time by reducing the distance a single warden has to travel to complete his tasks. This introduces another layer of complexity, as the number of event times (which are strongly dependent on the travel speeds and therefore the environmental conditions) doubles. As in the previous non-coupled example the event times have been determined by calculating the time it takes the warden to travel from point A to point B. The two door event times for door 25 and door 26 can be seen in Table 7.14, they are identical as the starting locations of the wardens (see Figure 5.6 and Figure 5.7) are in the same area but on different floors.

Table 7.14: Door events scenario 3 serial.

Door number	Time [s]	Action
Door 26	38	Closes
Door 25	38	Closes

As it can be seen in Table 7.14 the door on the ground floor closes at 38 seconds and the first floor door at 38 seconds compared to 54 seconds and 70 seconds respectively in the coupled scenario shown in Table 7.21. This is due to the fact that for the serial version, where the times are purely based on the distance and travel speeds and the environmental conditions were disregarded. As for the coupled version it can be seen that it takes the first floor warden 16 seconds longer to reach his target, which can be explained by the raised smoke levels on the first floor. Tables 7.15 and 7.16 show the narcotic properties for the ground floor warden and first floor warden. The coupled results in Table 7.22 and Table 7.23 show that the FIN readings are slightly higher, 0.00045890 non-coupled versus 0.00049078 coupled for the warden on the ground floor and 0.00056039 non-coupled versus 0.00059830 coupled for the first floor warden. Which can be explained by the delayed arrival at the two doors and the therefore raised temperature and radiation levels.

Table 7.15: Fractional Incapacitating Doses for the ground floor warden at the end of the simulation for scenario 3 serial.

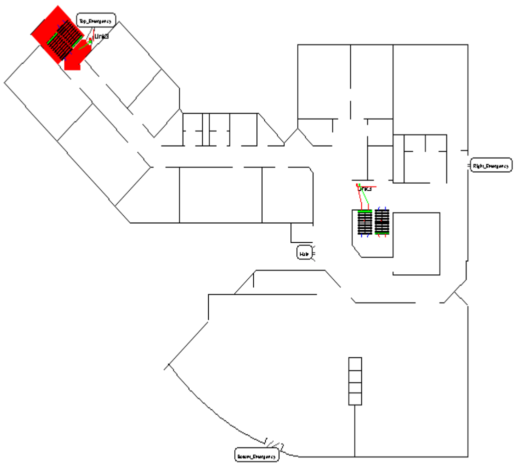
Person	FIN	FIH	FIHc	FIHr
Warden	0.00045890	0.0	0.0	0.0

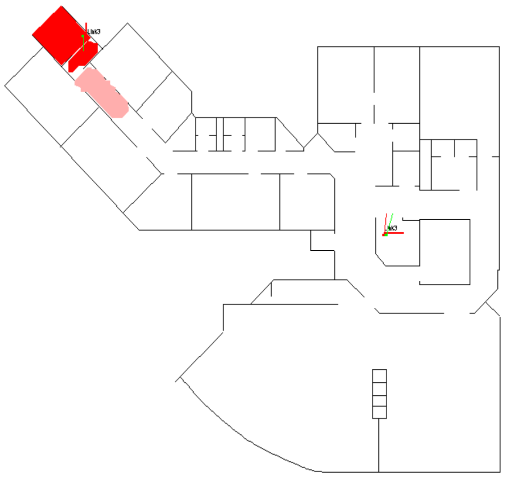
Table 7.16: Fractional Incapacitating Doses for the first floor warden at the end of the simulation for scenario 3 serial.

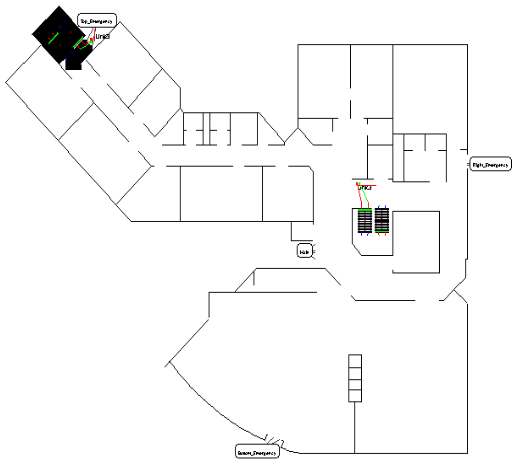
Person	FIN	FIH	FIHc	FIHr
Warden	0.00056039	0.0	0.0	0.0

The temperature and smoke distribution over both floors can be seen in Figure 7.18. In comparison to the coupled results in Figure 7.31 it can be seen that by closing the doors earlier the temperature level and smoke concentration next to the door on the first floor differs. Closing the door earlier results in higher readings. Figures 7.19 to Figure 7.22 show the temperature and smoke distributions in a cut-plane-view through the building at different times during the simulation for the coupled and non-coupled execution. The staircase can be seen on the left. The measurement times haven been chosen around the actual door events at 38 seconds for non-coupled and 70 seconds for the coupled case. It can be seen that at the end of the simulation at 245 seconds, for the non-coupled version the upper temperature layer descends more than in the coupled version.

Figure 7.21 shows the same cut plane view for the smoke output. As in the temperature version the times the measurements were taken fall around the door event time. Comparing the results against Figure 7.22 it can be seen that the smoke outputs follows the same pattern around the door as the temperature output.


(a) Ground floor temperature output.


(b) First floor temperature output.


(c) Ground floor smoke output.

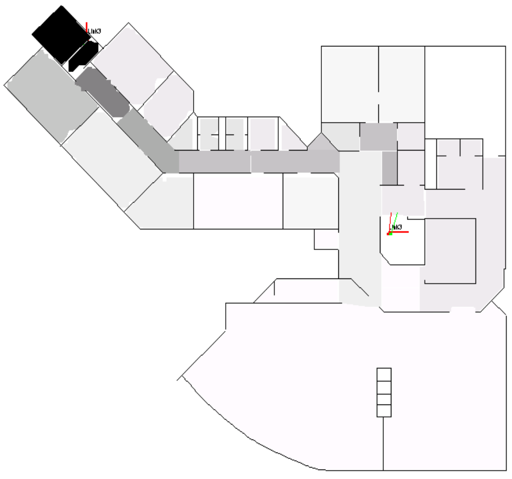
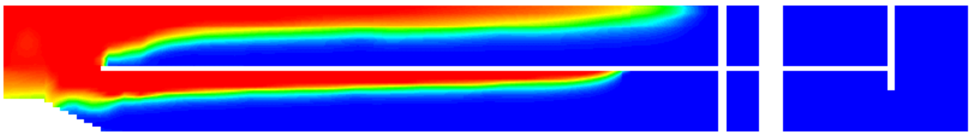
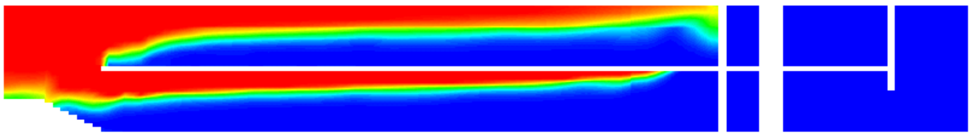

(d) First floor smoke output.

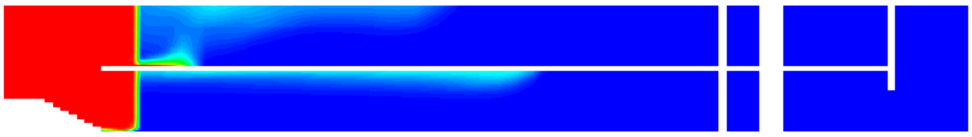
Figure 7.18: Exodus temperature and smoke outputs for both floors for scenario 3 non-coupled.



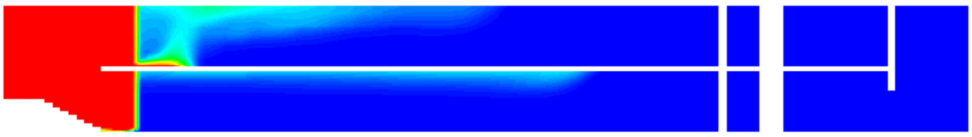
(a) 35 seconds.



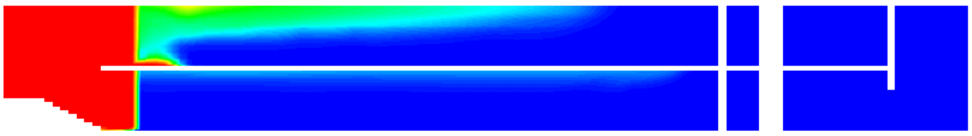
(b) 40 seconds.



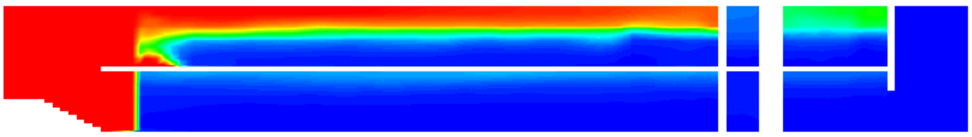
(c) 45 seconds.



(d) 55 seconds.



(e) 75 seconds.



(f) 245 seconds.

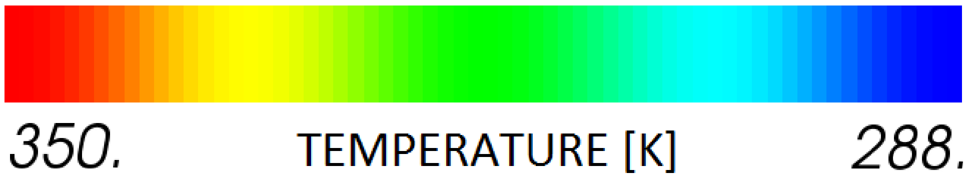
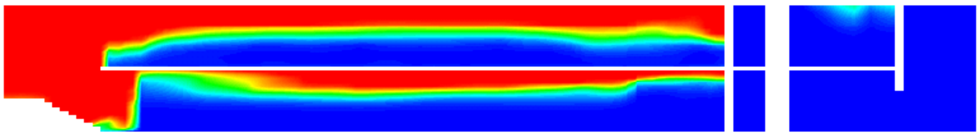
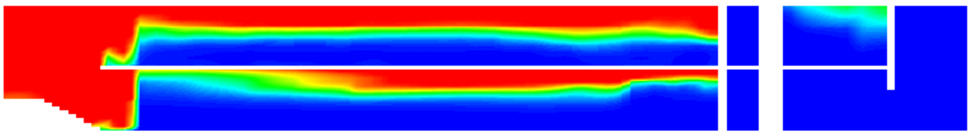


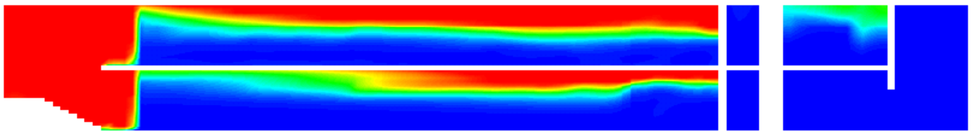
Figure 7.19: Non-coupled CFD temperature outputs around the first floor door event and at the end of simulation, door event at 38 seconds.



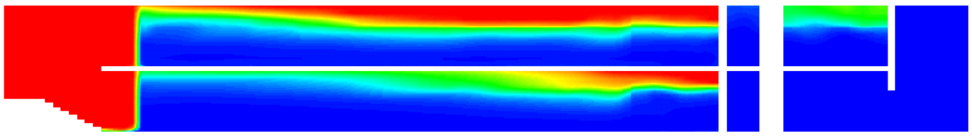
(a) 65 seconds.



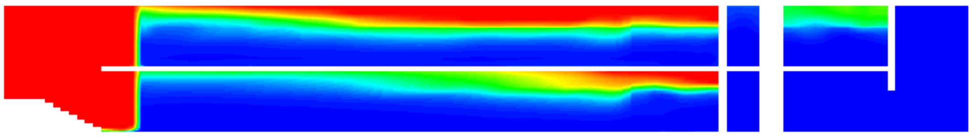
(b) 70 seconds.



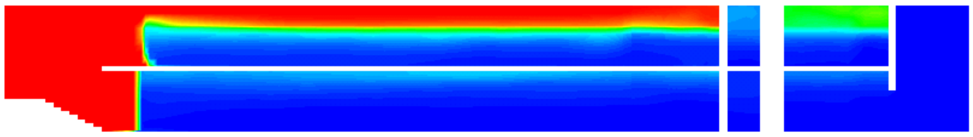
(c) 75 seconds.



(d) 85 seconds.



(e) 100 seconds.



(f) 251 seconds.

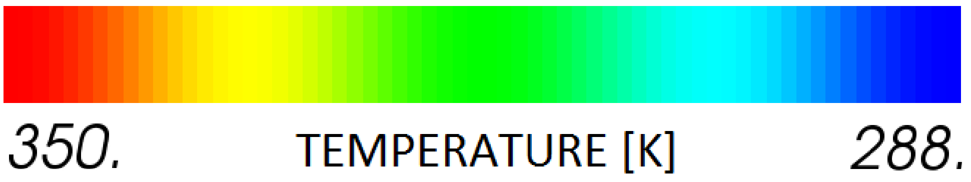
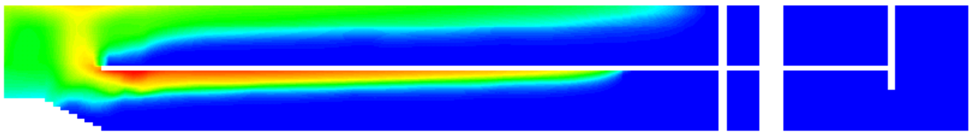
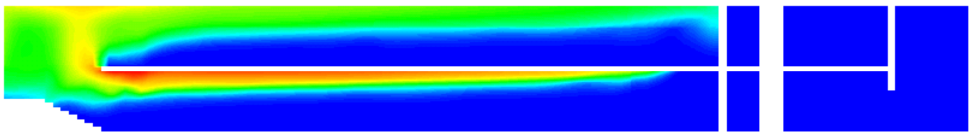


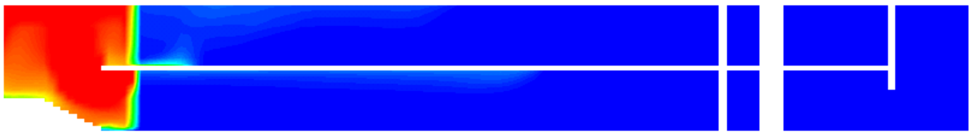
Figure 7.20: Coupled CFD temperature outputs around the first floor door event and at the end of simulation, door event at 70 seconds.



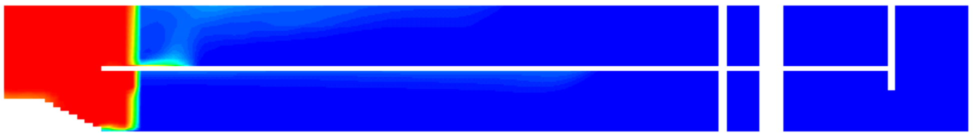
(a) 35 seconds.



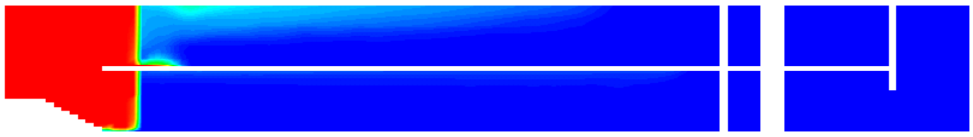
(b) 40 seconds.



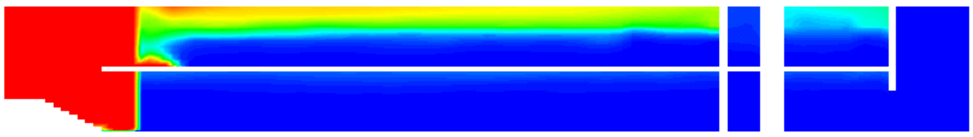
(c) 45 seconds.



(d) 55 seconds.



(e) 75 seconds.



(f) 245 seconds.

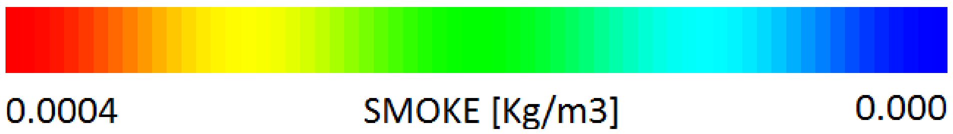
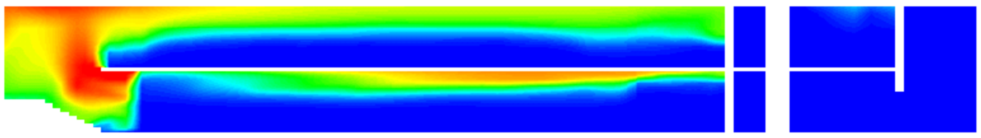
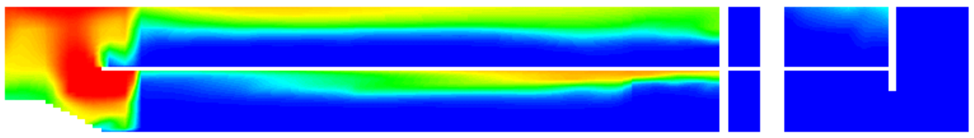


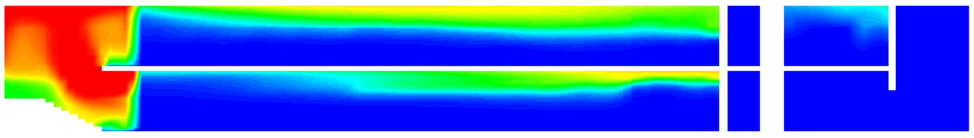
Figure 7.21: Non-coupled CFD smoke outputs around the first floor door event and at the end of simulation, door event at 38 seconds.



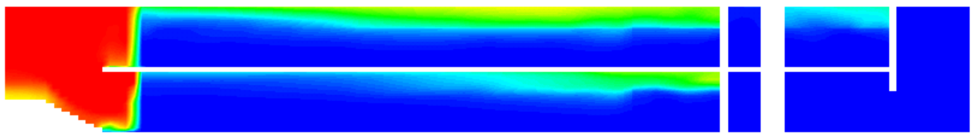
(a) 65 seconds.



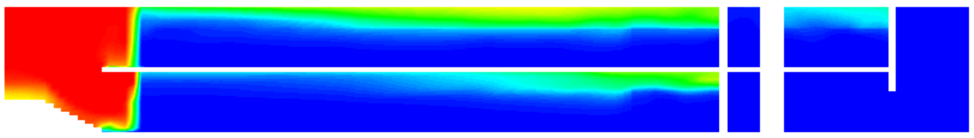
(b) 70 seconds.



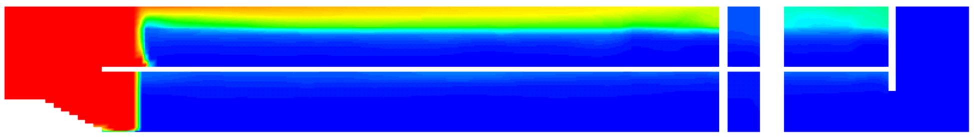
(c) 75 seconds.



(d) 85 seconds.



(e) 100 seconds.



(f) 251 seconds.

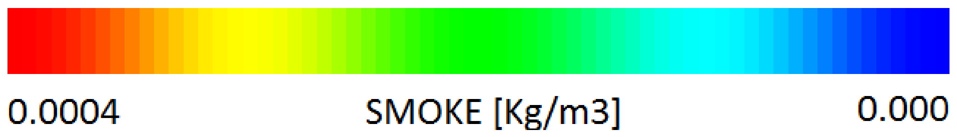
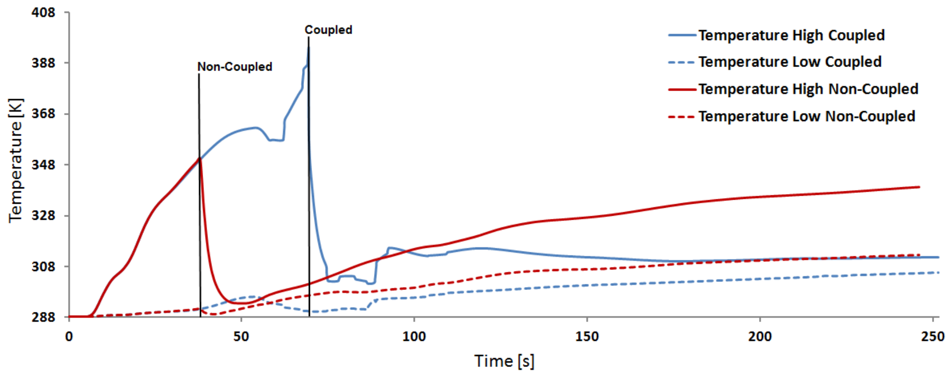
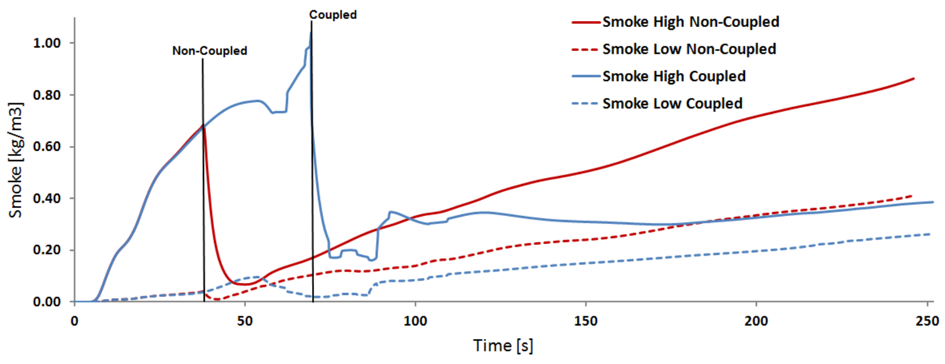


Figure 7.22: Coupled CFD smoke outputs around the first floor door event and at the end of simulation, door event at 70 seconds.

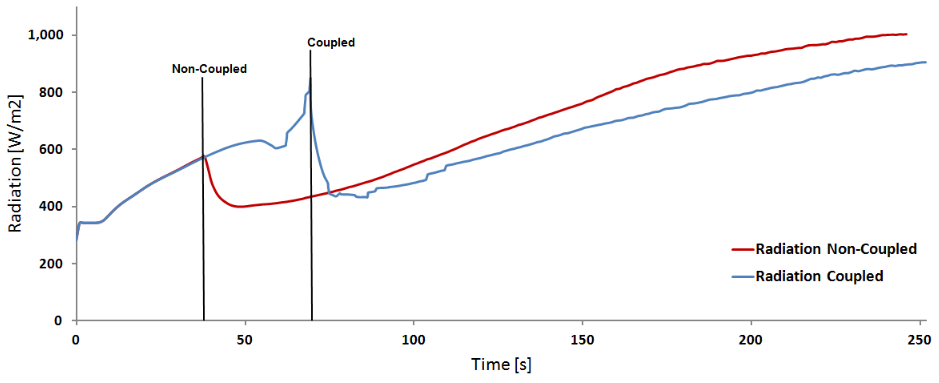
The following Figures 7.23,7.24,7.25,7.26,7.27 and 7.28 show the temperature, smoke and radiation outputs for the probes centrally located in zones 67,86,84,91,63 and 98 , see Figure 5.6 and Figure 5.7. It can be seen how the results for the coupled and serial version differ, due to the different times when the staircase doors were closed. The steep drops in values highlight how the closing of the door changes the conditions around the staircase. Figure 7.25 and Figure 7.28 are an exception as they both had another door event happening which wasn't covered in the serial version. This can be seen in the in the premature drop of values before the staircase door was closed. These graphs show again what a big difference in event times and therefore in environmental conditions the two way coupling makes.



(a) Temperature

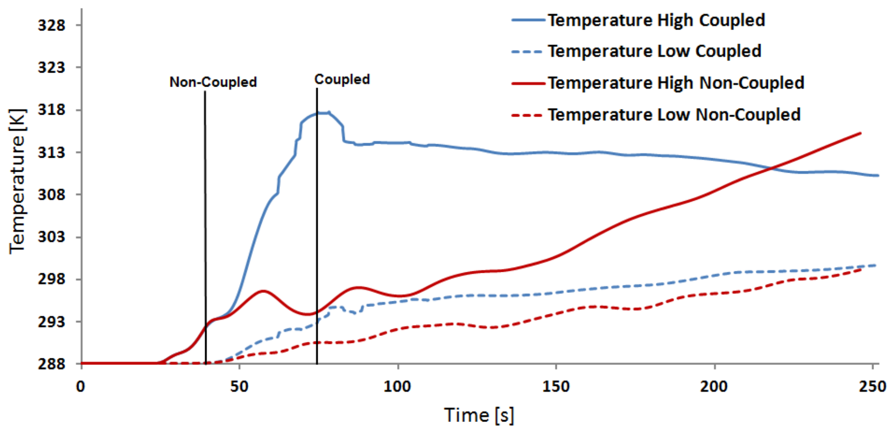


(b) Smoke

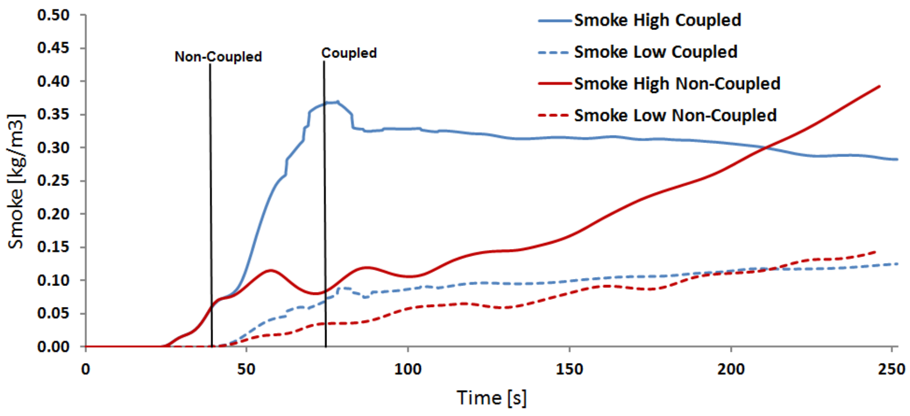


(c) Radiation

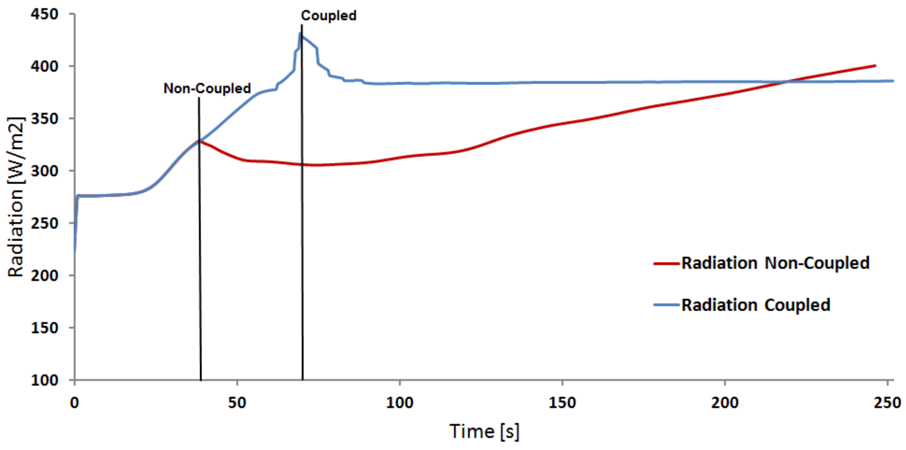
Figure 7.23: Temperature, Smoke and Radiation outputs for Zone 67



(a) Temperature

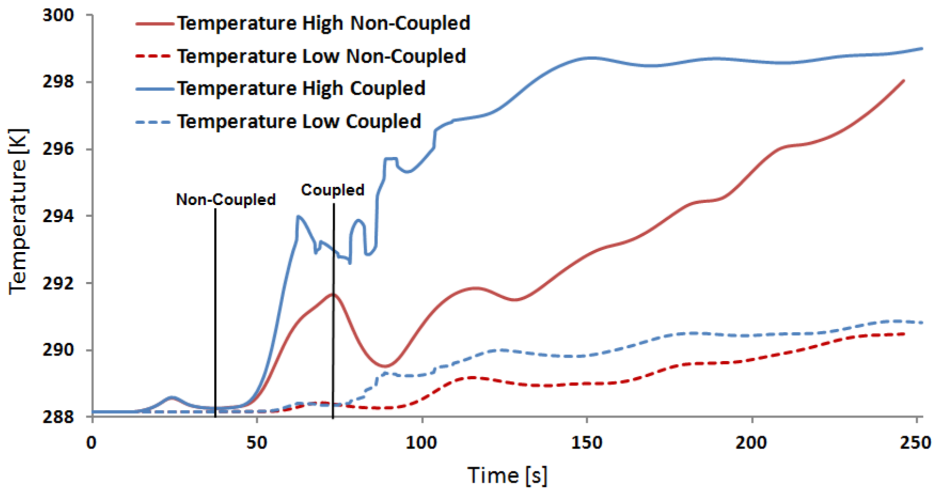


(b) Smoke

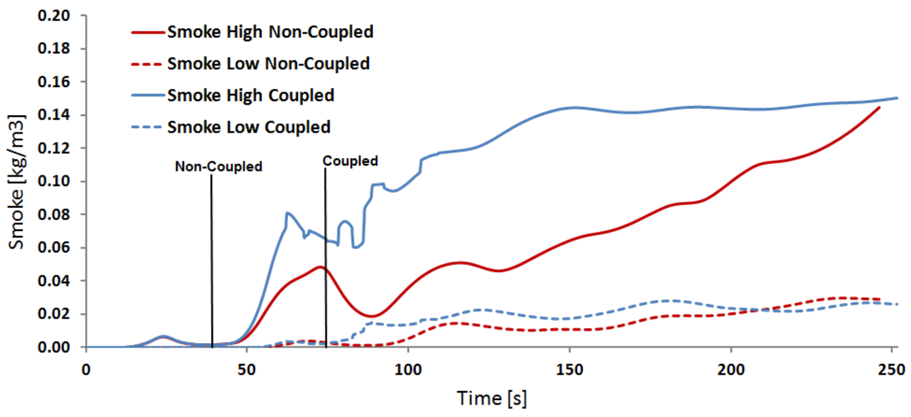


(c) Radiation

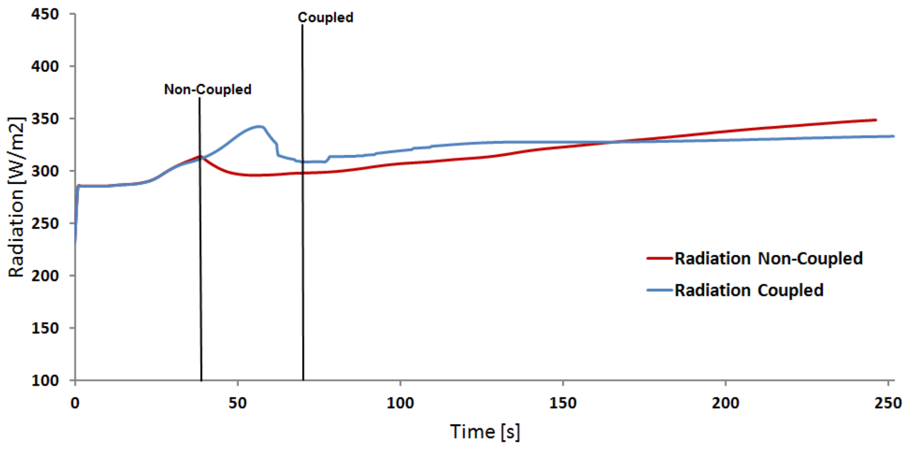
Figure 7.24: Temperature, Smoke and Radiation outputs for Zone 86



(a) Temperature

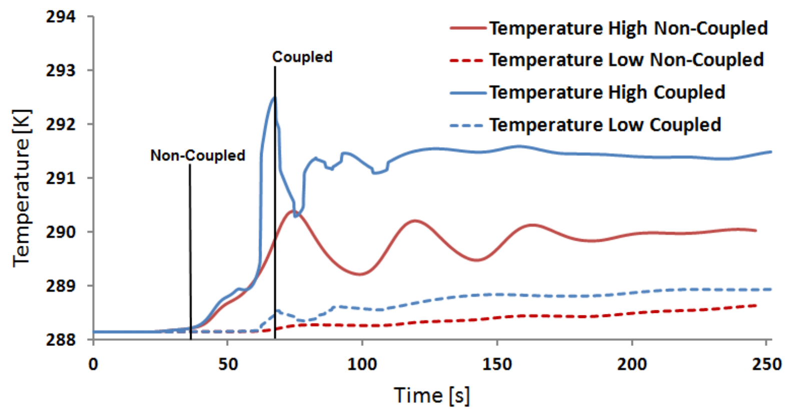


(b) Smoke

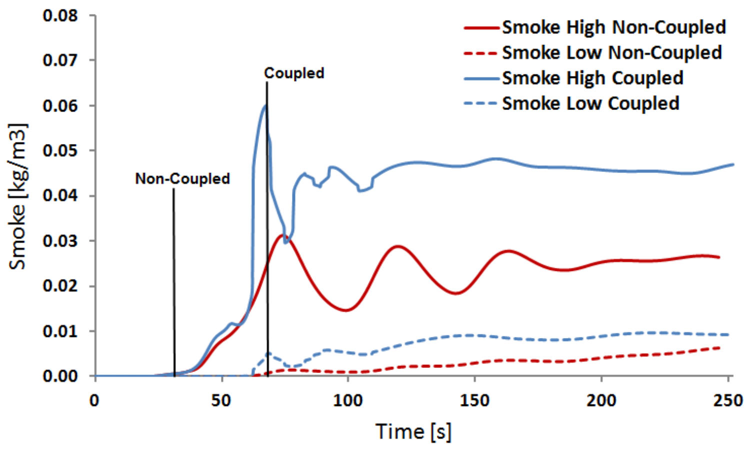


(c) Radiation

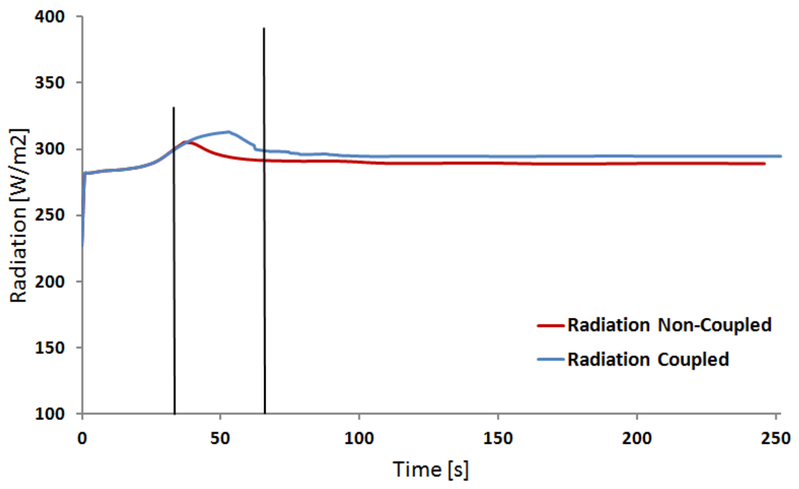
Figure 7.25: Temperature, Smoke and Radiation outputs for Zone 84



(a) Temperature

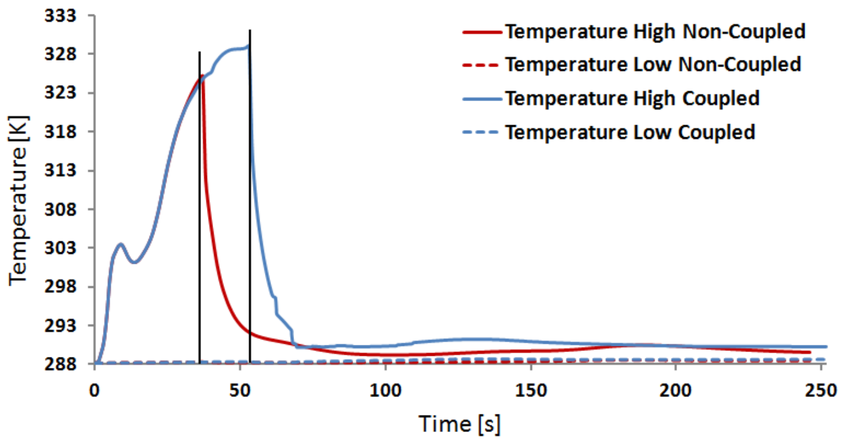


(b) Smoke

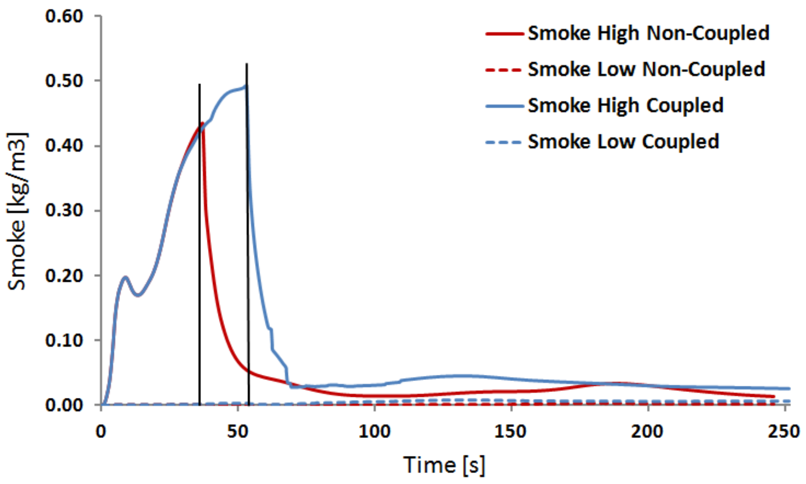


(c) Radiation

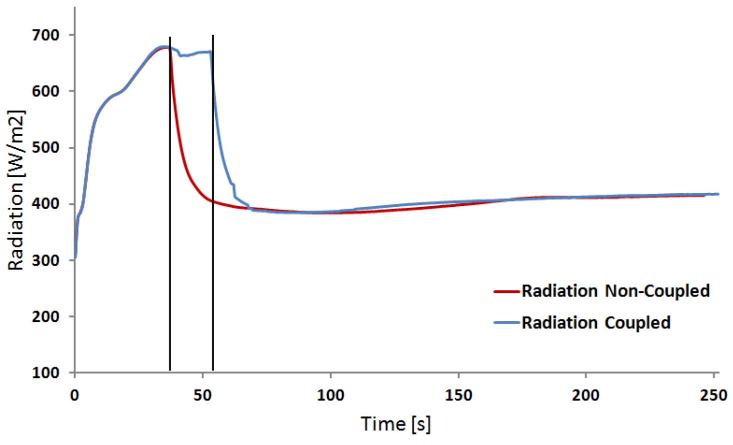
Figure 7.26: Temperature, Smoke and Radiation outputs for Zone 91



(a) Temperature

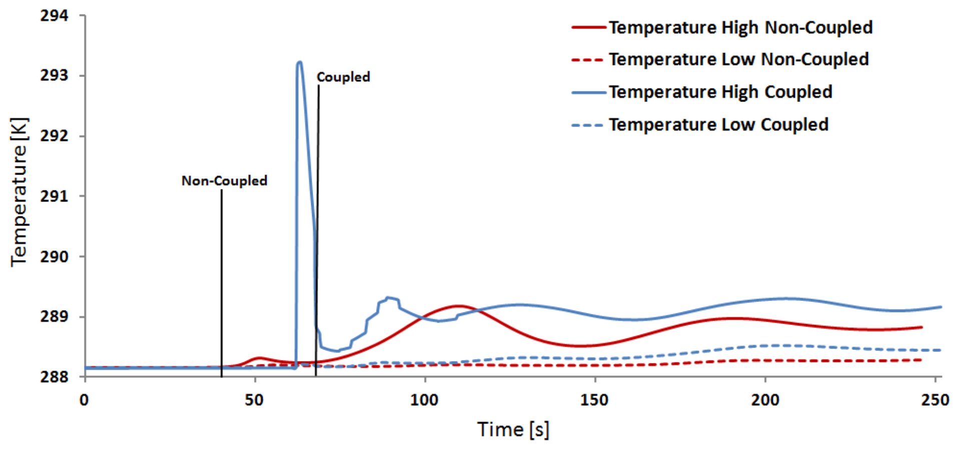


(b) Smoke

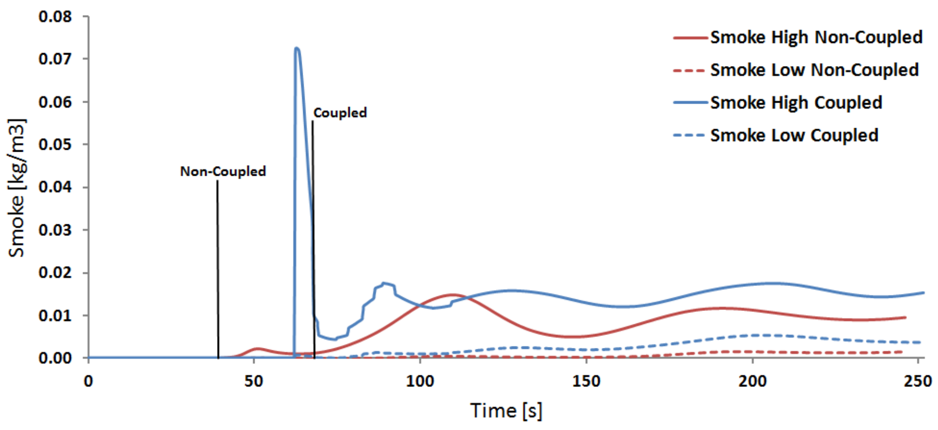


(c) Radiation

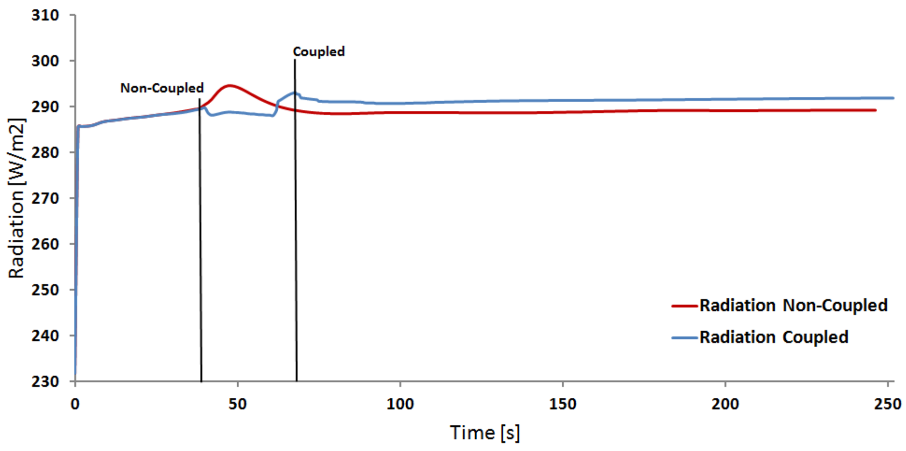
Figure 7.27: Temperature, Smoke and Radiation outputs for Zone 63



(a) Temperature



(b) Smoke



(c) Radiation

Figure 7.28: Temperature, Smoke and Radiation outputs for Zone 89

7.2.2 Further Coupling Scenarios

After highlighting the differences the two-way coupled simulation produces compared to the one-way coupled, this section will show the possibilities it offers. The two-way coupling enables a much more complex interaction of the agents with the environment enabling the Fire Safety Engineer (FSEs) to accurately model agent-geometry interaction. The following three scenarios are all using the same care facility geometry but introducing different scenarios in which one or more warden manages the evacuation procedure to reduce the impact the fire, smoke and radiation have on the other agents.

Scenario 1 Results

In Scenario 1 the warden starts off in a room opposite the second staircase (not the one with the fire source) on the ground floor, making his way towards the door which connects the other staircase on the ground floor first. After closing the door he returns to his starting area to go up the staircase to the first floor to close the second door which connects the other staircase to the floors. On his way through both corridors he closes all the doors to rooms with people in. It is assumed that these people are elderly, bed-bound patients of the care facility who are not very mobile. In order to protect them from smoke and heat the warden closes the doors first. He then shields of the hazard from the rest of the building by closing the staircase doors and opens up the rooms again to safely evacuate those people.

Table 7.17 shows the times when the warden closes and re-opens the doors, Door 26 and Door 25 (in bold) are the doors to the staircase with the hazard in it. The effects the environment has on the warden can be seen in Table 7.18. The warden's final FIH and FIHr levels are 0.82904 respectively, where 1.0 resembles the fatal threshold. FIH stands for the individual's exposure to radiative and convective heat. FIHr stands for the individual's exposure to radiative heat. These high readings are the result of the exposure to the fire source when the warden closes the door. The majority of the high readings come from the second door event at 255 seconds, the fire had 255 seconds to develop and the bottom door to the staircase had already been closed at 55 seconds so all the smoke and the heat got more concentrated inside the staircase.

Table 7.17: Door events scenario 1.

Door number	Time [s]	Action
Door 1	12	Closes
Door 2	17	Closes
Door 3	24	Closes
Door 5	28	Closes
Door 11	31	Closes
Door 12	41	Closes
Door 26 (GF)	**55**	**Closes**
Door 12	62	Opens
Door 11	69	Opens
Door 5	72	Opens
Door 3	81	Opens
Door 2	87	Opens
Door 1	94	Opens
Door 24	153	Closes
Door 23	171	Closes
Door 22	201	Closes
Door 21	213	Closes
Door 14	220	Closes
Door 13	242	Closes
Door 25 (FF)	**255**	**Closes**
Door 13	263	Opens
Door 14	283	Opens
Door 21	290	Opens
Door 22	304	Opens
Door 23	333	Opens
Door 24	351	Opens

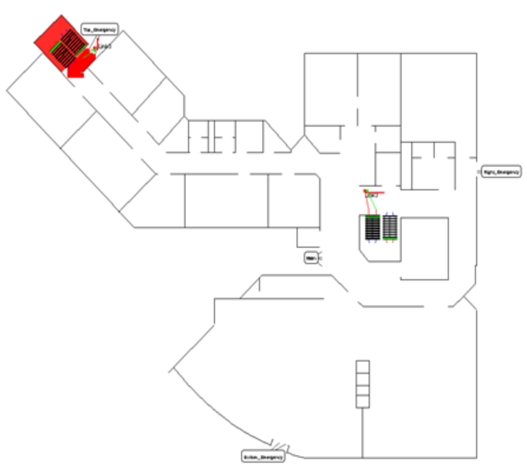
Table 7.18: Property output for the warden at the end of the simulation for scenario 1 coupled.

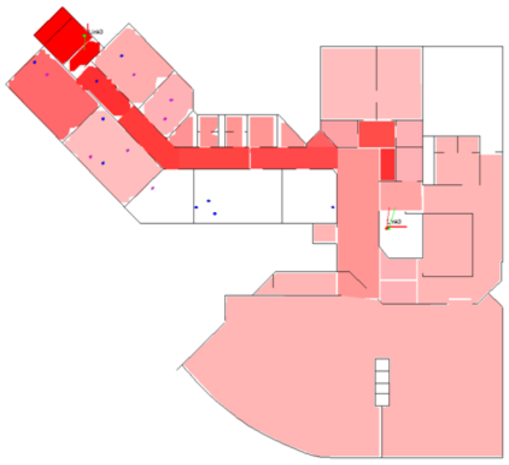
Person	FIN	FIH	FIHc	FIHr
Warden	0.00186	0.82904	0.0	0.82904

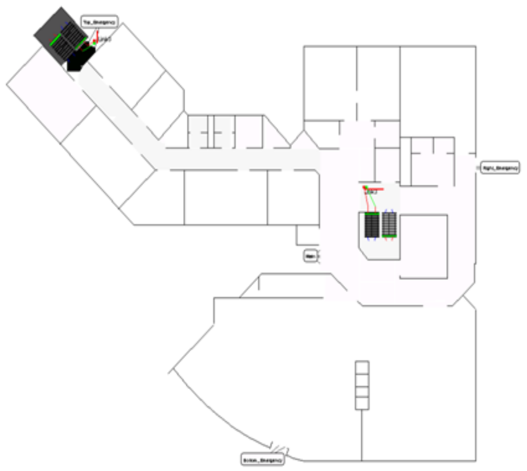
The smoke and temperature distribution on both floors can be seen in Figure 7.29, (a) shows how the closed door on the ground floor increases the temperature readings (around 650 ° C for zone 4) within the staircase. Figure 7.29 (b) shows how the higher temperatures spread through the corridor away from the staircase door. The same can be seen for the smoke distribution in Figure (c) and (d). The smoke concentration has a direct effect on the agents travel speeds and therefore effects the evacuation times for each agent. The first person leaves the building at 66 seconds, the last one at 458 seconds, the warden exits at 408 seconds.

The following graphs show the detailed temperature, smoke and radiation values for the most important zones, 84,86,67 on the first floor and 89,91 and 63 (zones and measuring probes highlighted in Figure 5.6 and Figure 5.7 on the ground floor.

All full list of all temperature, smoke and radiation values versus simulation time curves for zones 84 to 93 can be seen in Appendix Section A.7. The two fine blue lines in the graphs shows the two times when the doors close and re-open.


(a) Ground floor temperature output.


(b) First floor temperature output.


(c) Ground floor smoke output.

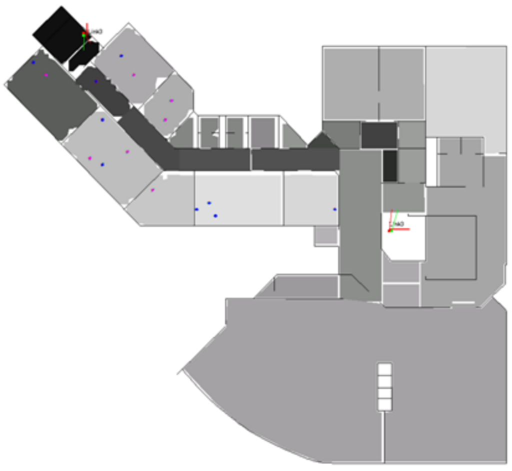

(d) First floor smoke output.

Figure 7.29: Exodus temperature and smoke outputs for both floors for scenario 1.

Scenario 2 Results

In scenario 2 the warden's starting position remains the same as in scenario 1, but in contrast to scenario 1 the warden makes his way upstairs first. He essentially executes the same tasks only in reverse order, closing doors and informing people on floor one first followed by the same procedure on the ground floor.

Table 7.19 shows the times when the warden closes and re-opens the doors, Door 26 and Door 25 (in bold) are the doors to the staircase with the hazard in it. The effects the environment has on the warden can be seen in Table 7.20. The warden's final FIH and FIHr levels are 0.92444. The majority of the high readings come from the second door event at 370 seconds, the fire had 370 seconds to develop and the first floor door to the staircase had already been closed at 108 seconds so all the smoke and the heat got more concentrated inside the staircase and spread throughout the ground floor corridor, this can be seen by comparing Figures 7.29 and 7.30.

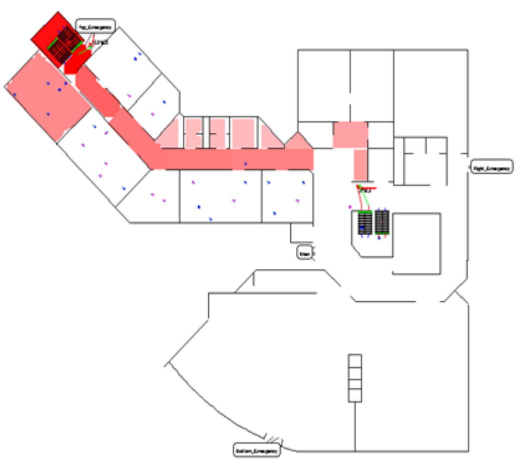
The smoke and temperature distribution for scenario 2 can be seen in Figure 7.30, (a) shows how the open door on the ground floor increases the temperature readings within the corridor. Figure (b) shows how the lower temperature spread through the corridor due to the earlier closing of the staircase door. The same can be seen for the smoke distribution in Figure (c) and (d). The temperature in zone 63 on the ground floor near the staircase reaches levels of up to 123 $^\circ$ C. The smoke concentration has a direct effect on the agents travel speeds and therefore effects the evacuation times for each agent. The first person leaves the building at 66 seconds, the last one at 582 seconds. It can also seen that in comparison to scenario 1 the total evacuation time increases by around 27 per cent from 458 seconds to 582 seconds. This can be explained by the fact that the fire is located on the ground floor, close to the staircase door, the longer the fire has time to develop and spread through the corridor the worser the conditions for the warden get to close the second door, exposing him to higher temperatures. First person out : 65.96 seconds Last person out : 581.84 seconds
Detailed information about temperature, smoke and radiation values versus simulation time for zones 84 to 93 (measuring probes highlighted in Figures 5.6 and 5.7) can be seen in Appendix section A.7.

Table 7.19: Door events scenario 2.

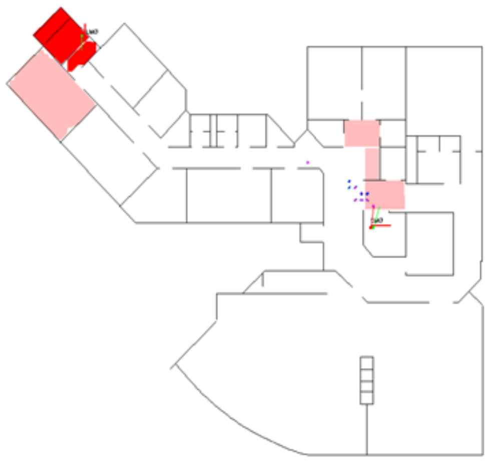
Door number	Time [s]	Action
Door 24	24	Closes
Door 23	29	Closes
Door 22	55	Closes
Door 21	67	Closes
Door 14	75	Closes
Door 13	96	Closes
Door 25 (FF)	**108**	**Closes**
Door 13	115	Opens
Door 14	133	Opens
Door 21	140	Opens
Door 22	154	Opens
Door 23	183	Opens
Door 24	201	Opens
Door 1	267	Closes
Door 2	286	Closes
Door 3	316	Closes
Door 5	328	Closes
Door 11	336	Closes
Door 12	357	Closes
Door 26 (GF)	**370**	**Closes**
Door 12	377	Opens
Door 11	397	Opens
Door 5	403	Opens
Door 3	417	Opens
Door 2	448	Opens
Door 1	466	Opens

Table 7.20: Property output for the warden at the end of the simulation for scenario 1 coupled.

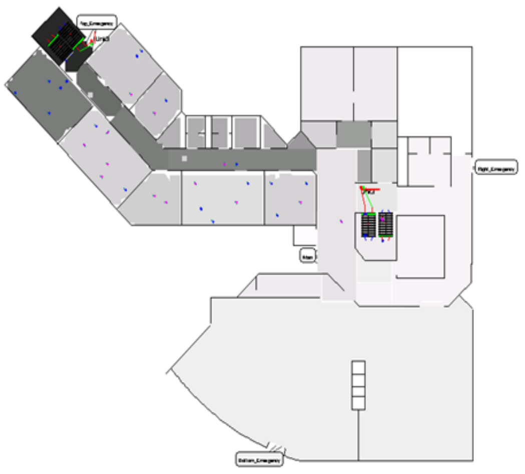
Person	FIN	FIH	FIHc	FIHr
Warden	0.00240	0.92444	0.0	0.92444



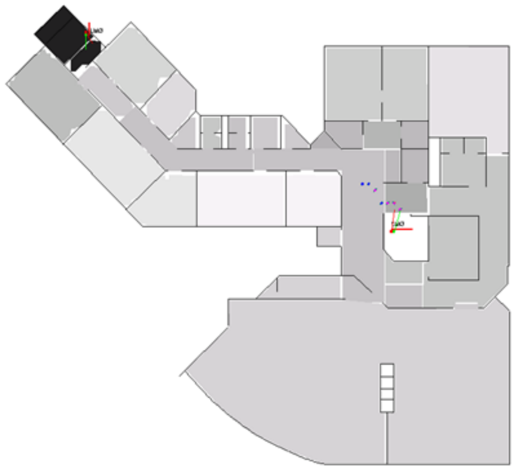
(a) Ground floor temperature output.



(b) First floor temperature output.



(c) Ground floor smoke output.



(d) First floor smoke output.

Figure 7.30: Exodus temperature and smoke outputs for both floors for scenario 2.

Scenario 3 Results

In Scenario 3 two wardens have been allocated, one on each floor. The ground floor warden also starts off in the in the same location as in scenario 1 and scenario 2, the warden upstairs starts in the same location just one floor higher up. The wardens execute the same tasks, closing doors and informing people on their floors respectively.

Table 7.21 shows the times when the wardens close and re-open the doors, Door 26 and Door 25 (in bold) are the doors to the staircase with the hazard in it. The effects the environment has on the wardens can be seen in Table 7.22 and Table 7.23. The wardens final FIH and FIHr levels are 0.0. The low readings are a direct result of the early door events at 54 seconds and 70 seconds, the fire didn't have enough time to develop and reach the high levels of the previous scenarios.

The smoke and temperature distribution for scenario 3 can be seen in Figure 7.31, Figure (a) and Figure (b) show how the closed doors shield of the staircase causing the temperature throughout the building to stay low. The same can be seen for the smoke distribution in Figure (c) and (d). The smoke density, which is directly related to the fire output is also much lower causing the two wardens a lot less problems travelling through the corridors. The first person also leaves the building at 66 seconds, the last one at 251 seconds.

It can also seen that in comparison to scenario 1 and scenario 2 that the total evacuation times decreased by around 45 per cent from 458 seconds to 251 seconds for scenario 1 and by around 57 per cent from 582 seconds to 251 seconds for scenario 2. This can be explained simply by the fact that the two wardens execute their tasks simultaneously, saving valuable time to securely exit the building. The combination of these simultaneous executions of opening and closing doors and informing people represents a complex scenario which would be very difficult to implement accurately without using a two-way coupled simulation model.

Detailed information about temperature, smoke and radiation values versus simulation time for zones 84 to 93 (measuring probes highlighted in Figures 5.6 and 5.7) can be seen in Appendix section A.7.

Table 7.21: Door events scenario 3.

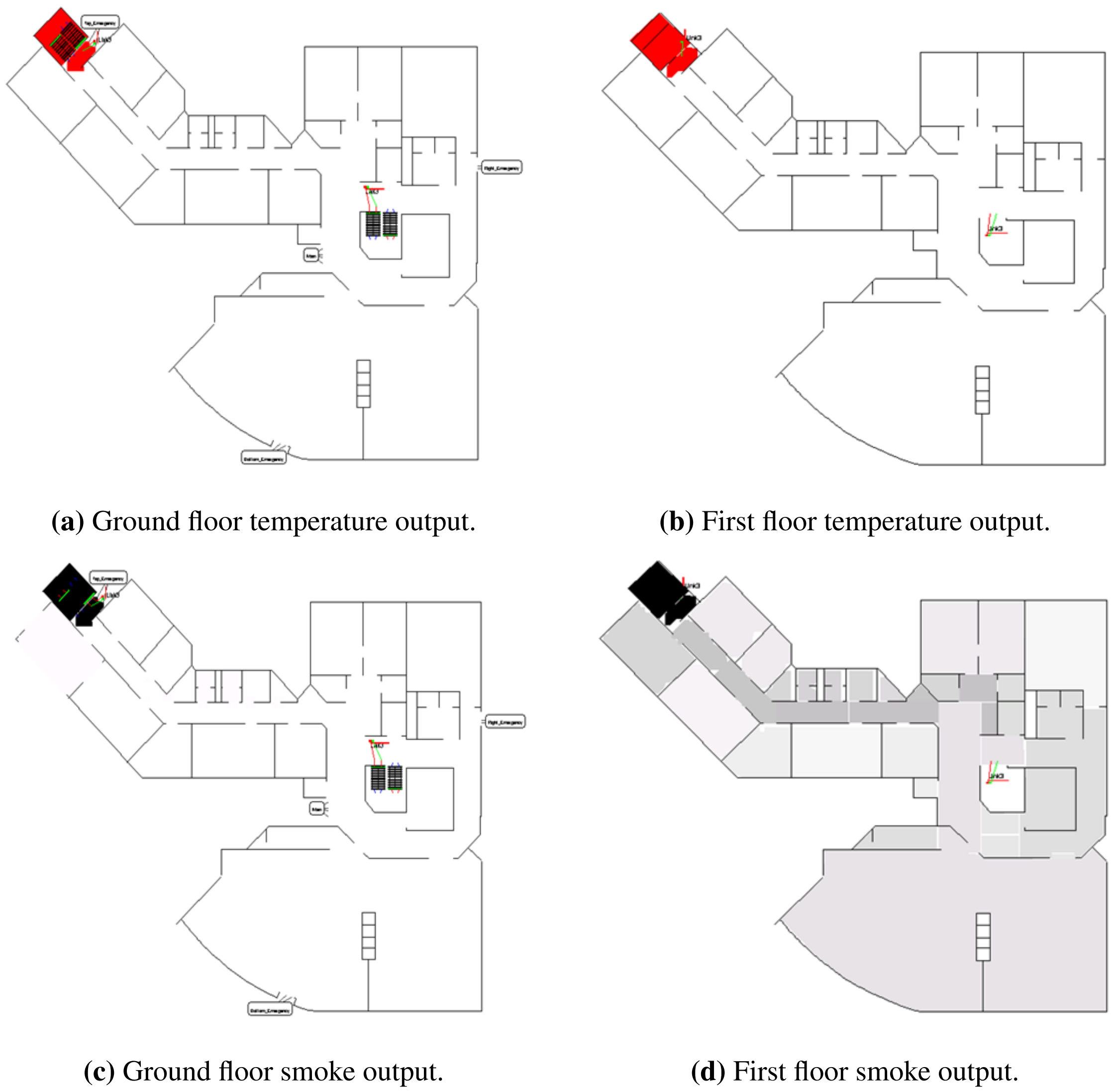
Door number	Time [s]	Action
Door 24	12	Closes
Door 1	13	Closes
Door 23	17	Closes
Door 2	18	Closes
Door 22	24	Closes
Door 3	25	Closes
Door 5	28	Closes
Door 11	31	Closes
Door 21	32	Closes
Door 14	39	Closes
Door 12	41	Closes
Door 26 (GF)	**54**	**Closes**
Door 13	59	Closes
Door 12	62	Opens
Door 11	68	Opens
Door 5	69	Opens
Door 25 (FF)	**70**	**Closes**
Door 3	75	Opens
Door 13	78	Opens
Door 2	83	Opens
Door 14	86	Opens
Door 21	89	Opens
Door 1	89	Opens
Door 22	92	Opens
Door 23	104	Opens
Door 24	110	Opens

Table 7.22: Property output for the ground floor warden at the end of the simulation for scenario 3 coupled.

Person	FIN	FIH	FIHc	FIHr
Warden	0.00049078	0.0	0.0	0.0

Table 7.23: Property output for the first floor warden at the end of the simulation for scenario 3 coupled.

Person	FIN	FIH	FIHc	FIHr
Warden	0.00059830	0.0	0.0	0.0



(a) Ground floor temperature output.

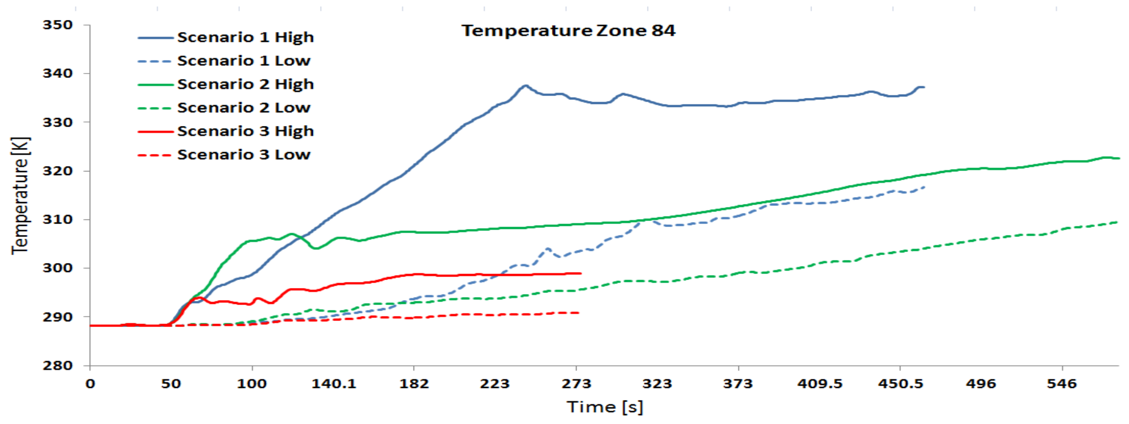
(b) First floor temperature output.

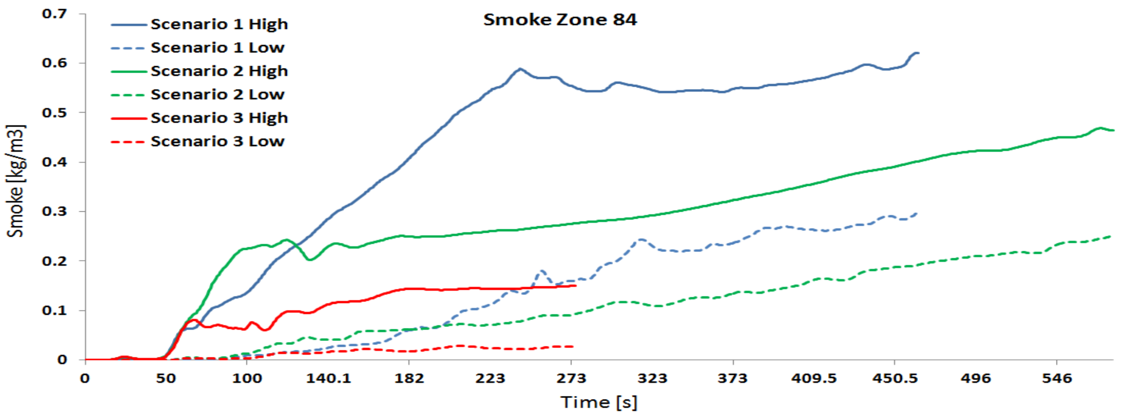
(c) Ground floor smoke output.

(d) First floor smoke output.

Figure 7.31: Exodus temperature and smoke outputs for both floors for scenario 3.

Combined Zone Outputs





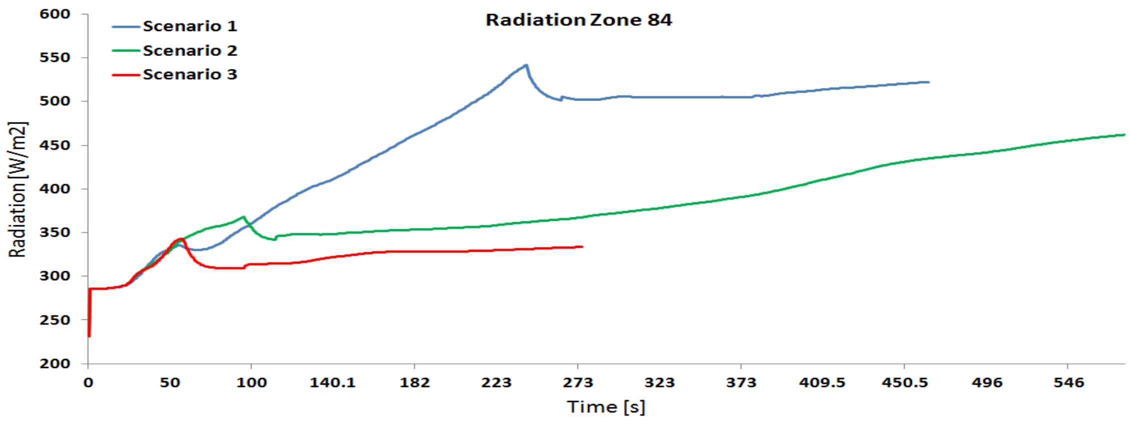
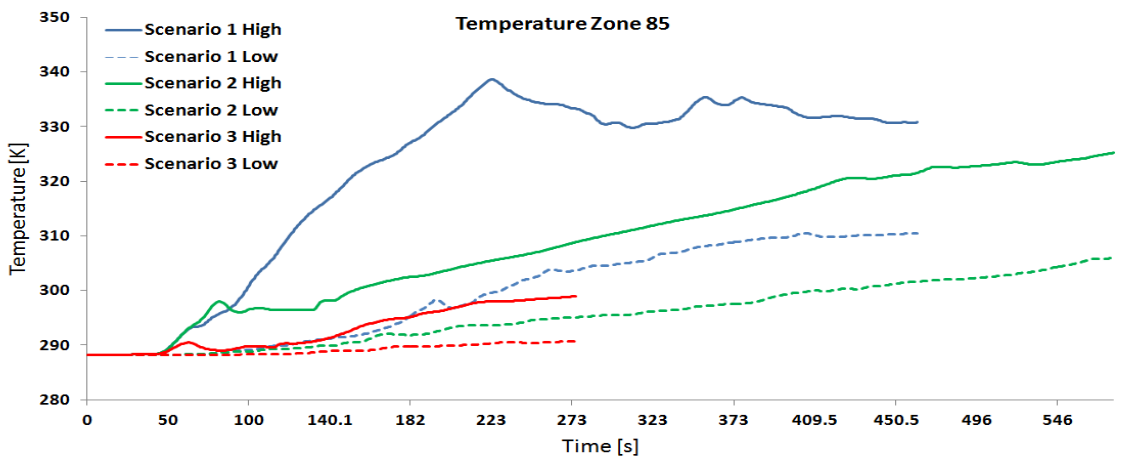
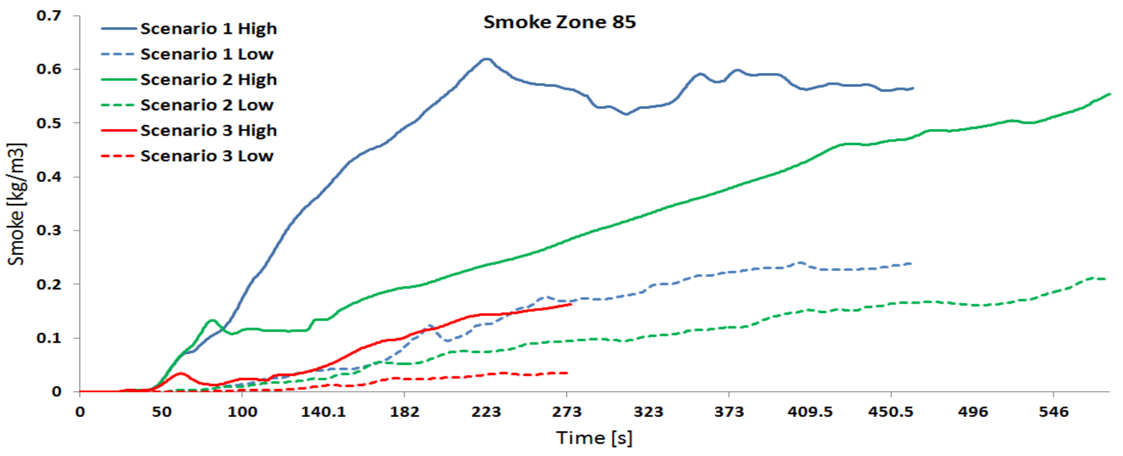


Figure 7.32: Temperature, smoke and radiation vs. time for zone 84.





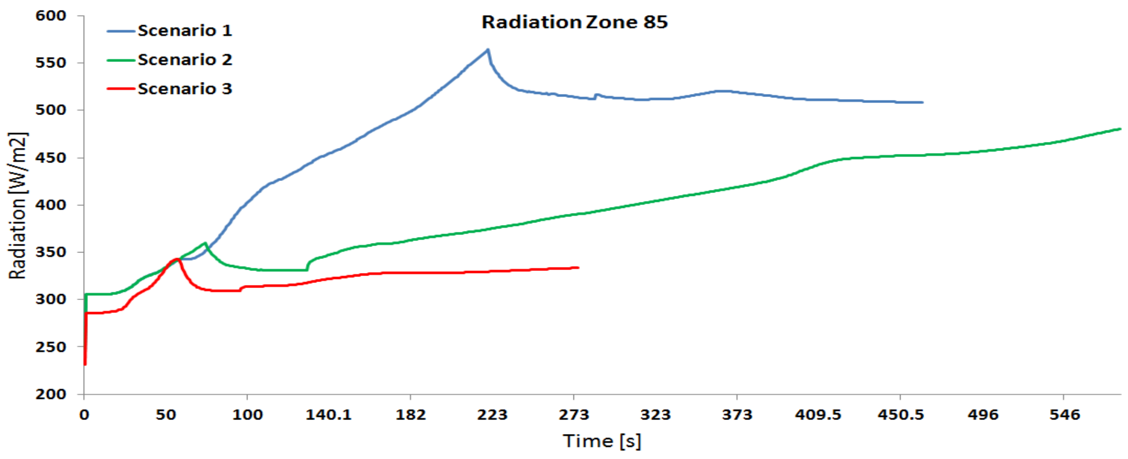


Figure 7.33: Temperature, smoke and radiation vs. time for zone 85.

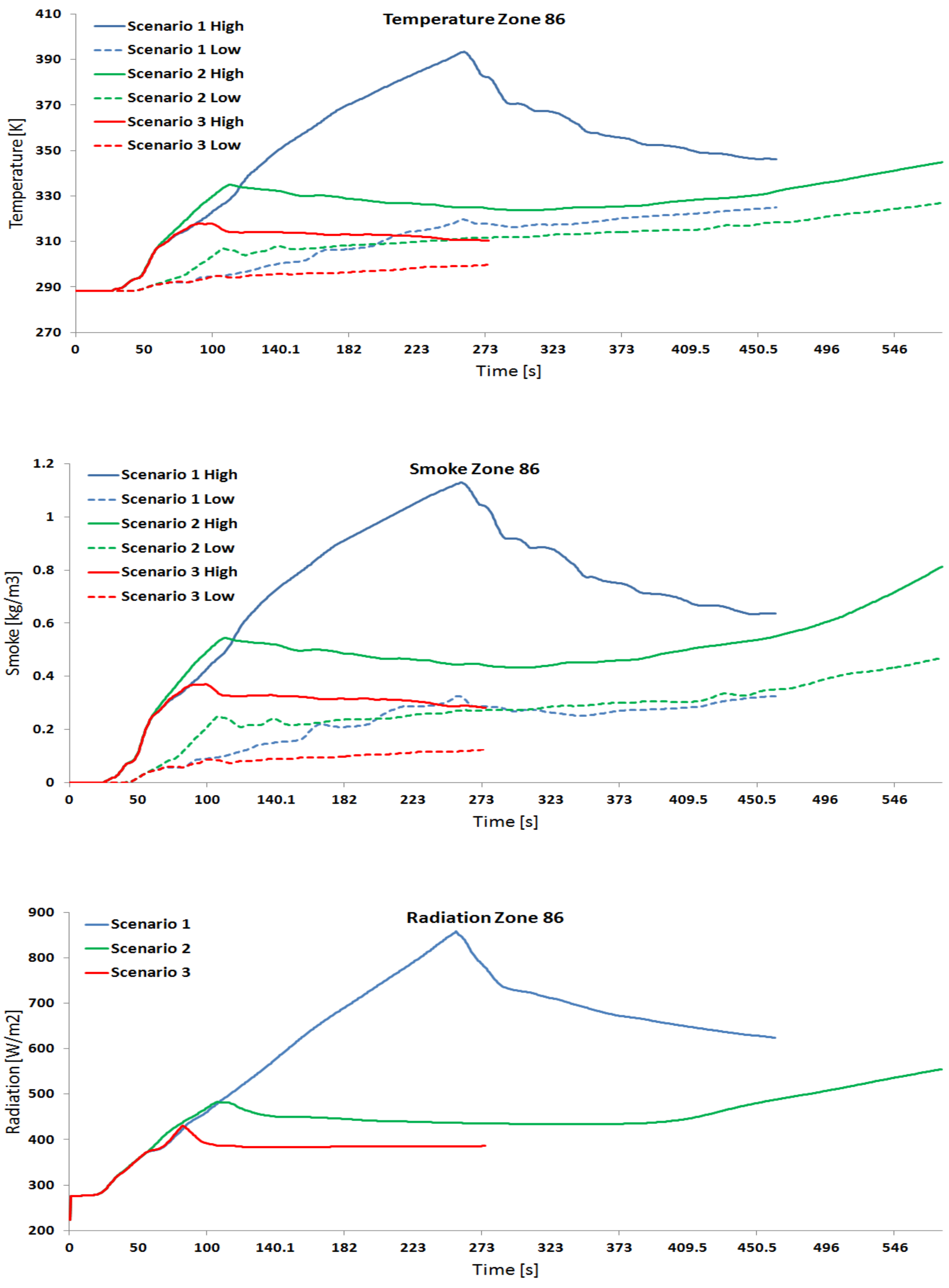
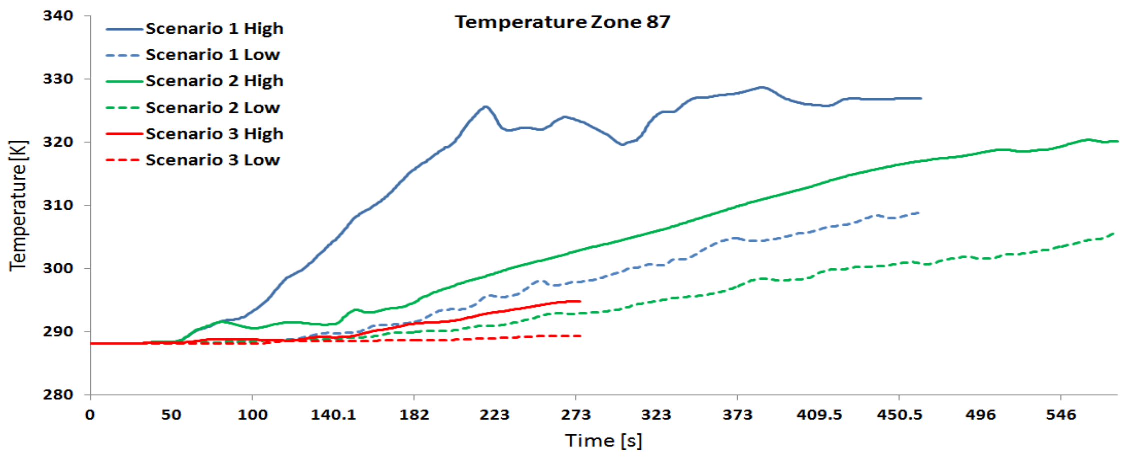
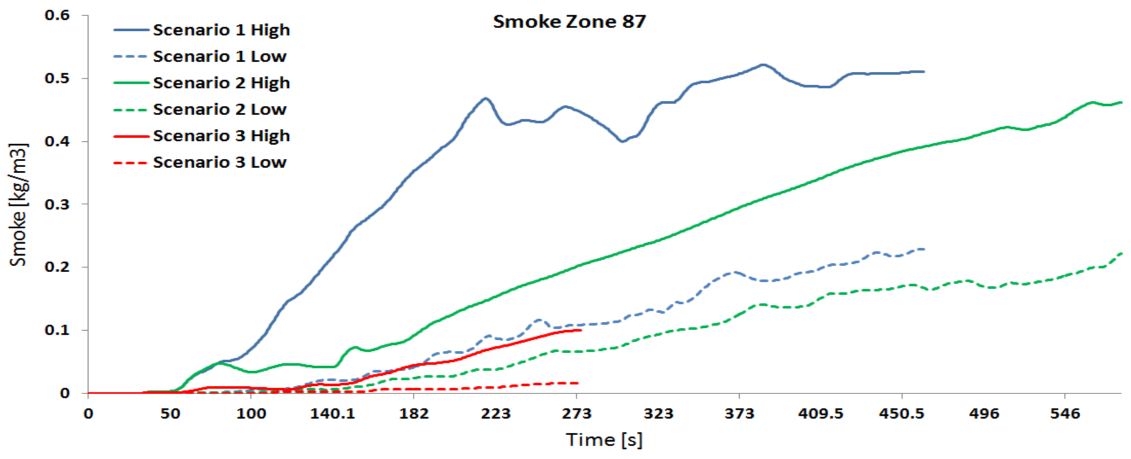


Figure 7.34: Temperature, smoke and radiation vs. time for zone 86.





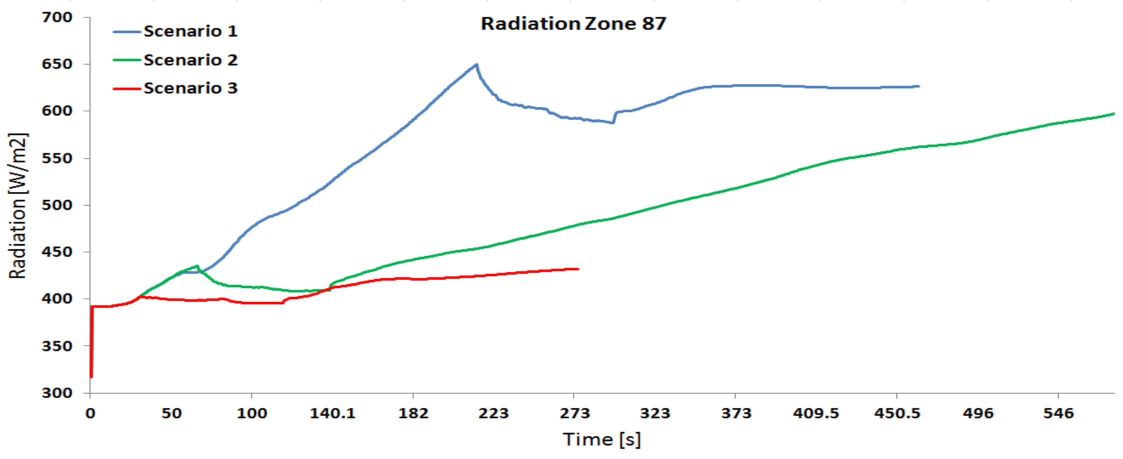
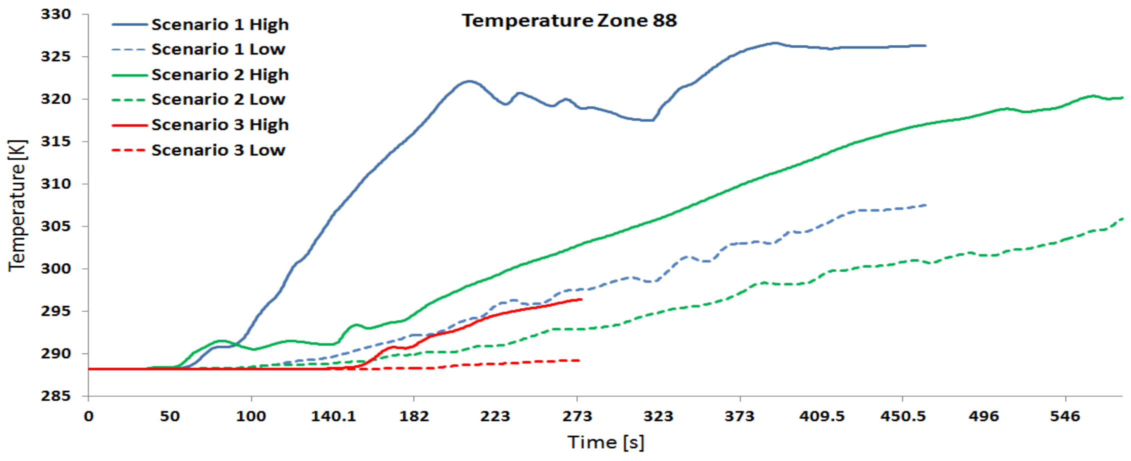
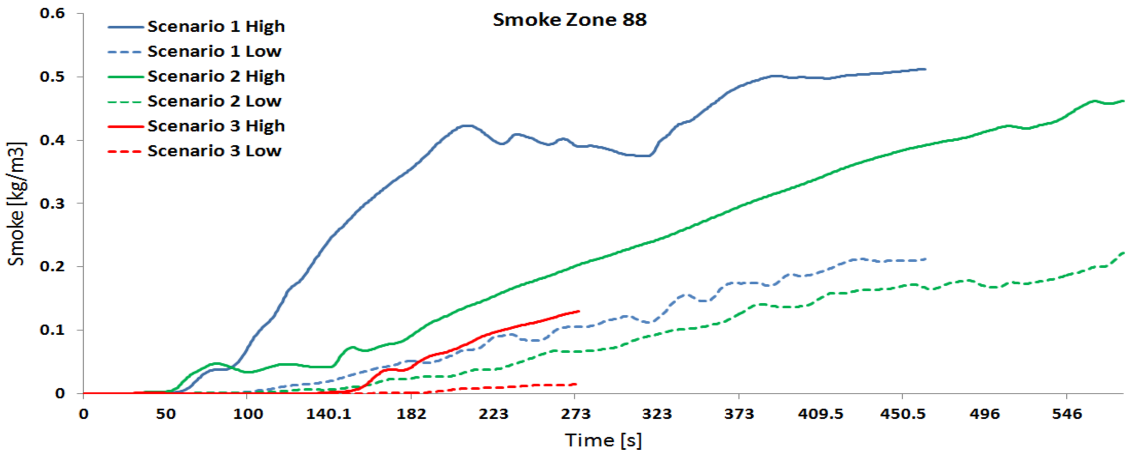


Figure 7.35: Temperature, smoke and radiation vs. time for zone 87.





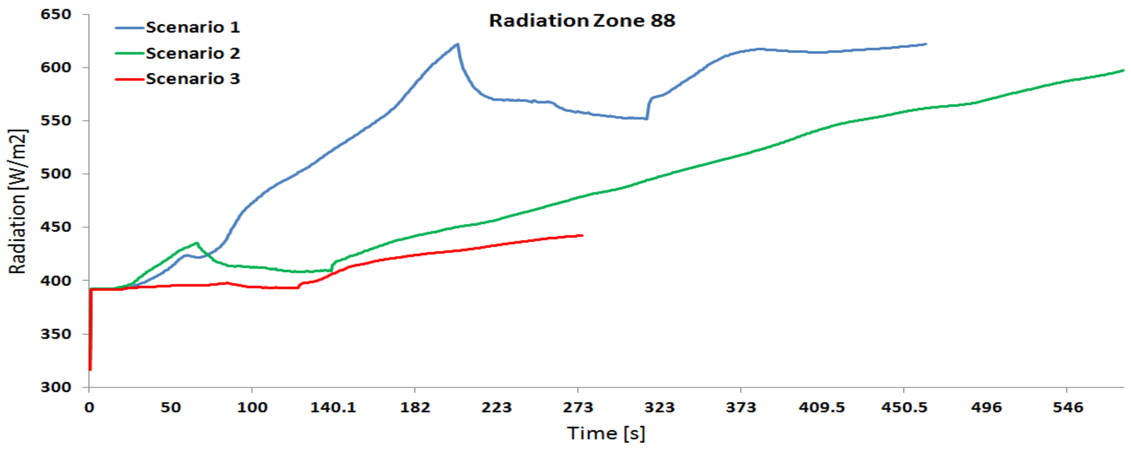
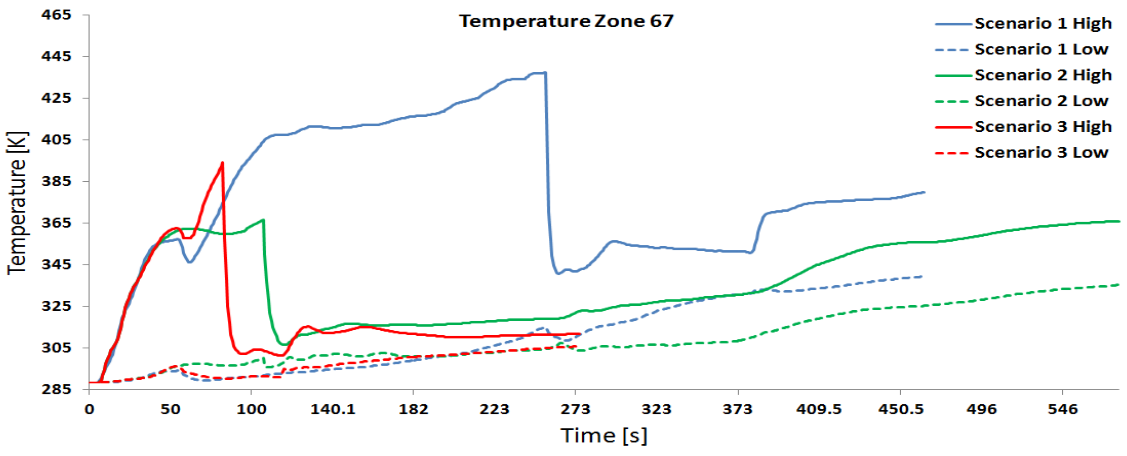
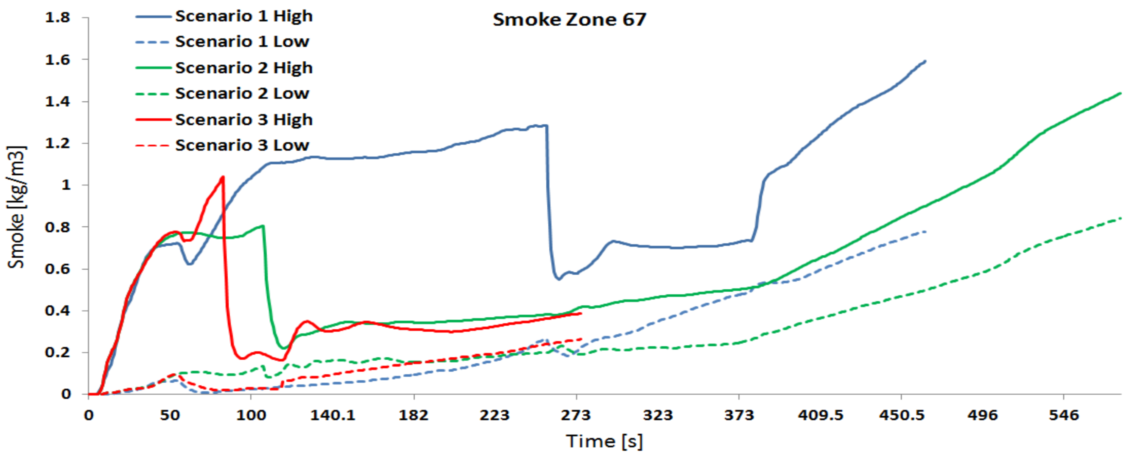


Figure 7.36: Temperature, smoke and radiation vs. time for zone 88.





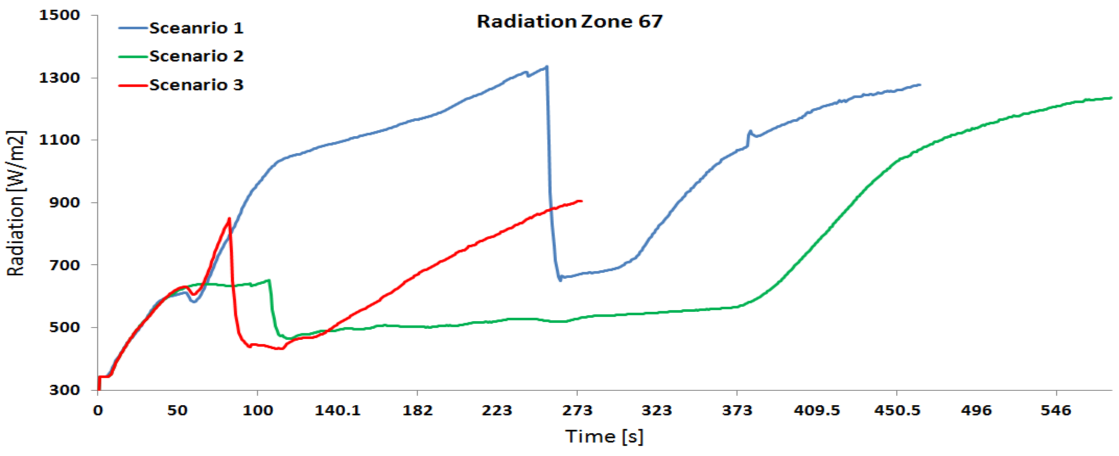


Figure 7.37: Temperature, smoke and radiation vs. time for zone 67.

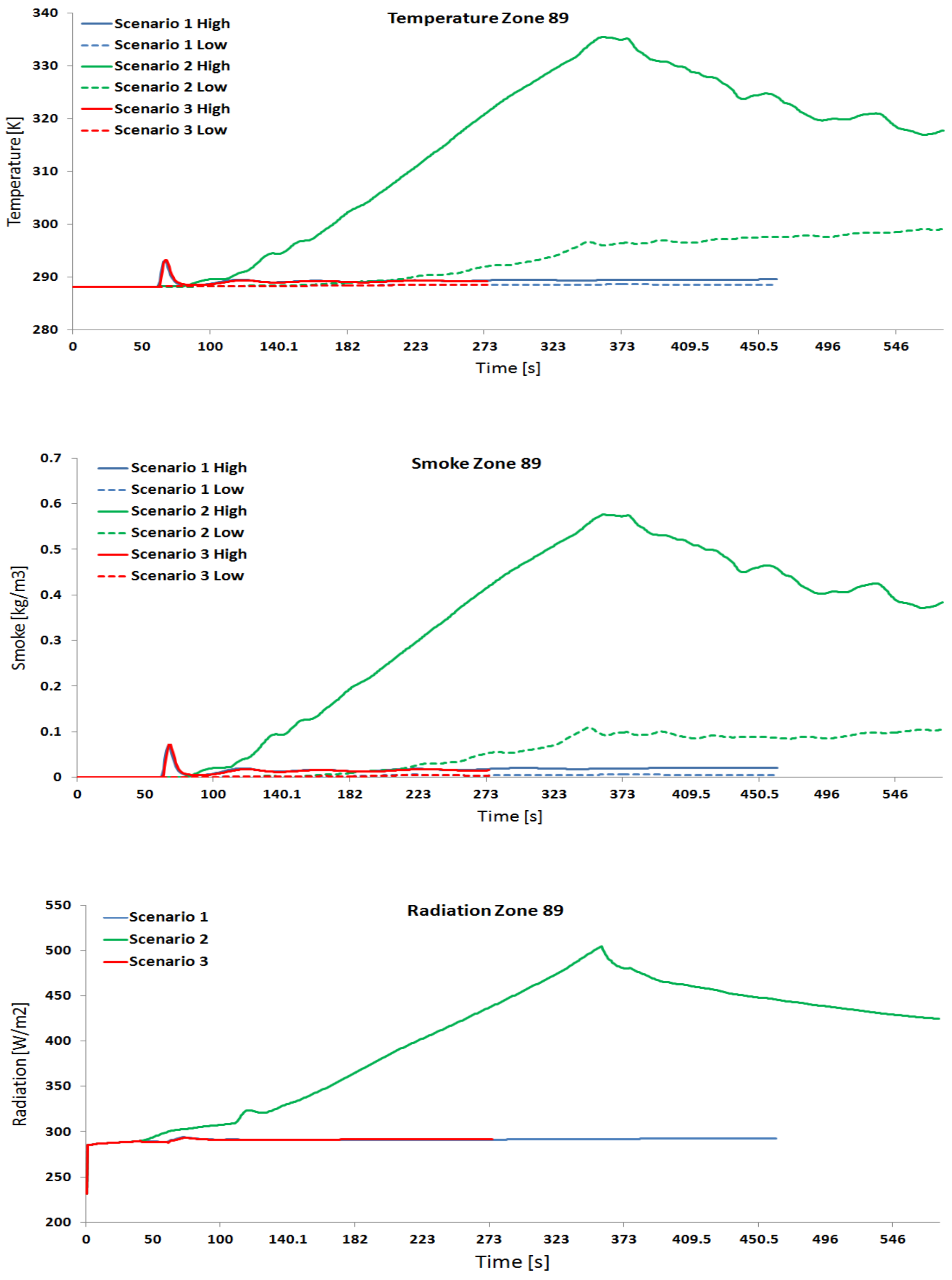
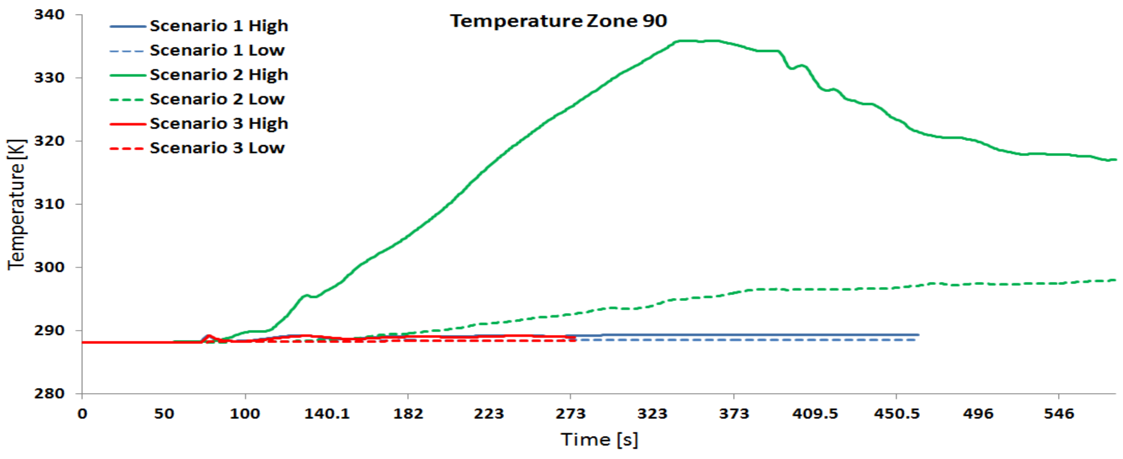
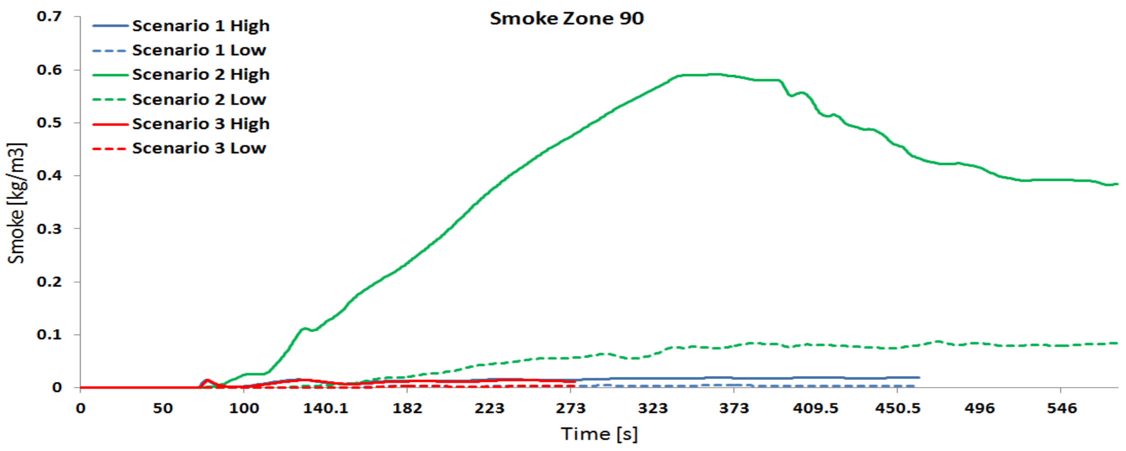


Figure 7.38: Temperature, smoke and radiation vs. time for zone 89.





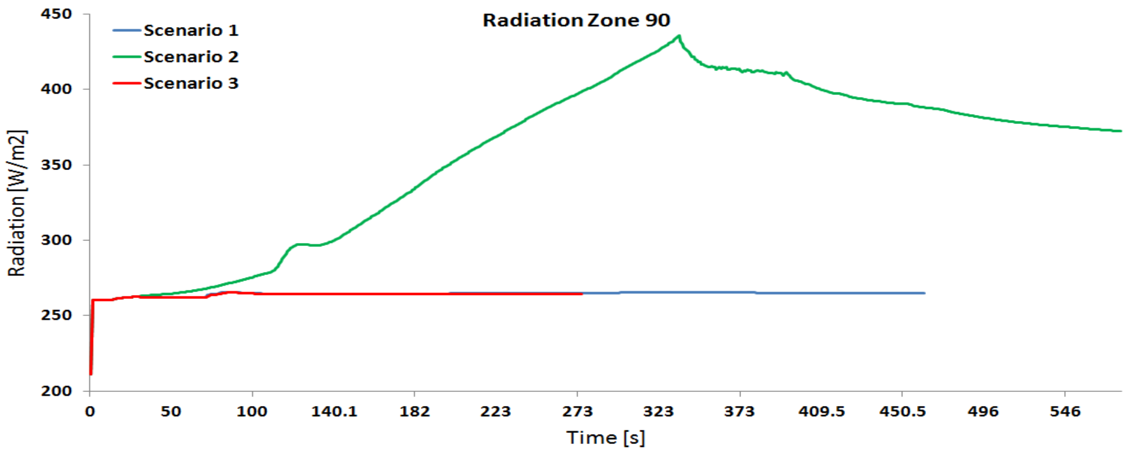


Figure 7.39: Temperature, smoke and radiation vs. time for zone 90.

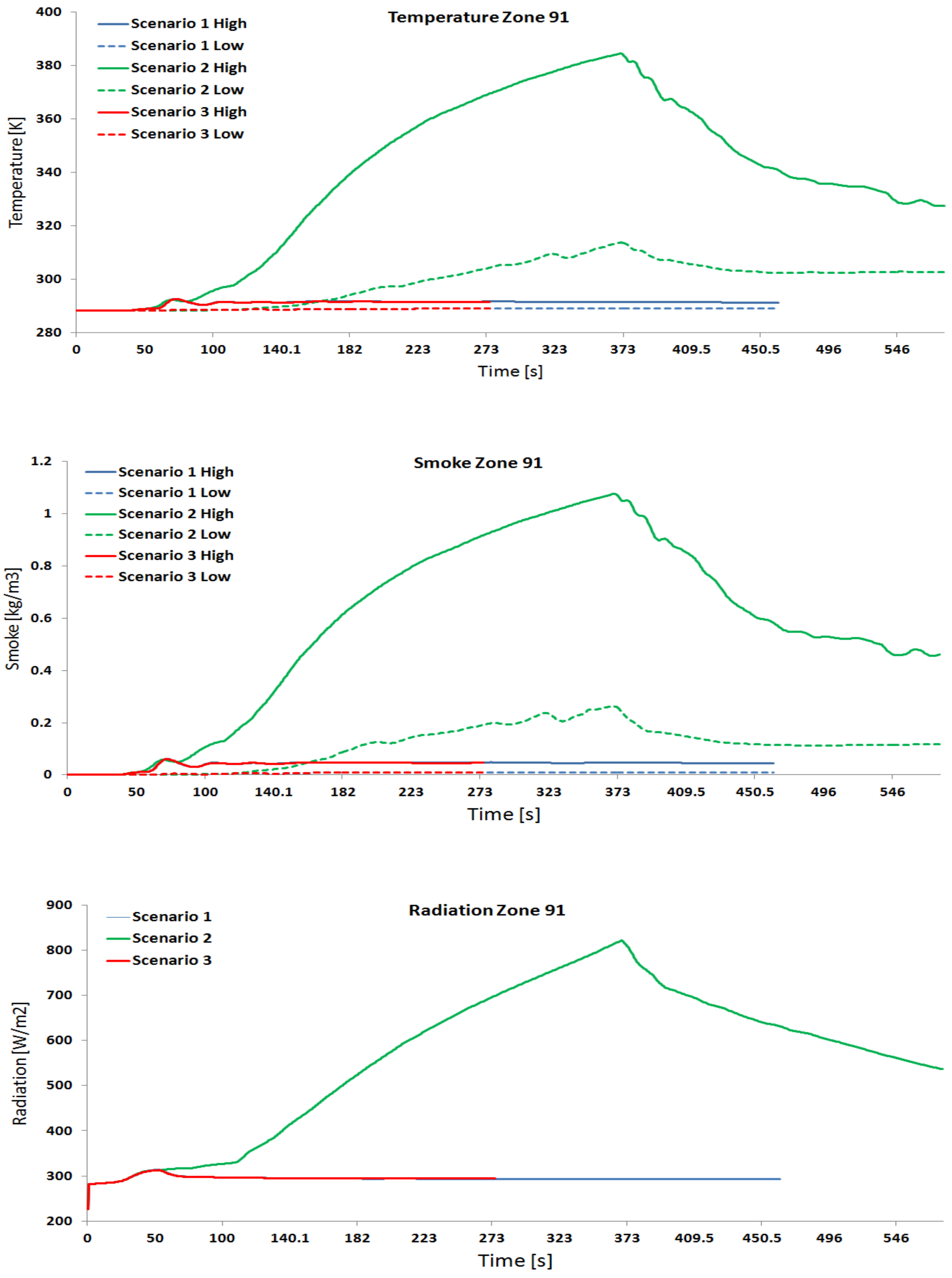
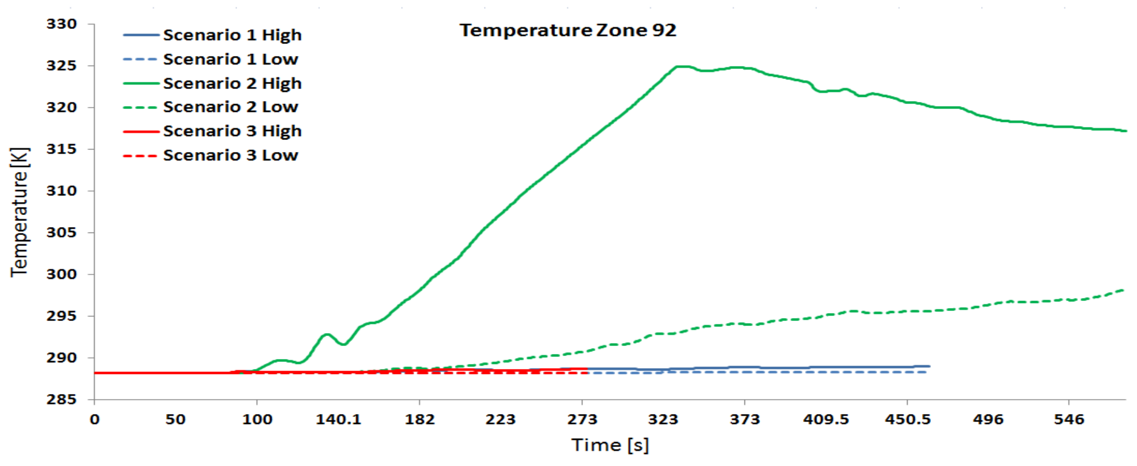
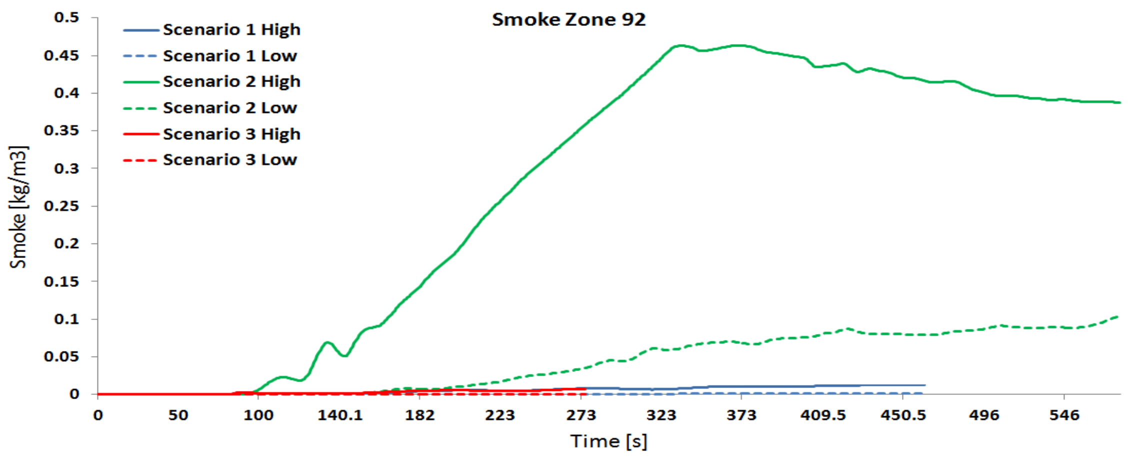


Figure 7.40: Temperature, smoke and radiation vs. time for zone 91.





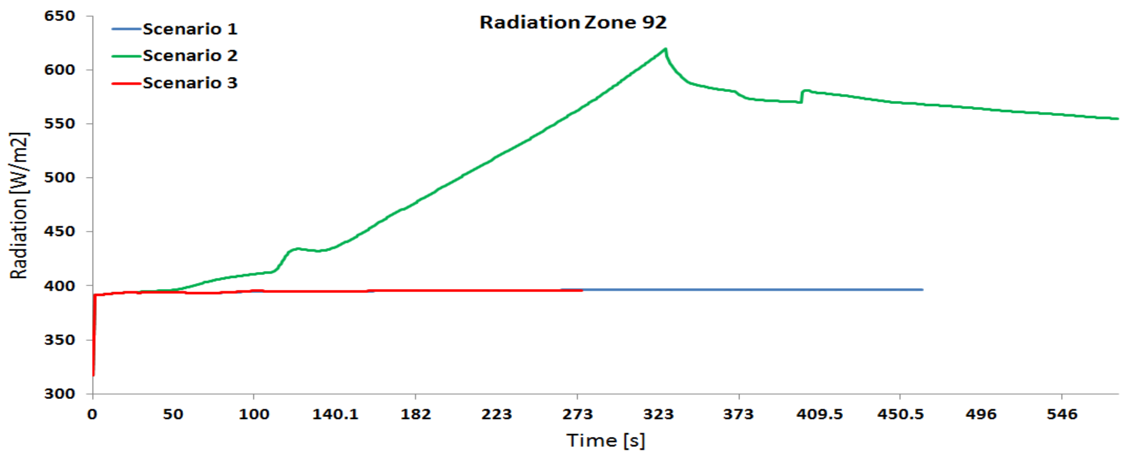
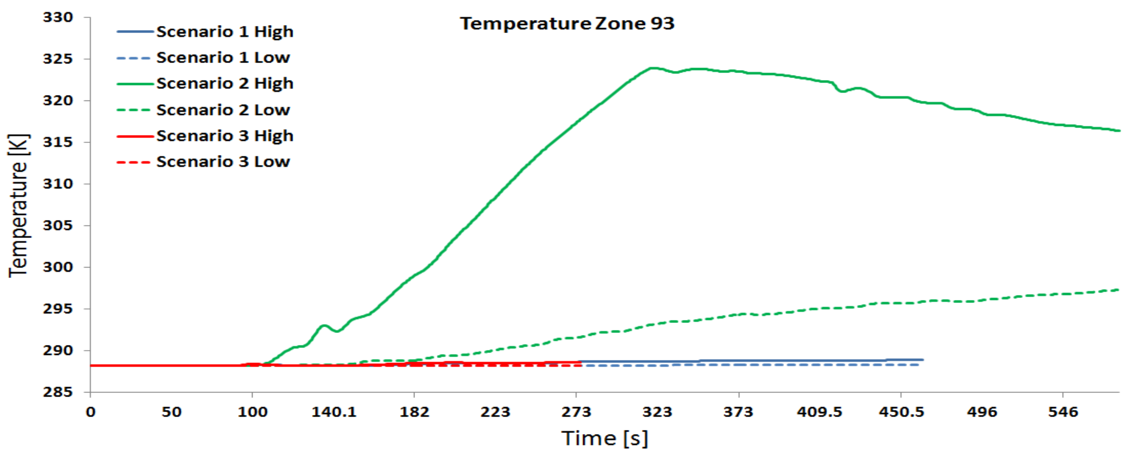
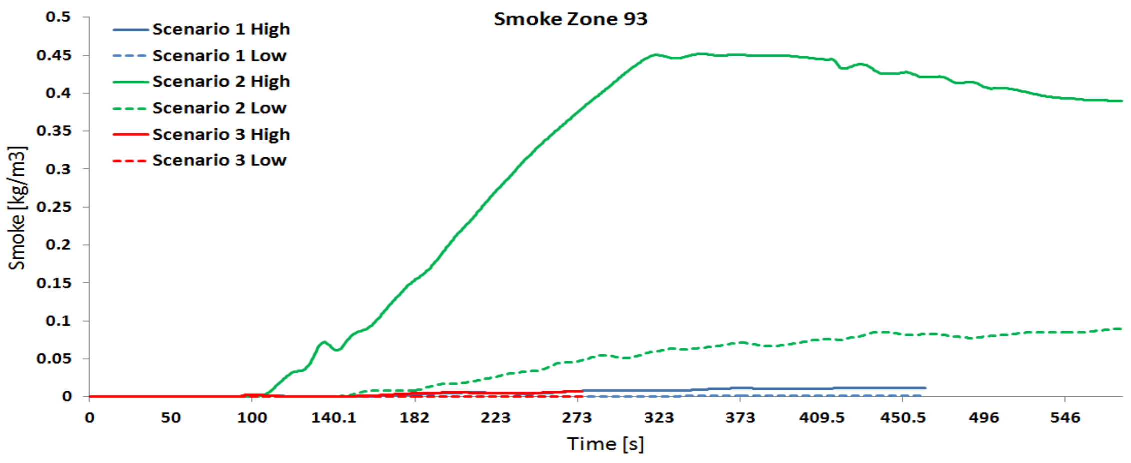


Figure 7.41: Temperature, smoke and radiation vs. time for zone 92.





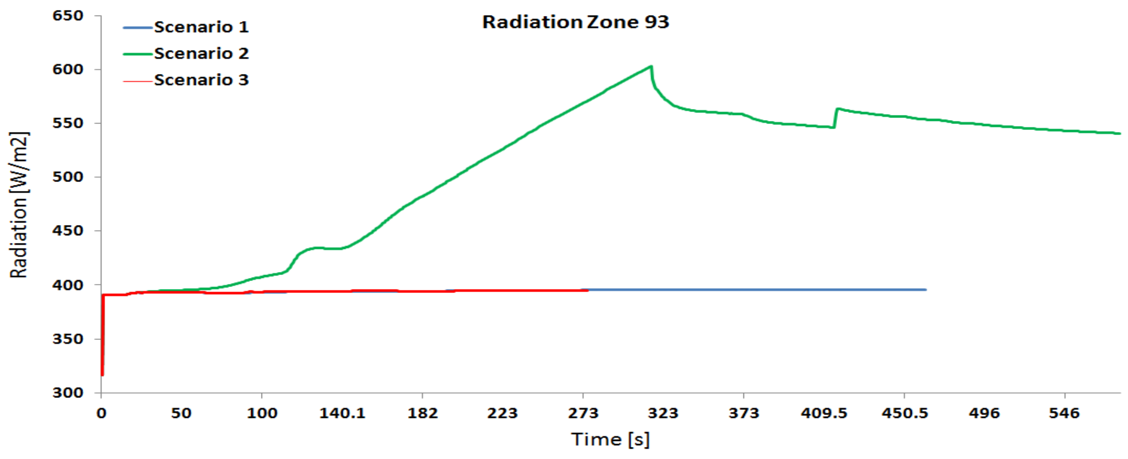
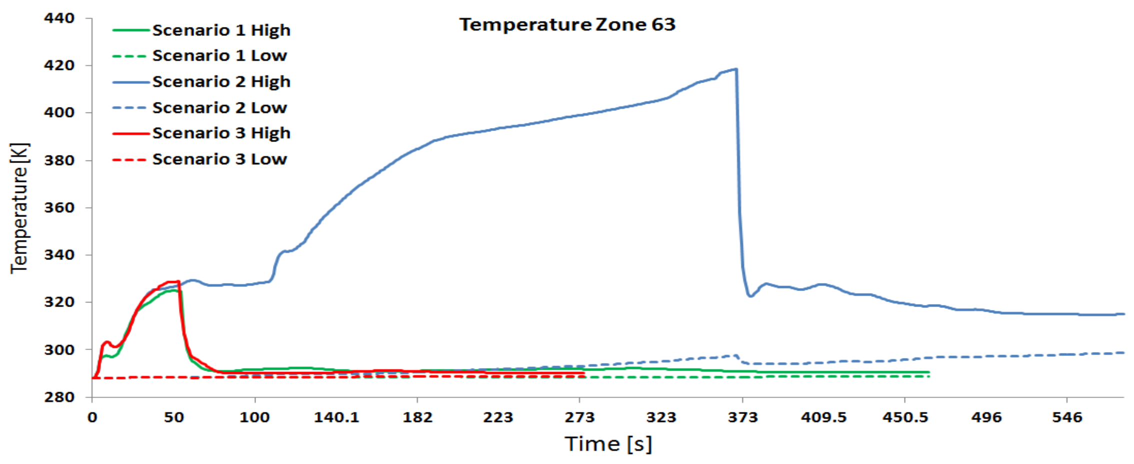
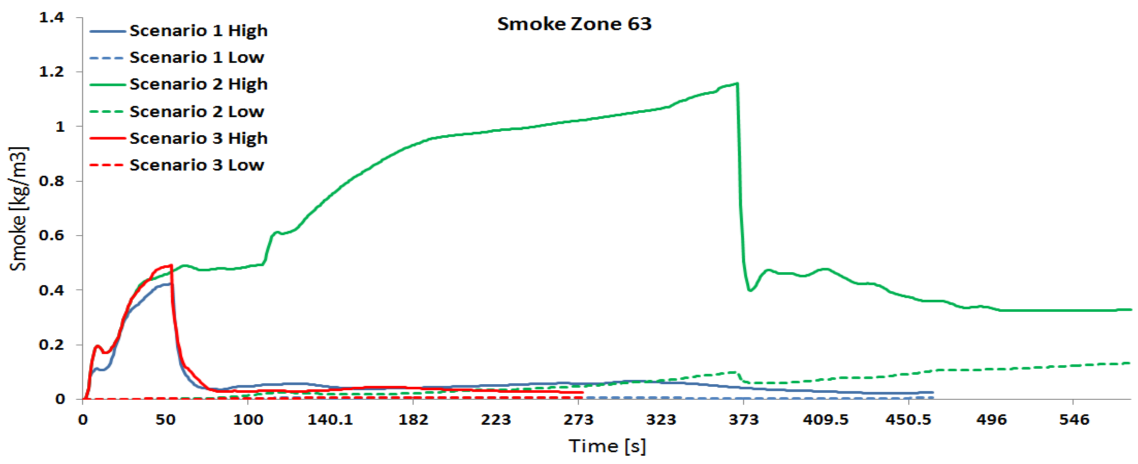


Figure 7.42: Temperature, smoke and radiation vs. time for zone 93.





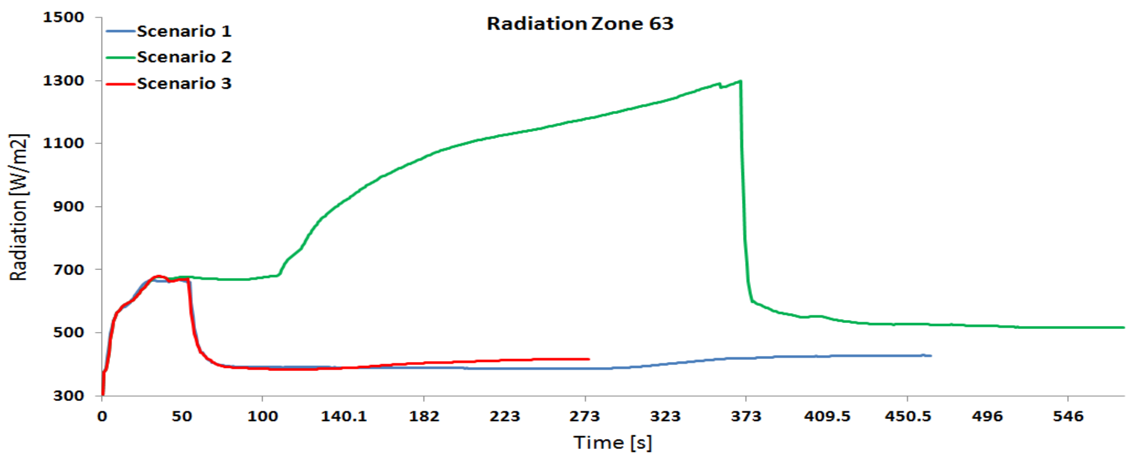


Figure 7.43: Temperature, smoke and radiation vs. time for zone 63.

Chapter 8

CONCLUSION

This chapter will sum up the findings and review how the thesis has addressed the initial research questions from Chapter 1. The conclusions can be found in more detail in the previous chapters, the conclusions here will be of a more general nature and provide a summary.
Building upon the already existing SMARTFIRE numerical CFD model a parallel GPU extension has been implemented. This extension consists of a graph partitioning routine which handles how the geometry is split up, based on certain rules to optimise the surface between the individual domains within the hardware boundaries of the GPU being used. Furthermore a complete rewrite of the coefficient building routines and the solver has been implemented to accommodate the completely different execution structures used on the GPU. This required 50.000 lines of code to be written. The GPU code has been embedded into the serial code in order to use the other parts of the code which do not benefit from being ported to a parallel version, i.e meshing routines, geometry parsing, zone data outputs and visualisation routines.

In addition to the changes in the CFD code a binary message parsing system has been implemented to enable the communication between SMARTFIRE and buildingEXODUS at the end of each time step. This two-way coupling offers new possibilities for fire safety engineers to examine more realistic scenarios in a reasonable amount of time. The novelty of this coupling lies within its ability to communicate bi-directional, from SMARTFIRE to EXO-

DUS (update environmental values) and vice versa (controlling changes in the CFD code), so it becomes possible to feed back decision that agents take in EXODUS, like opening doors, to the CFD code in almost real time (lag of 1 time step). The CFD code dynamically updates it's boundary conditions and continues calculating and solving the geometry.

The aim was to provide the fire safety industry with a new simulation tool which reduces the number of assumptions an engineer has to make in order to model a scenario. The engineer will not have to assume times any more when he thinks an agent will reach certain destinations. These times will then have to be used in order to set up and calculate the CFD scenarios. At the end the CFD results will get fed back into the evacuation simulation. These techniques get very error prone for bigger, complex scenarios. It has been shown that environmental factors like heat, radiation and smoke have to be taken into account to accurately determine how agents behave during the simulation. Smoke for example reduces the walking speeds drastically adding sometimes minutes to the predicted time when an agent would reach a destination. The two way coupling eliminates the need of these assumptions as the agent only needs a predefined task e.g open door X, the conditions which he faces on his way to door X determine the time it will take him to get there. When he finally arrives the opening event for door X triggers a boundary update in the CFD code which from the following time step on will treat the door as open and therefore adjust the temperatures/ radiation and smoke levels accordingly. Test cases have shown that this can have an huge impact on the outcomes of even simple scenarios, the bigger and more complex scenarios get the more complicated the flow dynamics get as well, so can for example a closed/ opened door or window change the whole ventilation characteristics of a geometry.

Furthermore a new parallel GPU version has been provided which uses state of the art technology and will offer more opportunities in the future to speed up simulations, which will directly reduce costs and enable the simulation of more complex scenarios.

Test cases

The selected test cases highlight three main aspects of the newly introduced version:

- correctness of results against the serial version

- execution speed compared to the serial version
- differences in the evacuation outcomes due to the two way coupling

The benchmark process was split into different cases, the well documented Steckler room case [46; 171] to demonstrate the numerical results of GPU and serial version against the experimental results, a very simplified test case to show the capabilities and differences the two-way coupling can make and a complex, two storey geometry that shows the outcomes for a realistic scenario. The benchmark case for the correctness of results was purely focused on the numerical results and evaluates the numerical results on an individual cell bases against the experimental data of the original experiment and the numerical results which FSEG has previously computed using the serial version of SMARTFIRE.

A simplified geometry was used to show how the basic concept of the coupling interacts with buildingEXODUS based on different evacuation strategies. Big difference in outcomes could already be demonstrated by executing simplified tasks.

Furthermore the coupling has been compared to traditional approaches which a fire safety engineer would have tried in order to model these scenarios using the non coupled tools available. The outcomes showed that the results consistently differed from the fully coupled version in a way that the impact of the environmental factors was underestimated resulting in a lot less significant outcomes. These outcomes varied from big differences in overall simulation times to cases where the complexity of the model couldn't be accurately represented without using the coupling. Especially in cases where more than a single event happens and a series of manually prepared and executed CFD simulations would have to be applied in order to incorporate every geometry change.

The two way coupling showed great potential to be accurately and cost-effectively used in fire safety engineering. It also highlighted parts of the model which did not fully benefit from the new hardware approach and will give a good guidance to which parts of a CFD model give the biggest benefit if re-writing the full model is not an option.

The coupled fire and evacuation model version of SMARTFIRE and EXODUS comes with pros and cons, listed below:

Pros

- eliminates the need to predefine factors in the model
- more realistic scenarios
- more complex scenarios can be modelled
- foundation for other dynamic changes e.g structural changes
- more interaction of agents with the geometry/ environment
- only one CFD simulation scenario set up required

Cons

- added complexity to set up cases for both CFD and evacuation model
- requires more computation as no assumptions made any more

Looking at the GPU CFD fire model as a standalone version it has also several pros and cons, which are listed below:

Pros

- execution speed
- only one machine required to achieve high speed-ups
- more complex scenarios can be modelled
- energy efficient

Cons

- memory limitations
- limited output capabilities
- only runs on NVIDIA hardware

The main disadvantage is first of all that one is dependent on NVIDIA continuing the CUDA programming model and their hardware range. Although as mentioned in Section 2.4.2.1 there are alternatives but at the time this research was conducted there was no real alternative present. As it looks today it is highly unlikely that NVIDIA will discontinue the technology as a lot of mainstream software manufacturer now provide CUDA versions of their products which offer much higher performance for very little hardware investments. Nevertheless the big hurdle will always remain the complexity of the programming model and the underlying problem, as there will not be a ”one solution fits all” solution. The CUDA SMARTFIRE version tried to eliminate as many bottlenecks i.e memory transfers, as possible at the cost of rewriting a lot of code that doesn’t directly gain further speed-ups from the parallel implementation. The other big disadvantage that will always be present is lack of input/ output properties, although NVIDIA has introduced a unified memory concept recently it goes against the parallel concept and will always resemble less efficient implementation. It removes the manual allocation of memory and the subsequent memory copies in order to get data onto the GPU. The unified memory concept allows users therefore to directly write code as they would do in a traditional host programming language, but behind the scenes the API will still allocate and copy memory onto the GPU. Currently NVIDIA provides hardware with memory sizes of multiples of 6 Gigabytes, all higher memory sizes are a combination of 6 Gigabyte blocks and must be treated as basically two separate pieces of hardware. If a geometry requires more memory space the only way around this will be to introduce a multi-GPU version of the code. This multi-GPU code will face the same problems as the single-GPU version and will not necessarily provide any further speed ups. The problem still remains the communication between computational domains and in this case even across different GPUs. Anyway the advantages using GPU for parallel number crunching outweigh the disadvantages especially as the achieved speed gains can be achieved on a single machine now instead of using a network of machines.

Chapter 9

FUTURE WORK

Although much has been achieved and a complete, fully working CFD model has been implemented, there is still more that can be implemented and improved in the future. This chapter, will give some ideas for further work that can be done. Some of these ideas will be carried through in the near future and others may or may not happen depending on circumstances.

CFD model extensions

Although, as mentioned above, a complete CFD model has been developed, one important part for the fire safety research community is missing which the serial SMARTFIRE version already offers:

- Gaseous combustion
- Toxicity
- HCN
- HCL
- Sprinkler models

The current GPU SMARTFIRE CFD model uses temperature, radiation and smoke outputs to influence the behaviour and properties of the agents in buildingEXODUS. The temperature and radiation will have a direct effect on the health of an agent and will kill him if the exposure is too high for too long. The temperature will reach high levels in the areas directly surrounding the fire source, the radiation has a bigger extension area but overall in big geometries, the majority of the agents might be quite far away from the fire source and therefore only experience raised temperatures. Where it is unlikely that a few degrees raised temperature cause any serious risk for the agents well-being. The radiation has an effect through FIHc and FIHr, the cumulative to convective heat and cumulative exposure to radiative heat as mentioned in Section 7.1. Radiation has a bigger effect on the agents but they still need to be fairly close to the heat source for the radiation to become life threatening. The third output, smoke, only effects the visibility and therefore the travel speeds of the agents, indirectly hurting them by eventually exposing them to the fire source for longer.
In real fire scenarios, the majority of injuries comes from inhaling toxic products as CO (Carbon Monoxide), HCN (Hydrogen cyanide) and HCl (Hydrogen chloride). The spread of these species is pretty similar to the smoke spread within a geometry, with the difference that even smaller doses can already cause fatalities or serious injuries. The other models will be needed to be able to model things like fire spreads and various extinction techniques (water mist, foam, etc.)
For the next version of the SMARTFIRE GPU version the current additional models mentioned above will have to be completely re-written and optimised in order to achieve the same level of accuracy as the serial version of SMARTFIRE.

Another idea for a future version of SMARTFIRE could include a model to incorporate structural changes caused by the influence of fire or extinguishing agents like water. A structural change for example could be a collapsing ceiling, wall or any change in the geometry that could change the CFD dynamic within.

Exodus extensions

Although the CFD part of the coupling has been fully rewritten, the evacuation part only had a few changes to enable the communication and basic interaction of the models. BuildingEX-

ODUS essentially only acts as the trigger for changes in SMARTFIRE. One part that will be crucial in the future for the coupling to be fully able to simulate real life scenarios is the ability of the agents to react to the newly occurring events. This means for example the warden could make the decisions himself which doors to close depending on the environment and the routes available. Or an agent might take opening windows into account to get fresh air in. These approaches will require extensive analysis of evacuation trials and the human behaviour to come up with a heuristic model.

Performance improvements

There are still some parts of the code left which will contribute to a higher overall performance. At the moment the stand-alone CFD model operates almost at peak performance on the hardware which was available for this research, but when used coupled, one of the bottlenecks mentioned earlier (see Chapter 8) becomes very obvious, the limited input/output capabilities. After each coupled time step SMARTFIRE has to send 5 values per hazard zone back to buidlingEXODUS. The values are upper and lower level temperature, upper and lower level smoke and radiation. Radiation only has one value as the nature of being a ray and the relatively small rooms it is assumed that the values do not differ much. To calculate those values averages are calculated over the cells within one zone. This process still resides in the serial version of SMARTFIRE at the moment. The CUDA version of SMARTFIRE has to copy every single cell value back to the serial code, re-map them in the original cell ordering (without the domain setup) and finally calculate the 5 float numbers per zone. It can already be seen from this description that eliminating this memory copy bottleneck will speed up the overall time again. This will be achieved by doing the zone averaging on the GPU as well, even though the actual averaging will probably not be speed up much (reduction operations across multiple domains are not optimal for parallel implementation as it cannot be achieved in one single execution and will require some kind of buffering) the time spent for the memory copy will be almost eliminated. Instead of copying GBs of memory it will be down to 5 variables * number of zones.

Usability improvements

Another useful improvement in the future will be the usability of the coupling. At the moment the coupling requires several steps to be manually executed when loading up the case. If one misses a step the process has to be restarted. The steps require to be executed independently in the right order. These steps are:

- starting SMARTFIRE CUDA version and let it load in the CFD case
- starting buildingEXDOUS and connecting it to the EXODUS server
- SMARTFIRE door initialisation for the geometry
- loading the EXODUS evacuation geometry
- applying EXODUS settings e.g enabling the hazard mode and setting the various properties for the agents

Now, as soon as the CFD simulation is started the coupling starts communicating and works independently, so no further interaction is required. The process flow can be seen in Figure 6.3 and Figure 6.4. In the future this will have to be automated so that the user only has to start one process which handles the rest of the execution.

Furthermore it has to be ensured that the EXODUS geometry and the SMARTFIRE geometry are synchronized. At the moment a user could enter different door identifiers in both geometries leading to a mismatch of events or a crash in the simulation model. This could be eliminated by e.g introducing a process which maps the SMARTFIRE ids into EXODUS.

Next generation hardware usage

The final disadvantage, mentioned in the previous chapter (see Chapter 8), is the limitation in memory which directly translates to limitations in the complexity or size of scenarios to be run. Future cases which will require much finer meshes or are much larger in nature will require an extension of the memory. Nvidia has recently introduced the ability to utilize multiple GPUs to increase the available size of memory. It has to be determined if this adds any benefits regarding further speed ups on top of the benefit of increasing the memory size. The main problems will stay the same, or even become more obvious as a second level of bottlenecks, the communication and synchronisation across GPUs will be introduced.

Case Base Reasoning System

Another way of reducing the execution time in the future could be the usage of a case base reasoning system [175; 176], where CFD simulation results will get stored for future usage. Instead of rerunning a CFD simulation for a given scenario, the case base reasoning system will be able to apply a previously run case that will meet the current criteria, e.g door changes 10 seconds into the simulation.

REFERENCES

[1] **Fire Statistics : Great Britain**. `https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/410287/Fire_Statistics_Great_Britain_2013-14___PDF_Version_.pdf`. Accessed March, 2015. 1

[2] Suhas Patankar. Numerical heat transfer and fluid flow. CRC Press, 1980. 7, 8, 58, 65, 67, 209

[3] Louis Melville Milne-Thomson. Theoretical aerodynamics. Courier Dover Publications, 1966. 7

[4] Wan Ki Chow. **A comparison of the use of fire zone and field models for simulating atrium smoke-filling processes**. Fire Safety Journal, **25**(4):337–353, 1995. 8

[5] Edwin Galea. **On the field modelling approach to the simulation of enclosure fires**. Journal of Fire Protection Engineering, **1**(1):11–22, 1989. 8

[6] JP Boris, FF Grinstein, ES Oran, and RL Kolbe. **New insights into large eddy simulation**. Fluid dynamics research, **10**(4-6):199, 1992. 8

[7] HI Rosten, DB Spalding, and DG Tatchell. **PHOENICS: a general-purpose program for fluid-flow, heat-transfer and chemical-reaction processes**. In Engineering Software III, pages 639–655. Springer, 1983. 8

[8] Nikos C Markatos and G Cox. Hydrodynamics and heat transfer in enclosures containing a fire source, **5**. CHAM Limited, 1983.

[9] GEOFFREY COX. Combustion fundamentals of fire. Acad. press London etc, 1995.

[10] F JIA, ER GALEA, AND MK PATEL. **Simulating FLASHOVER and BACKDRAFT Type Events Using Fire Field ModelsA First Approximation**. Journal of Fire Protection Engineering, **9**(4):1–17, 1999.

[11] PA RUBINI. **SOFIE–Simulation of fires in enclosures**. In Proceedings of the 5th International Symposium on Fire Safety Science, 1997.

[12] KEVIN B MCGRATTAN, SIMO HOSTIKKA, JE FLOYD, HR BAUM, RG REHM, WILLIAM MELL, AND RANDALL MCDERMOTT. **Fire dynamics simulator (version 5), technical reference guide**. NIST special publication, **1018**:5, 2004. 8

[13] S SIMCOX, NS WILKES, AND IP JONES. **Computer simulation of the flows of hot gases from the fire at King's Cross underground station**. Fire Safety Journal, **18**(1):49–73, 1992. 8

[14] ZHENGHUA YAN AND G HOLMSTEDT. **Investigation of the dance hall fire in Gothenburg, October 1998- a comparison between human observations and CFD simulation.** In InterFlam 2001: 9 th International Fire Science & Engineering Conference, pages 951–963, 2001.

[15] MINGCHUN LUO AND VAUGHAN BECK. **The fire environment in a multi-room building comparison of predicted and experimental results**. Fire safety journal, **23**(4):413–438, 1994.

[16] M LUO AND V BECK. **Flashover fires in a full scale building: prediction and experiment**. In Interflam, **96**, pages 361–370, 1996.

[17] ZHENG WANG, F JIA, ER GALEA, MK PATEL, AND J EWER. **Simulating one of the CIB W14 round robin test cases using the SMARTFIRE fire field model**. Fire Safety Journal, **36**(7):661–677, 2001. 10

[18] F JIA, MK PATEL, ER GALEA, A GRANDISON, AND J EWER. **CFD fire simulation of the Swissair flight 111 in-flight fire-part II: fire spread analysis**. Aeronautical Journal, **110**(1107):303–314, 2006. 18

[19] GH Yeoh, RKK Yuen, SM Lo, and DH Chen. **On numerical comparison of enclosure fire in a multi-compartment building**. Fire Safety Journal, **38**(1):85–94, 2003.

[20] F Liu and Jennifer X Wen. **The effect of turbulence modelling on the CFD simulation of buoyant diffusion flames**. Fire Safety Journal, **37**(2):125–150, 2002.

[21] Cándido Gutiérrez-Montes, Enrique Sanmiguel-Rojas, Antonio Viedma, and Guillermo Rein. **Experimental data and numerical modelling of 1.3 and 2.3 MW fires in a 20m cubic atrium**. Building and Environment, **44**(9):1827–1839, 2009.

[22] Juan Abanto, Marcelo Reggio, Daniel Barrero, and Eddy Petro. **Prediction of fire and smoke propagation in an underwater tunnel**. Tunnelling and underground space technology, **22**(1):90–95, 2007. 8

[23] ER Galea and CS Ierotheou. **Fire-field modelling on parallel computers**. Fire safety journal, **19**(4):251–266, 1992. 8

[24] AJ Grandison, ER Galea, MK Patel, and J Ewer. **The Development of Parallel Implementation for a CFD Based Fire Model Utilising Conventional Office Based PCs**. Journal of Applied Fire Science, **12**(2):137–157, 2003. 8, 28

[25] AJ Grandison, Edwin R Galea, MK Patel, and John Ewer. **Parallel CFD fire modelling on office PCs with dynamic load balancing**. International journal for numerical methods in fluids, **55**(1):29–39, 2007. 8

[26] OpenMP Architecture Review Board. **OpenMP Application Program Interface V3.0**. 2008. 8, 34

[27] N Hurst-Clark, J Ewer, A Grandison, E Galea, et al. **Group solvers: a means of reducing run-times and memory overheads for CFD based fire simulation software**. 2004. 9

[28] Daniel Burton, Angus Grandison, Mayur Patel, Edwin Galea, John Ewer, et al. **Development of a hybrid field/zone fire model**. In Fire Safety Science: Proceedings of the Tenth International Symposium, **10**, pages 1373–1385. International Association for Fire Safety Science, 2011. 9

[29] HENK KAARLE VERSTEEG AND WEERATUNGE MALALASEKERA. An introduction to computational fluid dynamics: the finite volume method. Pearson Education, 2007. 9

[30] OSBORNE REYNOLDS. **On the dynamical theory of incompressible viscous fluids and the determination of the criterion**. Philosophical Transactions of the Royal Society of London. A, pages 123–164, 1895. 9

[31] DAVID C WILCOX. Turbulence modeling for CFD, **2**. DCW industries La Canada, CA, 1998. 9

[32] J. BOUSSINESQ. **Theorie de L'Ecoulement Toubillant**. Mem. Acad. Sci., **23**:46, 1877. 9, 49

[33] KEVIN B MCGRATTAN, HOWARD R BAUM, AND RONALD G REHM. **Large eddy simulations of smoke movement**. Fire Safety Journal, **30**(2):161–178, 1998. 10

[34] STEVEN J EMMERICH AND KEVIN B MCGRATTAN. **Application of a large eddy simulation model to study room airflow**. TRANSACTIONS-AMERICAN SOCIETY OF HEATING REFRIGERATING AND AIR CONDITIONING ENGINEERS, **104**:1128–1140, 1998. 10

[35] VYTENIS BABRAUSKAS, BARBARA C LEVIN, RICHARD G GANN, MAYA PAABO, RICHARD H HARRIS JR, RICHARD D PEACOCK, AND SHYUITSU YUSA. **Toxic potency measurement for fire hazard analysis**. Fire technology, **28**(2):163–167, 1992. 10

[36] VYTENIS BABRAUSKAS. **The generation of CO in bench-scale fire tests and the prediction for real-scale fires**. Fire and Materials, **19**(5):205–213, 1995. 10

[37] WILLIAM M PITTS. **The global equivalence ratio concept and the formation mechanisms of carbon monoxide in enclosure fires**. Progress in Energy and Combustion Science, **21**(3):197–237, 1995. 10

[38] DA PURSER AND JA PURSER. **The potential for including fire chemistry and toxicity in fire safety engineering**. BRE Project Report, (202804), 2003. 10

[39] H XUE, JC HO, AND YM CHENG. **Comparison of different combustion models in enclosure fire simulation**. Fire Safety Journal, **36**(1):37–54, 2001. 10

[40] FALIN CHEN AND JC LEONG. **Smoke flow phenomena and turbulence characteristics of tunnel fires**. Applied Mathematical Modelling, **35**(9):4554–4566, 2011.

[41] L WANG AND Q CHEN. **Theoretical and numerical studies of coupling multizone and CFD models for building air distribution simulations**. Indoor Air, **17**(5):348–361, 2007.

[42] G. H YEOH, V. (VUTE) CHANDRASEKARAN, AND E LEONARDI. Numerical prediction of fire and smoke spread / G.H. Yeoh, V. Chandrasekaran and E. Leonardi. [S.l.] : CSIRO. Division of Building, Construction and Engineering, 1995. Repr: Australian refrigeration, air conditioning and heating, vol. 49, no. 1, Jan. 1995, p. 13-14, 16-18. 10

[43] JE FLOYD AND KB MCGRATTAN. **Extending the mixture fraction concept to address under-ventilated fires**. Fire Safety Journal, **44**(3):291–300, 2009. 10

[44] L KERRISON, ER GALEA, N HOFFMANN, AND MK PATEL. **A comparison of a FLOW3D based fire field model with experimental room fire data**. Fire safety journal, **23**(4):387–411, 1994. 10

[45] L KERRISON, N MAWHINNEY, E.R GALEA, N HOFFMANN, AND M.K. PATEL. **A Comparison of Two Fire Field Models with Experimental Room Fire Data**. Proceedings of the 4th International Symposium on Fire Safety Science, pages 161–172, 1994.

[46] JOUNI BJÖRKMAN AND OLAVI KESKI-RAHKONEN. **Simulation of the Steckler room fire experiment by using SOFIE CFD-model**. VTT PUBLICATIONS, 1996. 10, 180

[47] GV HADJISOPHOCLEOUS AND CJ MCCARTNEY. **Guidelines for the use of CFD simulations for fire and smoke modeling**. ASHRAE transactions, pages 583–594, 2005. 11

[48] Gordon Moore. **Progress in Digital Integrated Electronics IEEE**. IEDM Tech Digest, **24**:11–13, 1975. 11, 31

[49] Hadi Esmaeilzadeh, Emily Blem, Renee St Amant, Karthikeyan Sankaralingam, and Doug Burger. **Dark silicon and the end of multicore scaling**. In Computer Architecture (ISCA), 2011 38th Annual International Symposium on, pages 365–376. IEEE, 2011. 11

[50] ER Galea et al. **The use of computer simulation for aircraft evacuation certification: A report from the VERRES project**. In The 4th Triennial International Fire and Cabin Safety Research Conference, Lisbon, Portugal, 2004. 11

[51] Vytenis Babrauskas, Joseph M Fleming, and B Don Russell. **RSET/ASET, a flawed concept for fire safety assessment**. Fire and Materials, **34**(7):341–355, 2010. 11, 18

[52] E Galea, B Knight, M Patel, J Ewer, M Petridis, and S Taylor. **SMARTFIRE V2. 01 build 369D, User Guide and Technical Manual**. SMARTFIRE CD and bound manual, 1999. 12, 18, 20

[53] Kevin B McGrattan and Glenn P Forney. Fire Dynamics Simulator: User's Manual. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2000. 12

[54] A Yu Snegirev, JA Marsden, J Francis, and GM Makhviladze. **Numerical studies and experimental observations of whirling flames**. International journal of heat and mass transfer, **47**(12):2523–2539, 2004. 12

[55] G.V. Hadjisophocleous and A. Yakan. **Computer modeling of compartment fires**. Internal Report, (613), 1991. 12

[56] M Aksit, P Mackie, and PA Rubini. **Coupled radiative heat transfer and flame spread simulation in a compartment**. In Proceedings of the 3rd International seminar on Fire and Explosion Hazards, Windermere, UK, 2000. 12

[57] AJ Gardiner. The mathematical modelling of the interaction between sprinkler sprays and the thermally buoyant layers of gases from fires. PhD thesis, Polytechnic of the South Bank, 1988. 12

[58] N.S. Wilkes, J.H. Alderton, and L.M. MacIntosh. A comparison of predictions with experimental data for a fire in a hospital ward: 1 - preliminary precautions. AERE memorandum / M: AERE memorandum. Atomic Energy Research Establishment, Engineering Sciences Division, 1988. 12

[59] DF Fletcher, JH Kent, VB Apte, and AR Green. **Numerical simulations of smoke movement from a pool fire in a ventilated tunnel**. Fire Safety Journal, **23**(3):305–325, 1994. 12

[60] G Cox and S Kumar. **Field modelling of fire in forced ventilated enclosures**. Combustion Science and Technology, **52**(1-3):7–23, 1987. 12

[61] CHAM Limited. **Documentation for PHOENICS TR003 version 2006**. `http://www.cham.co.uk/documentation/tr003.pdf`. Accessed Jan 26, 2015. 12

[62] Kevin B McGrattan and William H Twiley. Smoke plume trajectory from in situ burning of crude oil in Alaska. United states department of commerce, National technical information service, 1993. 12

[63] ER Galea and JM Perez Galparsoro. **A computer-based simulation model for the prediction of evacuation from mass-transport vehicles**. Fire Safety Journal, **22**(4):341–366, 1994. 13

[64] Lawrence P J Gwynne S Filippidis L Blackshields D Galea, E R and D Cooney. **buildingEXODUS v5.1 User Guide and Technical Manual**. 2013. 13, 18, 23, 24, 25

[65] Thunderhead Engineering. **Agent Based Emergency Egress Simulation Technical Reference**. `http://www.thunderheadeng.com/pathfinder/documentation.html`. Accessed Feb 19, 2014. 13

[66] BRE Fire Research and Consultancy. **Human behaviour in fire and emergency evacuation design**. `http://www.bre.co.uk/fire/page.jsp?id=269`. Accessed Feb 19, 2014. 13

[67] D Purser and KE Boyce. **Implications of Modelling and Experimental Studies of Evacuation Behaviour on Stairs for Multi-storey Building Design**. Human Behaviour in Fire, pages 147–160, 2009. 13

[68] Simo Hostikka, Timo Korhonen, Tuomas Paloposki, Tuomo Rinne, Katri Matikainen, and Simo Heliövaara. **Development and validation of FDS+ Evac for evacuation simulations**. VTT RESEARCH, 2007. 13, 18

[69] Xiaoshan Pan. Computational modeling of human and social behaviors for emergency egress analysis. PhD thesis, Stanford University, 2006. 13

[70] Jean Louis Berrou, Jonathan Beecham, Philippe Quaglia, Marios A Kagarlis, and Alex Gerodimos. **Calibration and validation of the Legion simulation model using empirical data**. In Pedestrian and Evacuation Dynamics 2005, pages 167–181. Springer, 2007. 13

[71] Hubert Klüpfel and Tim Meyer-König. **Simulation of the Evacuation of a football stadium using the CA Model PedGo**. In Traffic and Granular Flow03, pages 423–428. Springer, 2005. 13

[72] Dracos Vassalos, HS Kim, Guro Christiansen, and Jayanta Majumder. **A mesoscopic model for passenger evacuation in a virtual ship-sea environment and performance-based evaluation**. 2002. 13

[73] Peter A Thompson and Eric W Marchant. **A computer model for the evacuation of large building populations**. Fire Safety Journal, **24**(2):131–148, 1995. 13

[74] Peter Thompson, Jianhua Wu, and E Marchant. **Modelling evacuation in multi-storey buildings with Simulex**. Fire Engineers Journal, **56**:6–11, 1996. 13

[75] University of Florida. **Evacnet4 Users Guide**. `http://tomkisko.com/ise/files/evacnet/EVAC4UG.HTM`. Accessed Feb 19, 2014. 13

[76] NP Waterson and E Pellissier. **The STEPS Pedestrian Microsimulation Tool-A Technical Summary**. Mott MacDonald Limited, Croydon, UK, 2012. 13, 17

[77] Oasys Software Limited. **MassMotion v6.1.0 Manual - Design Simulate Optimize**. Oasis Software Ltd., 2014. 13

[78] Fred I Stahl. **BFIRES-II: a behavior based computer simulation of emergency egress during fires**. Fire technology, **18**(1):49–65, 1982. 13

[79] ED Kuligowski. **Review of 28 egress models**. In Kuligowski (Eds.), Proceedings of the Workshop on Building Occupant Movement during Fire Emergencies, pages 68–90, 2004. 14

[80] Kevin McGrattan, Simo Hostikka, Randall McDermott, Jason Floyd, Craig Weinschenk, and Kristopher Overholt. **Fire dynamics simulator, technical reference guide, volume 2: Verification**. NIST Special Publication, **1018**, 2013. 15

[81] Stanford University. **What is Egress?** `http://eil.stanford.edu/pengao/ResentFocus/`. Accessed Feb 19, 2014. 15

[82] traffgo ht. **PedGo**. `http://traffgo-ht.com/en/pedestrians/products/pedgo/`. Accessed Feb 19, 2014. 16

[83] Suiping Zhou, Dan Chen, Wentong Cai, Linbo Luo, Malcolm Yoke Hean Low, Feng Tian, Victor Su-Han Tay, Darren Wee Sze Ong, and Benjamin D Hamilton. **Crowd modeling and simulation technologies**. ACM Transactions on Modeling and Computer Simulation (TOMACS), **20**(4):20, 2010. 16

[84] RW Bukowski, TE Waterman, and WJ Christian. **Detector sensitivity and siting requirements for dwellings: Report of the NBS Indiana Dunes Tests**. NFPA No. SPP-43, National Fire Protection Association, Quincy, MA, 1975. 18

[85] Edwin R Galea, Zhaozhi Wang, Anand Veeraswamy, Fuchen Jia, Peter J Lawrence, John Ewer, et al. **Coupled fire/evacuation analysis of the Station Nightclub fire**. In Proc of 9th IAFSS Symp, pages 465–476, 2008. 18

[86] S Gwynne, ER Galea, PJ Lawrence, and L Filippidis. **Modelling occupant interaction with fire conditions using the buildingEXODUS evacuation model**. Fire Safety Journal, **36**(4):327–357, 2001. 18, 23, 25

[87] WALTER W JONES, RICHARD D PEACOCK, GLENN P FORNEY, AND PA RENEKE. **CFAST–Consolidated model of fire growth and smoke transport (Version 6)**. Technical reference guide. NIST SP, **1030**:153, 2005. 18

[88] JN FRASER-MITCHELL. **Modelling human behaviour within the fire risk assessment tool CRISP**. Fire and Materials, **23**(6):349–355, 1999. 18, 19

[89] GEORGI S DJAMBAZOV, CHOI-HONG LAI, KOULIS A PERICLEOUS, ET AL. **Efficient computation of aerodynamic noise**. Contemporary Mathematics, **218**:500–506, 1998. 18

[90] BJORN KARLSSON AND KOKKALA M. **New Developments in Performance Based Test Methods for Fire safety Assessment of Products**. In Proceedings of the CIB W60 Workshop on New Developments in Performance Test Methods, 1995. 19

[91] BJORN KARLSSON, GREG NORTH, AND DANIEL GOJKOVIC. **Using results from performance-based test methods for material flammability in fire safety engineering design**. Journal of Fire Protection Engineering, **12**(2):93–108, 2002.

[92] SE MAGNUSSON, H FRANTZICH, B KARLSSON, AND S SÄRDQVIST. **Determination of safety factors in design based on performance**. In Proceedings of the Fourth International Symposium on Fire Safety Science, pages 937–948, 1994. 19

[93] KORHONEN TIMO AND SIMO HOSTIKKA. **Fire Dynamics Simulator with Evacuation: FDS+Evac, Users Guide**. VTT Working Papers, **119**, 2009. 19

[94] TIMO KORHONEN, SIMO HOSTIKKA, SIMO HELIÖVAARA, HARRI EHTAMO, AND KATRI MATIKAINEN. **Integration of an agent based evacuation simulation and the state-of-the-art fire simulation**. In Proceedings of the 7th Asia-Oceania symposium on fire science & technology, pages 20–22, 2007. 19

[95] JEREMY FRASER-MITCHELL, SUNG-HAN KOO, AND STEPHEN WELCH. **Sensor-linked simulation for emergency response**. 2009. 19

[96] JOHN ANDREW CLARK EWER ET AL. An Investigation Into the Feasibility, Problems and Benefits of Re-engineering a Legacy Procedural CFD Code Into an Event Driven, Object Oriented System that Allows Dynamic User........ University of Greenwich, 2000. 20

[97] S Kumar, AK Gupta, and G Cox. **Effects of thermal radiation on the fluid dynamics of compartment fires**. In Fire Safety ScienceProceedings of the Third International Symposium, pages 345–354, 1991. 20

[98] GD Raithby and EH Chui. **A finite-volume method for predicting a radiant heat transfer in enclosures with participating media**. Journal of Heat Transfer, **112**(2):415–423, 1990. 20

[99] MJ Lewis, MB Moss, and PA Rubini. **CFD modelling of combustion and heat transfer in compartment fires**. In Proc. 5th Int. Symp. on Fire Safety Science, pages 463–474. International Association for Fire Safety Science Australia, 1997. 20

[100] FSEG University of Greenwich. **SMARTFIRE INTRODUCTION**. `http://fseg.gre.ac.uk/smartfire/`. Accessed Feb 17, 2014. 22

[101] ER Galea, L Filippidis, Z Wang, and J Ewer. **Fire and evacuation analysis in BWB aircraft configurations: computer simulations and large-scale evacuation experiment**. Aeronautical Journal, **114**(1154):271–277, 2010. 23

[102] Galea E R Lawrence P Gwynne S Deere, S. **The Impact of the agent response time distribution on ship evacuation performance**. Journal of Maritime Engineering, **148**(Part A1()):35–44, 2006. 23

[103] Tadahisa Jin. **Visibility through fire smoke**. Journal of Fire and Flammability, **9**(2):135–155, 1978. 25

[104] Tadahisa Jin and Tokiyoshi Yamada. **Irritating effects of fire smoke on visibility**. Fire Science and Technology, **5**(1):79–90, 1985. 25

[105] DA Purser. **Assessment of hazards to occupants from smoke, toxic gases, and heat**. SFPE handbook of fire protection engineering, **4**:2–96, 2008. 25

[106] Stanley Gill. **Parallel programming**. The Computer Journal, **1**(1):2–10, 1958. 26, 27

[107] Michael J Flynn and Kevin W Rudd. **Parallel architectures**. ACM Computing Surveys (CSUR), **28**(1):67–70, 1996. 26

[108] Daniel L Slotnick, W Carl Borck, and Robert C McReynolds. **The SOLOMON computer**. In Proceedings of the December 4-6, 1962, fall joint computer conference, pages 97–107. ACM, 1962. 27

[109] JR Ball, RC Bollinger, TA Jeeves, RC McReynolds, and DH Shaffer. **On the use of the SOLOMON parallel-processing computer**. In Proceedings of the December 4-6, 1962, fall joint computer conference, pages 137–146. ACM, 1962. 27

[110] Arthur J Bernstein. **Analysis of programs for parallel processing**. Electronic Computers, IEEE Transactions on, (5):757–763, 1966. 27

[111] Alfredo Buttari, Julien Langou, Jakub Kurzak, and Jack Dongarra. **A class of parallel tiled linear algebra algorithms for multicore architectures**. Parallel Computing, **35**(1):38–53, 2009. 27

[112] TOP500.org (c). **TOP500 Supercomputer Site**. `http://www.top500.org`. Accessed Feb 19, 2015. 27

[113] Alberto Valli, Alfio Quarteroni, et al. Domain decomposition methods for partial differential equations. Number CMCS-BOOK-2009-019. Oxford University Press, 1999. 28

[114] K.A Duke and W.A Wall. **A professional graphics controller**. IBM Systems Journal, **24**(1):14–25, 1985. 29

[115] **VisiCorp Visi On**. `http://toastytech.com/guis/vision.html`, August 2013. 29

[116] Tim Dettmers. **Which GPU(s) to Get for Deep Learning: My Experience and Advice for Using GPUs in Deep Learning**, 2014. 30

[117] CUDA Nvidia. **Programming guide**, 2008. 30, 33, 35, 39, 42

[118] Khronos OpenCL Working Group et al. **The opencl specification**. A. Munshi, Ed, 2008. 30, 35

[119] P. Glaskowsky. **The first complete gpu computing architecture**. Technical report, NVIDIA, 2009. 31

[120] Computer Organisation and Design. Elsevier, 2008. 31

[121] Nvidia. **Nvidia's next generation cuda architecture: Fermi**. Technical report, Nvidia, 2009. 31, 37

[122] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krger, Aaron E. Lefohn, and Tim Purcell. **A survey of general-purpose computation on graphics hardware**, 2007. 31

[123] John Owens. **Data-parallel algorithms and data structures**. In ACM SIGGRAPH 2007 courses, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[124] Yi Yang, Ping Xiang, Jingfei Kong, and Huiyang Zhou. **A GPGPU compiler for memory optimization and parallelism management**. SIGPLAN Not., **45**(6):86–97, June 2010. 31

[125] J. Palacios and J. Triska. **A Comparison of Modern GPU and CPU Architectures: And the Common Convergence of Both**, 3 2011. 31

[126] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. **Scalable parallel programming with CUDA**. Queue, **6**(2):40–53, 2008. 31, 41

[127] Mark Harris. **Gpgpu: General-purpose computation on gpus**. SIGGRAPH 2005 GPGPU COURSE, 2005. 31

[128] Linux Magazine. **The Return of the Vector Processor**. `http://www.linux-mag.com/id/7575/`, Oct 2009. 32

[129] Nvidia Corp. **COMPUTATIONAL FLUID DYNAMICS**. `http://www.nvidia.com/object/computational_fluid_dynamics.html`. Accessed April 17, 2014. 32

[130] **Nvidia Corp.** `http://www.nvidia.com/`, August 2013. Accessed March 08, 2013. 33

[131] KHRONOS OPENCL WORKING GROUP ET AL. **OpenCL**. `http://www.khronos.org/opencl/`, October 2009. 33

[132] INTEL. **Intel Xeon Phi Coprocessor Developer's quick start guide**, 2013. 33

[133] PENG DU, RICK WEBER, PIOTR LUSZCZEK, STANIMIRE TOMOV, GREGORY PETERSON, AND JACK DONGARRA. **From {CUDA} to OpenCL: Towards a performance-portable solution for multi-platform {GPU} programming**. Parallel Computing, **38**(8):391 – 407, 2012. 34

[134] DONALD E THOMAS AND PHILIP R MOORBY. The Verilog® Hardware Description Language, **2**. Springer, 2002. 34

[135] T. HOSHINO, N. MARUYAMA, S. MATSUOKA, AND R. TAKAKI. **CUDA vs OpenACC: Performance Case Studies with Kernel Benchmarks and a Memory-Bound CFD Application**. In Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on, pages 136–143, May 2013. 34

[136] N BELL AND J HOBEROCK. **The thrust library**, 2010. 35

[137] JOHN R HUMPHREY, DANIEL K PRICE, KYLE E SPAGNOLI, AARON L PAOLINI, AND ERIC J KELMELIS. **CULA: hybrid GPU accelerated linear algebra routines**. In SPIE Defense, Security, and Sensing, pages 770502–770502. International Society for Optics and Photonics, 2010. 35

[138] ROB FARBER. CUDA application design and development. Access Online via Elsevier, 2011. 35

[139] D.H. LAWRIE. **Access and Alignment of Data in an Array Processor**. Computers, IEEE Transactions on, **C-24**(12):1145–1155, 1975. 37

[140] NVIDIA. **NVIDIAs Next Generation CUDA Compute Architecture: Kepler GK110**. `http://www.nvidia.co.uk/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf`, 2012. 37, 38

[141] NVIDIA. **NVIDIA Tesla C2075 Companion Processor**. `http://www.nvidia.com/content/PDF/data-sheet/NV_DS_Tesla_C2075_Sept11_US_HR.pdf`, 2011. 39

[142] CHARLES AR HOARE. **Quicksort**. The Computer Journal, **5**(1):10–16, 1962. 41

[143] DAVID B. KIRK AND WEN-MEI W. HWU. Programming Massively Parallel Processors: A Hands-on Approach. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2010. 41

[144] C.M. WITTENBRINK, E. KILGARIFF, AND A. PRABHU. **Fermi GF100 GPU Architecture**. Micro, IEEE, **31**(2):50–59, 2011. 41

[145] Y. TORRES, A. GONZALEZ-ESCRIBANO, AND D.R. LLANOS. **Understanding the impact of CUDA tuning techniques for Fermi**. In High Performance Computing and Simulation (HPCS), 2011 International Conference on, pages 631–639, 2011. 45

[146] SUSIM KUMAR. **Mathematical modeling of natural convection in fire A state of the art review of the field modelling of variable density turbulent flow**. Fire and Materials, **7**(1):1–24, 1983. 49

[147] JACK GM EGGELS. **Direct and large-eddy simulation of turbulent fluid flow using the lattice-Boltzmann scheme**. International journal of heat and fluid flow, **17**(3):307–323, 1996. 51, 210

[148] F JIA, ER GALEA, MK PATEL, ET AL. **The numerical simulation of fire spread within a compartment using an integrated gas and solid phase combustion model**. Journal of Applied Fire Science, **8**(4):327–352, 1999. 51

[149] JOHN R HOWELL. **Thermal radiation in participating media: the past, the present, and some possible futures**. Journal of Heat Transfer, **110**(4b):1220–1229, 1988. 52

[150] HOYT CLARKE HOTTEL AND ADEL F SAROFIM. Radiative transfer. McGraw-Hill New York, 1967. 52

[151] JOHN R HOWELL. **Application of Monte Carlo to heat transfer problems**. Advances in heat transfer, **5**(1):1–54, 1968. 52

[152] H. C. HAMAKER. **Radiation and Heat Conduction in Light-Scattering Material: Part I Reflection and Transmission**. Technical report, Philips Research, 1947. 52

[153] H. C. HAMAKER. **Radiation and Heat Conduction in Light-Scattering Material: Part II**. Technical report, Philips Research, 1947.

[154] ARTHUR SCHUSTER. **Radiation through a foggy atmosphere**. The astrophysical journal, **21**:1, 1905. 52

[155] FC LOCKWOOD AND NG SHAH. **A new radiation solution method for incorporation in general combustion prediction procedures**. In Symposium (international) on combustion, **18**, pages 1405–1414. Elsevier, 1981. 52, 63

[156] UNITED STATES DEPARTMENT OF TRASPORTATION. **RITA - Research and Innovation Techology Administartion**. `http://ntl.bts.gov/DOCS/ch5.html`. Accessed Jan 06, 2015. 56

[157] CM RHIE AND WL CHOW. **Numerical study of the turbulent flow past an airfoil with trailing edge separation**. AIAA journal, **21**(11):1525–1532, 1983. 58, 65

[158] FUCHEN JIA. The Simulation of Fire Growth and Spread within Enclosures using an integrated CFD Fire Field Model,. PhD thesis, University of Greenwich, 1999. 63, 72

[159] ADEL F SAROFIM. **Radiative heat transfer in combustion: Friend or foe**. In Symposium (International) on Combustion, **21**, pages 1–23. Elsevier, 1988. 63

[160] BENGT G CARLSON, KAYE D LATHROP, ET AL. Transport theory: the method of discrete ordinates. Los Alamos Scientific Laboratory of the University of California, 1965. 64

[161] SUHAS V PATANKAR AND D BRIAN SPALDING. **A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows**. International Journal of Heat and Mass Transfer, **15**(10):1787–1806, 1972. 67

[162] JP VAN DOORMAAL AND GD RAITHBY. **Enhancements of the SIMPLE method for predicting incompressible fluid flows**. Numerical heat transfer, **7**(2):147–163, 1984. 67

[163] GEORGE KARYPIS AND VIPIN KUMAR. **Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0**. 1995. 76

[164] Chris Walshaw and Mark Cross. **JOSTLE: parallel multilevel graph-partitioning software–an overview**. Mesh partitioning techniques and domain decomposition techniques, pages 27–58, 2007. 76

[165] Bruce Hendrickson and Robert Leland. **The Chaco users guide: Version 2.0**. Technical report, Technical Report SAND95-2344, Sandia National Laboratories, 1995. 76

[166] François Pellegrini and Jean Roman. **Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs**. In High-Performance Computing and Networking, pages 493–498. Springer, 1996. 76

[167] Angus Joseph Grandison. Improving the regulatory acceptance and numerical performance of CFD based fire-modelling software. PhD thesis, University of Greenwich, 2003. 76, 79, 80, 88, 106

[168] Bibi Yasmina Yashanaz Mohedeen. Domain Partitioning and software modifications towards the parallelisation of the buildingEXODUS evacuation software. PhD thesis, University of Greenwich, 2011. 76

[169] **NVIDIA Developer Zone**. `https://developer.nvidia.com/`. Accessed August 2013. 84

[170] Kevin McManus. A strategy for mapping unstructured mesh computational mechanics programs onto distributed memory parallel architectures. PhD thesis, University of Greenwich London, UK, 1996. 88

[171] KD Steckler, James G Quintiere, and WJ Rinkinen. **Flow induced by fire in a compartment**. In Symposium (international) on combustion, **19**, pages 913–920. Elsevier, 1982. 92, 93, 180

[172] AJ Grandison, ER Galea, MK Patel, and David Peace. Fire Modelling Standards/benchmark: Report on SMARTFIRE Phase 2 Simulations. University of Greenwich, Centre for Numerical Modelling and Process Analysis, 2001. 92, 93

[173] NATHAN WHITEHEAD AND ALEX FIT-FLOREA. **Precision & performance: Floating point and IEEE 754 compliance for NVIDIA GPUs**. nVidia technical white paper, **21**:1–1874919424, 2011. 94

[174] EDWIN R GALEA, ZHAOZHI WANG, MADELEINE TOGHER, FUCHEN JIA, AND PETER LAWRENCE. **Predicting the likely impact of aircraft post crash fire on aircraft evacuation using fire and evacuation simulation**. In Proc. international fire and cabin safety research conference, 2007. 116

[175] S TAYLOR, M PETRIDIS, B KNIGHT, J EWER, ER GALEA, M PATEL, ET AL. **SMARTFIRE: an integrated computational fluid dynamics code and expert system for fire field modelling**. 1997. 187

[176] M PETRIDIS, B KNIGHT, AND D EDWARDS. **A design for reliable CFD software**. In Reliability and Robustness of Engineering Software II, pages 3–17. Springer, 1991. 187

[177] NVIDIA CORP. **3D Graphics Demystified**. `http://www.intel.com/`, August 2013. 207

[178] THOMAS SCOTT CROW. Evolution of the Graphical Processing Unit. Master thesis, University of Nevada, Reno, December 2004.

[179] WIKIPEDIA. **Graphics pipeline**. `https://en.wikipedia.org/wiki/Graphics_pipeline`, August 2013. 207

[180] THOMAS VOGEL. **BASIC COMPUTER GRAPHICS**. `http://prosjekt.ffi.no/unik-4660/lectures04/chapters/Basic.html`, August 2013. 208

[181] JOHN CHARLES BUTCHER. The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods. Wiley-Interscience, 1987. 209

[182] RICHARD COURANT, KURT FRIEDRICHS, AND HANS LEWY. **Über die partiellen Differenzengleichungen der mathematischen Physik**. Mathematische Annalen, **100**(1):32–74, 1928. 209

[183] Richard Courant, Kurt Friedrichs, and Hans Lewy. **On the partial difference equations of mathematical physics**. IBM journal of Research and Development, **11**(2):215–234, 1967. 209

[184] George Karypis and V Kumar. **Metis manual**. University of Minnesota/Department of Science/Army HPC Research Center, 2011. 211

Appendix A

Appendix

A.1 Pipelines

A graphics pipeline or rendering pipeline refers to a sequence of steps to transform a 3D scene into a 2D raster. The concept exists in three versions, software implementation[1], hardware implementation[2] or a combination of both. Simply speaking it converts coordinates which are easier for the programmer to implement into something that is more convenient for the monitor to display. The process chain takes any 3D data i.e a geometry model or shapes consisting of triangles or coordinates (x,y,z) and transforms it to pixels that can be displayed on a 2D monitor, an example is shown in Figure A.1. The simplest pipeline concept consists of two stages, processing stage (3D $\rightarrow$ 2D) and rendering stage to create the surface of the 2D object[177–179].

[1]OpenGL or Microsoft Direct3D

[2]GPU

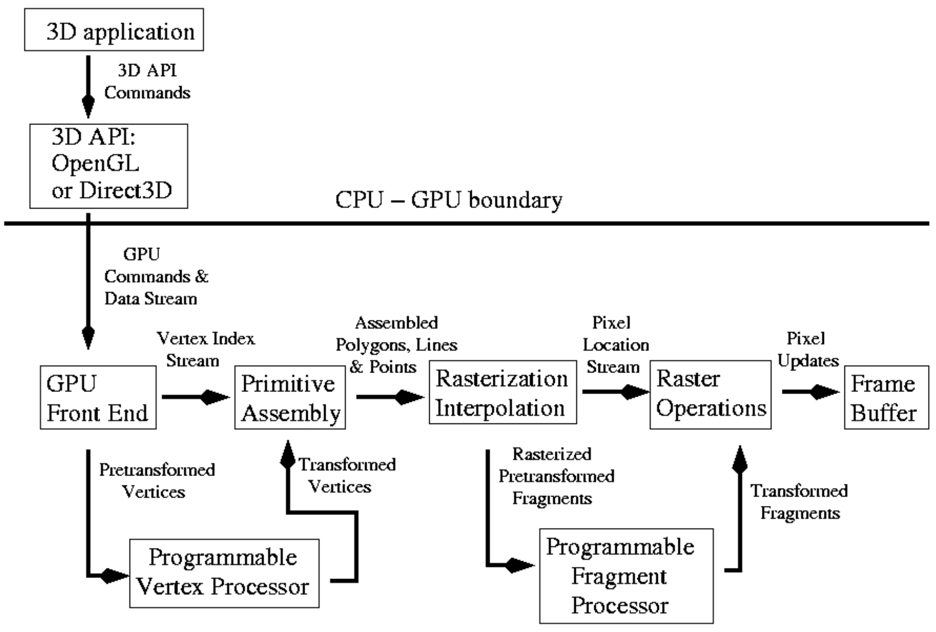


Figure A.1: Graphics pipeline in NVIDIA GeForce 8800 [180]

A.2 Explicit Discretisation

In the previous section an implicit temporal discretisation scheme has been introduced. An explicit discretisation will now be described here. The explicitly discretised general conservation equation(c.f. implicit equation (3.3.7) is given in (A.2.1)).

$$\frac{\left[(\rho\phi) - (\rho\phi)^{old}\right]_P}{\delta t} V_P + \sum_{\substack{\text{All faces}\\ \text{of FV}}} \left(\rho_f (\underline{u}.\underline{\hat{n}})_f \phi_f^{old} - \Gamma_\phi \left(\frac{\partial \phi}{\partial x} n_x + \frac{\partial \phi}{\partial y} n_y \frac{\partial \phi}{\partial z} n_z \right)^{old} \right) A_f = S_{\phi,P} V_P \tag{A.2.1}$$

which leads to equation (A.2.2)(c.f. implicit equation 3.3.18):

$$a_P \phi_P = \sum_{\substack{\text{All adjacent}\\ \text{nodes}}} (a_A \phi_A)^{old} + b \tag{A.2.2}$$

The difference between the explicit ((A.2.1) and (A.2.2))and the implicit ((3.3.7) and (A.2.2)) formulation is that the value of ϕ_P can now be calculated explicitly from the previously calculated old values of neighbouring ϕ and therefore there is no coupling of the neighbouring values of ϕ^{new} as found in the implicit formulation. The result is a set of independent ordinary differential equations(ODE) for each point in the physical domain that can be solved by using methods such as corrector-predictor or Runge-Kutta methods [181].

However it has to be ensured that sufficiently small time steps are utilised or the results can result in physically in-correct results [2]. The time step size δt is limited by the Courant-Friedrichs-Lewy (CFL) limit [182; 183] ($\delta t < \frac{\delta x}{u}|_{min}$) where δx is the grid cell size and u is the velocity within the corresponding grid cell.The implicit formulation is not limited in this way so that longer time steps can be used for the temporal discretisation. The general rule is that implicit methods are faster than explicit ones due to the longer time step size that can be used. However it may still be useful to apply an explicit method when small time step sizes are inherently required e.g Large Eddy Simulation (LES).

A.3 Large Eddy Simulation Model

In LES models the instantaneous equations are not time averaged but filtered instead to remove all eddies smaller than a certain size, usually the size equals the resolution of the cells in the domain. Mathematically this is expressed in A.3.1

$$\overline{f}(x) = \int_D f(x^{'})G(x, x^{'}; \overline{\Delta})dx^{'} \tag{A.3.1}$$

where the overbar stands for the already filtered function, D is the entire domain, G stands for the filtering function and $\overline{\Delta}$ is the filtering width. The filter function defines the size and structure scales. Similarly to the Reynolds averaging process in 3.1.5.1, the variable can be decomposed in two parts.

$$\phi = \overline{\phi} + \phi^{'} \tag{A.3.2}$$

but the definition is different here. $\phi^{'}$ stands for the fluctuation of ϕ at length $< \overline{\Delta}$. Applying the filter to the momentum equation and dropping the overbars leads to A.3.3.

$$\frac{\partial(\rho u_i)}{\partial t} + \nabla(\rho u_i \underline{u}) = \mu div(grad(u_i)) - \frac{\partial P}{\partial x_i} + S_i - \frac{\partial}{\partial x_j}(\rho \tau_{ij}) \tag{A.3.3}$$

The filtered fields do not need to resolve at scales $< \overline{\Delta}$ and can therefore calculated properly. The last term in equation A.3.3,$\tau_{ij} = (\overline{u_i u_j} - \bar{u}_i \bar{u}_j)$ is called the sub-grid- scale stress tensor and represents the effects that the small scales have. This term needs to be modelled.
The stress term of an eddy-viscosity model can be modelled as

$$\tau_{ij} - \frac{1}{3}\tau_{kk}\delta_{ij} = -V_t\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) = -2v_t\bar{S}_{ij} \tag{A.3.4}$$

S_{ij} is the deformation tensor of the filtered velocity(gradient). The unknown variables have been replaced with the known filtered values, similar to the time-averaged approach. Which leaves the only term required to model LES turbulent viscosities. The most common and widely used model was proposed by Smagorinsky [147]. The assumption about a local mixing-length is made and the eddy viscosity is assumed to be proportional to $\overline{\Delta}$ and the characteristic turbulent velocity determined by $\overline{\Delta}\left|\bar{S_{ij}}\right|$. The turbulent eddy viscosity is given by Equation A.3.5

$$v_t = \left(C_S\bar{\Delta}\right)^2\left|\bar{S_{ij}}\right| \tag{A.3.5}$$

C_S can either be constant or a variable depending on the method of modelling chosen. This term will get damped where turbulence can not totally develop e.g. during transitions and close to walls.

Small time steps are necessary to achieve time accuracy. Which means the advantage of using an implicit scheme, larger time steps, are lost. As implicit schemes have no real advantage any more explicit schemes become interesting. The explicit scheme uses a set of ODE for the thermodynamic quantities.
The main advantage of the LES in FSE that large eddies found in fire plumes are lost or get averaged out by using a time average approach. LES visually look more realistic in contrast to time averaged simulations. However the main advantage of using time averaged approaches still is the execution time.

A.4 Metis Input Format

Metis partitions the vertices of a graph in n roughly equal parts, such that the number of edges connecting vertices in different parts is minimized. Which essentially means the surface between the individual parts will be minimised to avoid additional communication overhead. In order to do this a certain input data format for the graph is required, which SMARTFIRE already provides. An example for a simplified graph is shown in Figure A.2 which can be expressed i the form of table A.1. Row ”xadj” has N elements, $N(i+1) - N(i)$ gives the number of neighbours per cell. The values in xadj represent cell numbers in row ”adjncy” where the group of neighbouring cells is listed. Therefore all information to describe the graphs are given. When implemented the adjncy row can been seen as places in memory and xadj essentially pointers into memory.

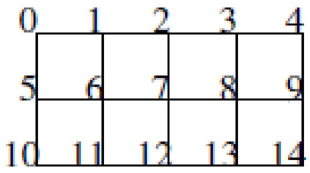


Figure A.2: A sample graph [184]

Table A.1: Metis input for a sample graph see Figure A.2 [184]

xadj	0	2	5	8	11	13	16	20	24	28	31	33	36	39	42	44
adjncy	1	5	0	2	6	1	3	7	2	4	8	3	9	0	6	10
	1	5	7	11	2	6	8	12	3	7	9	13	4	8	14	5
	11	6	10	12	7	11	13	8	12	14	9	13				

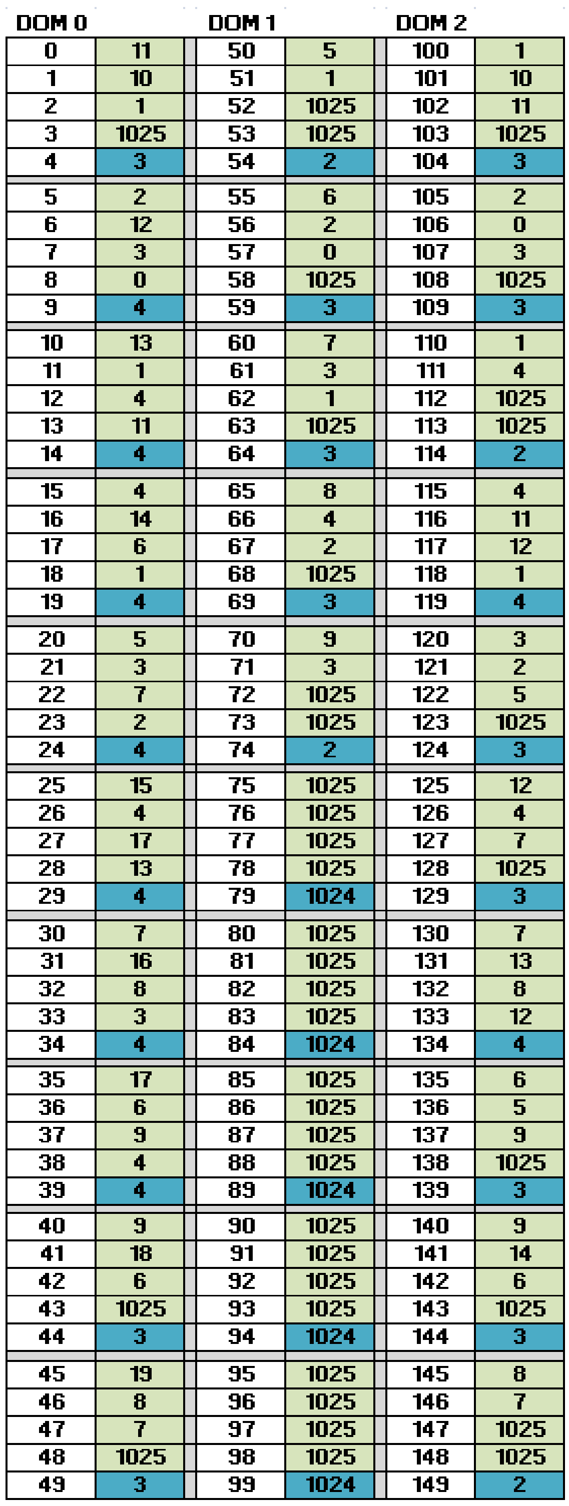
A.5 Shared Memory Mapping

1	6	7	11	12	13	16	17	21	22	0	2	5	8	10	14	15	18	20	23
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

0	5	10	15	20	1	6	11	16	21	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

2	3	4	8	9	10	18	19	23	24	1	7	13	17	22	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

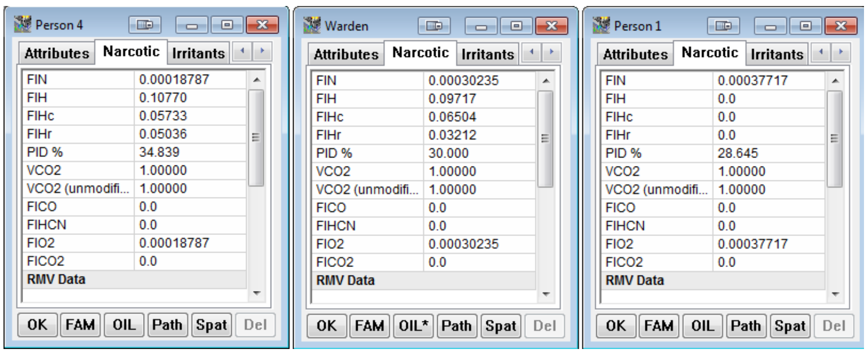
Figure A.4: Shared memory mapping on a block by block basis.



DOM 0		DOM 1		DOM 2	
0	11	50	5	100	1
1	10	51	1	101	10
2	1	52	1025	102	11
3	1025	53	1025	103	1025
4	3	54	2	104	3
5	2	55	6	105	2
6	12	56	2	106	0
7	3	57	0	107	3
8	0	58	1025	108	1025
9	4	59	3	109	3
10	13	60	7	110	1
11	1	61	3	111	4
12	4	62	1	112	1025
13	11	63	1025	113	1025
14	4	64	3	114	2
15	4	65	8	115	4
16	14	66	4	116	11
17	6	67	2	117	12
18	1	68	1025	118	1
19	4	69	3	119	4
20	5	70	9	120	3
21	3	71	3	121	2
22	7	72	1025	122	5
23	2	73	1025	123	1025
24	4	74	2	124	3
25	15	75	1025	125	12
26	4	76	1025	126	4
27	17	77	1025	127	7
28	13	78	1025	128	1025
29	4	79	1024	129	3
30	7	80	1025	130	7
31	16	81	1025	131	13
32	8	82	1025	132	8
33	3	83	1025	133	12
34	4	84	1024	134	4
35	17	85	1025	135	6
36	6	86	1025	136	5
37	9	87	1025	137	9
38	4	88	1025	138	1025
39	4	89	1024	139	3
40	9	90	1025	140	9
41	18	91	1025	141	14
42	6	92	1025	142	6
43	1025	93	1025	143	1025
44	3	94	1024	144	3
45	19	95	1025	145	8
46	8	96	1025	146	7
47	7	97	1025	147	1025
48	1025	98	1025	148	1025
49	3	99	1024	149	2

Figure A.3: ϕ_P^i mapping on a block by block basis for the two dimensional domain as in Figure 4.9.

A.6 Simple Geometry - Narcotic properties



Person 4

FIN	0.00018787
FIH	0.10770
FIHc	0.05733
FIHr	0.05036
PID %	34.839
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00018787
FICO2	0.0
RMV Data	

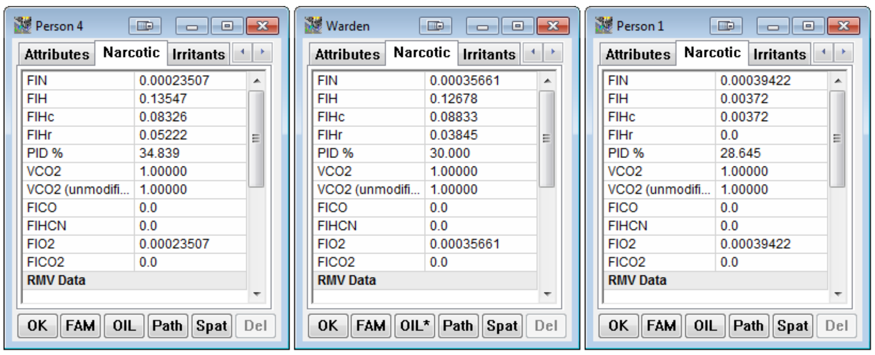
Warden

FIN	0.00030235
FIH	0.09717
FIHc	0.06504
FIHr	0.03212
PID %	30.000
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00030235
FICO2	0.0
RMV Data	

Person 1

FIN	0.00037717
FIH	0.0
FIHc	0.0
FIHr	0.0
PID %	28.645
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00037717
FICO2	0.0
RMV Data	

(a) Scenario 1 with wardens RT 10s delayed.



Person 4

FIN	0.00023507
FIH	0.13547
FIHc	0.08326
FIHr	0.05222
PID %	34.839
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00023507
FICO2	0.0
RMV Data	

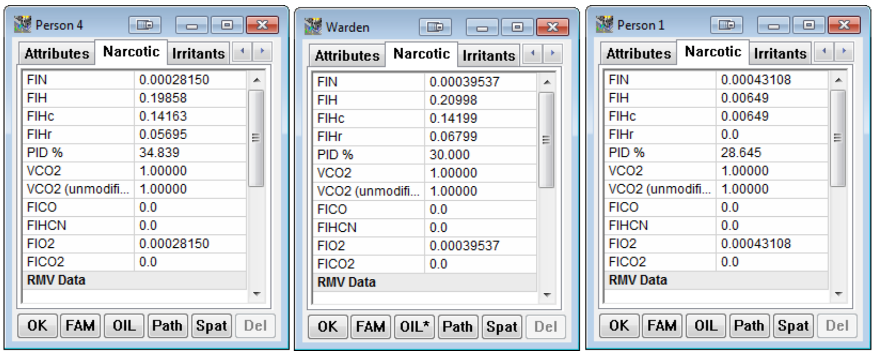
Warden

FIN	0.00035661
FIH	0.12678
FIHc	0.08833
FIHr	0.03845
PID %	30.000
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00035661
FICO2	0.0
RMV Data	

Person 1

FIN	0.00039422
FIH	0.00372
FIHc	0.00372
FIHr	0.0
PID %	28.645
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00039422
FICO2	0.0
RMV Data	

(b) Scenario 1 with wardens RT 20s delayed.



Person 4

FIN	0.00028150
FIH	0.19858
FIHc	0.14163
FIHr	0.05695
PID %	34.839
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00028150
FICO2	0.0
RMV Data	

Warden

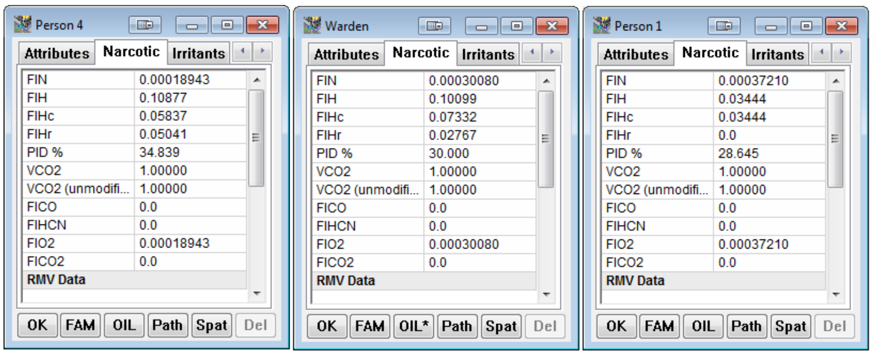
FIN	0.00039537
FIH	0.20998
FIHc	0.14199
FIHr	0.06799
PID %	30.000
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00039537
FICO2	0.0
RMV Data	

Person 1

FIN	0.00043108
FIH	0.00649
FIHc	0.00649
FIHr	0.0
PID %	28.645
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00043108
FICO2	0.0
RMV Data	

(c) Scenario 1 with wardens RT 30s delayed.

Figure A.5: Narcotic properties for person 4, warden and person 1 for scenario 1.



Person 4

FIN	0.00018943
FIH	0.10877
FIHc	0.05837
FIHr	0.05041
PID %	34.839
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00018943
FICO2	0.0
RMV Data	

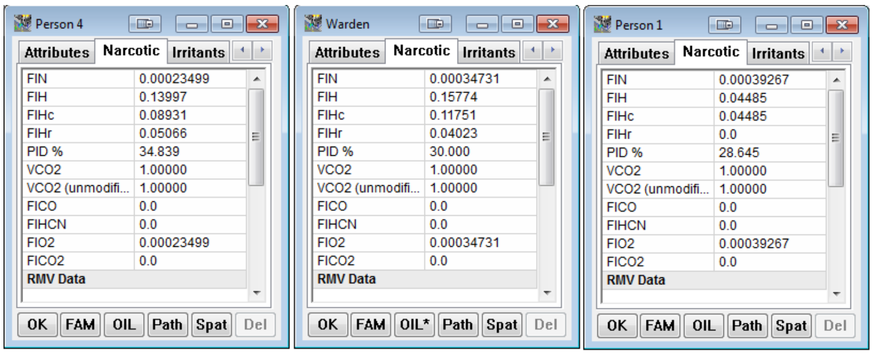
Warden

FIN	0.00030080
FIH	0.10099
FIHc	0.07332
FIHr	0.02767
PID %	30.000
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00030080
FICO2	0.0
RMV Data	

Person 1

FIN	0.00037210
FIH	0.03444
FIHc	0.03444
FIHr	0.0
PID %	28.645
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00037210
FICO2	0.0
RMV Data	

(a) Scenario 2 with wardens RT 10s delayed.



Person 4

FIN	0.00023499
FIH	0.13997
FIHc	0.08931
FIHr	0.05066
PID %	34.839
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00023499
FICO2	0.0
RMV Data	

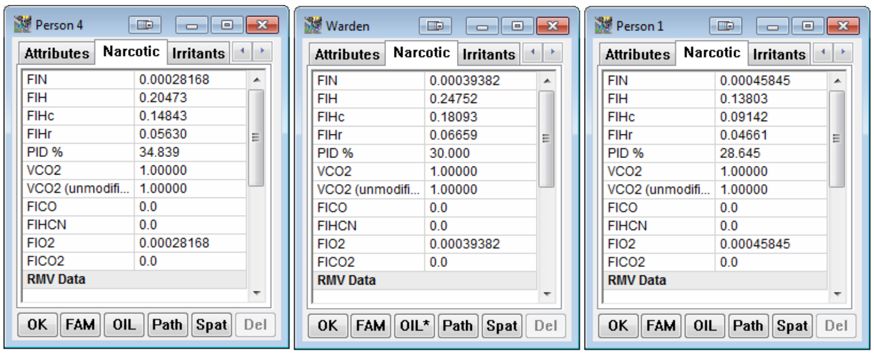
Warden

FIN	0.00034731
FIH	0.15774
FIHc	0.11751
FIHr	0.04023
PID %	30.000
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00034731
FICO2	0.0
RMV Data	

Person 1

FIN	0.00039267
FIH	0.04485
FIHc	0.04485
FIHr	0.0
PID %	28.645
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00039267
FICO2	0.0
RMV Data	

(b) Scenario 2 with wardens RT 20s delayed.



Person 4

FIN	0.00028168
FIH	0.20473
FIHc	0.14843
FIHr	0.05630
PID %	34.839
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00028168
FICO2	0.0
RMV Data	

Warden

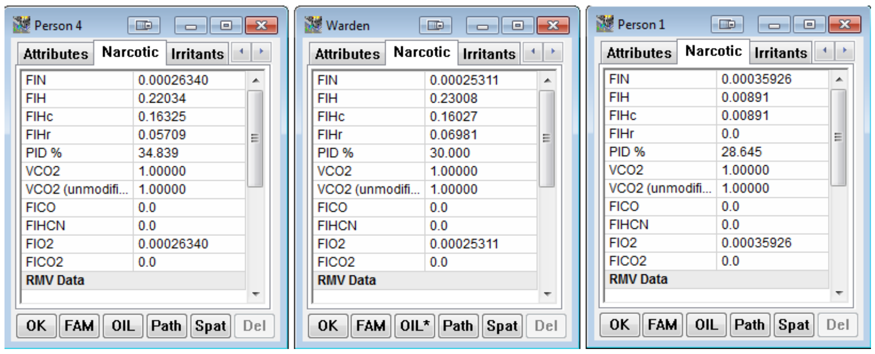
FIN	0.00039382
FIH	0.24752
FIHc	0.18093
FIHr	0.06659
PID %	30.000
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00039382
FICO2	0.0
RMV Data	

Person 1

FIN	0.00045845
FIH	0.13803
FIHc	0.09142
FIHr	0.04661
PID %	28.645
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00045845
FICO2	0.0
RMV Data	

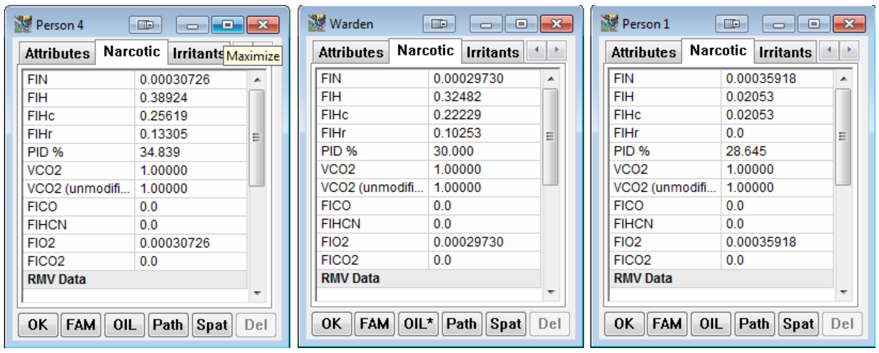
(c) Scenario 2 with wardens RT 30s delayed.

Figure A.6: Narcotic properties for person 4, warden and person 1 for scenario 2.



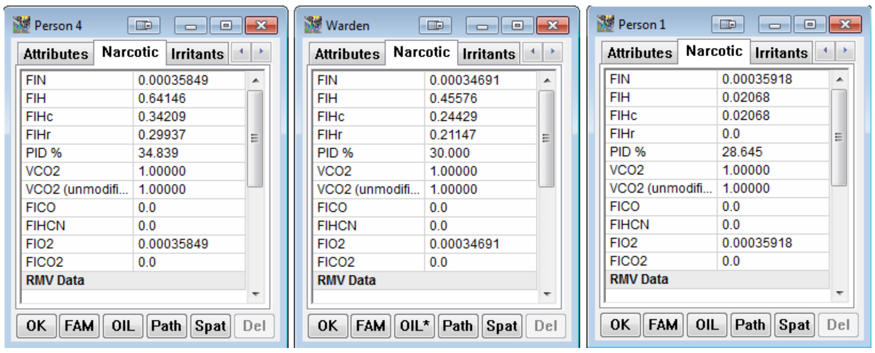
	Person 4	Warden	Person 1
FIN	0.00026340	0.00025311	0.00035926
FIH	0.22034	0.23008	0.00891
FIHc	0.16325	0.16027	0.00891
FIHr	0.05709	0.06981	0.0
PID %	34.839	30.000	28.645
VCO2	1.00000	1.00000	1.00000
VCO2 (unmodifi...	1.00000	1.00000	1.00000
FICO	0.0	0.0	0.0
FIHCN	0.0	0.0	0.0
FIO2	0.00026340	0.00025311	0.00035926
FICO2	0.0	0.0	0.0
RMV Data			

(a) Scenario 3 with wardens RT 10s delayed.



	Person 4	Warden	Person 1
FIN	0.00030726	0.00029730	0.00035918
FIH	0.38924	0.32482	0.02053
FIHc	0.25619	0.22229	0.02053
FIHr	0.13305	0.10253	0.0
PID %	34.839	30.000	28.645
VCO2	1.00000	1.00000	1.00000
VCO2 (unmodifi...	1.00000	1.00000	1.00000
FICO	0.0	0.0	0.0
FIHCN	0.0	0.0	0.0
FIO2	0.00030726	0.00029730	0.00035918
FICO2	0.0	0.0	0.0
RMV Data			

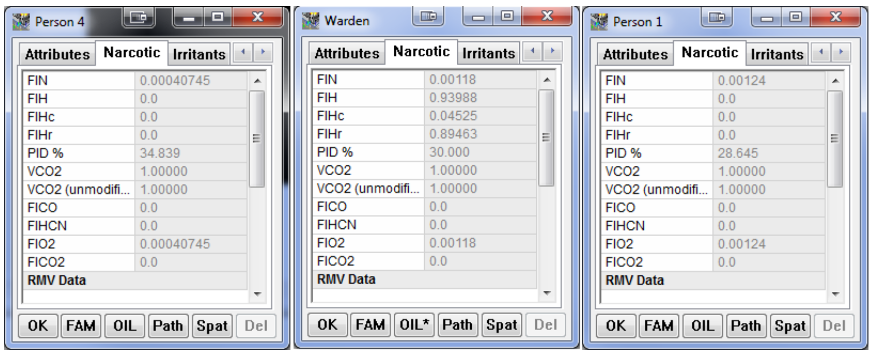
(b) Scenario 3 with wardens RT 20s delayed.



	Person 4	Warden	Person 1
FIN	0.00035849	0.00034691	0.00035918
FIH	0.64146	0.45576	0.02068
FIHc	0.34209	0.24429	0.02068
FIHr	0.29937	0.21147	0.0
PID %	34.839	30.000	28.645
VCO2	1.00000	1.00000	1.00000
VCO2 (unmodifi...	1.00000	1.00000	1.00000
FICO	0.0	0.0	0.0
FIHCN	0.0	0.0	0.0
FIO2	0.00035849	0.00034691	0.00035918
FICO2	0.0	0.0	0.0
RMV Data			

(c) Scenario 3 with wardens RT 30s delayed.

Figure A.7: Narcotic properties for person 4, warden and person 1 for scenario 3.



Person 4

FIN	0.00040745
FIH	0.0
FIHc	0.0
FIHr	0.0
PID %	34.839
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00040745
FICO2	0.0
RMV Data	

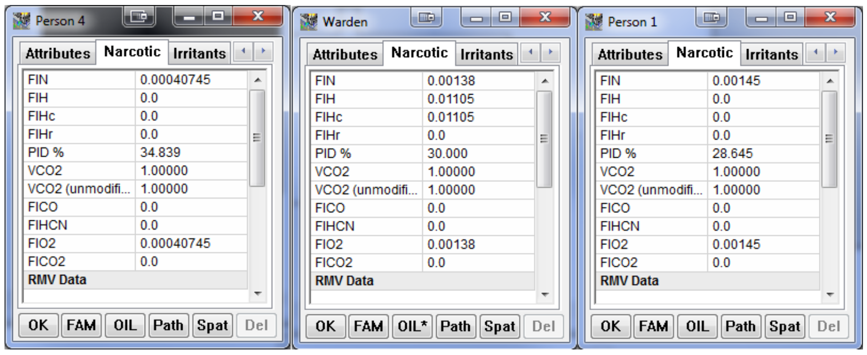
Warden

FIN	0.00118
FIH	0.93988
FIHc	0.04525
FIHr	0.89463
PID %	30.000
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00118
FICO2	0.0
RMV Data	

Person 1

FIN	0.00124
FIH	0.0
FIHc	0.0
FIHr	0.0
PID %	28.645
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00124
FICO2	0.0
RMV Data	

(a) Scenario 1 with 60s delay for warden, non coupled.



Person 4

FIN	0.00040745
FIH	0.0
FIHc	0.0
FIHr	0.0
PID %	34.839
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00040745
FICO2	0.0
RMV Data	

Warden

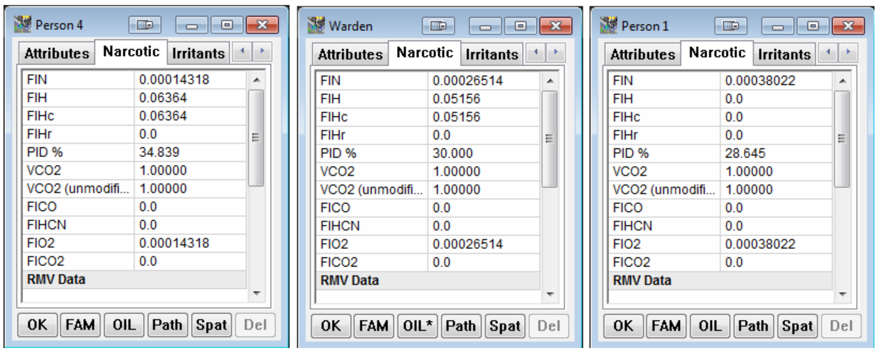
FIN	0.00138
FIH	0.01105
FIHc	0.01105
FIHr	0.0
PID %	30.000
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00138
FICO2	0.0
RMV Data	

Person 1

FIN	0.00145
FIH	0.0
FIHc	0.0
FIHr	0.0
PID %	28.645
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00145
FICO2	0.0
RMV Data	

(b) Scenario 1 with 60s delay for warden, coupled.

Figure A.8: Properties for person 4, warden and person 1.



Person 4

FIN	0.00014318
FIH	0.06364
FIHc	0.06364
FIHr	0.0
PID %	34.839
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00014318
FICO2	0.0
RMV Data	

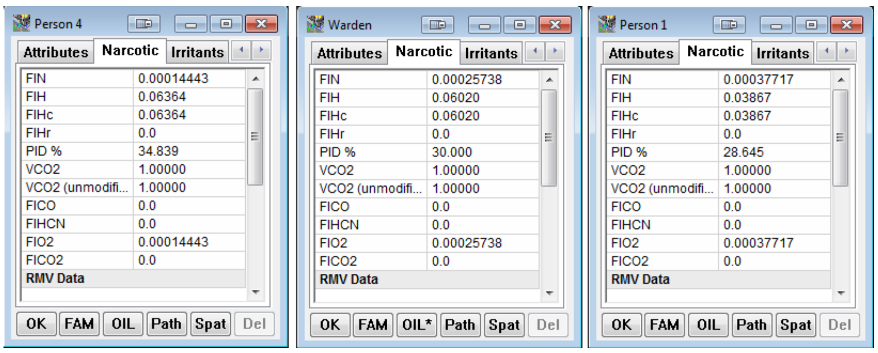
Warden

FIN	0.00026514
FIH	0.05156
FIHc	0.05156
FIHr	0.0
PID %	30.000
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00026514
FICO2	0.0
RMV Data	

Person 1

FIN	0.00038022
FIH	0.0
FIHc	0.0
FIHr	0.0
PID %	28.645
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00038022
FICO2	0.0
RMV Data	

(a) Person one's, four's and the warden's properties after evacuation for scenario 1.



Person 4

FIN	0.00014443
FIH	0.06364
FIHc	0.06364
FIHr	0.0
PID %	34.839
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00014443
FICO2	0.0
RMV Data	

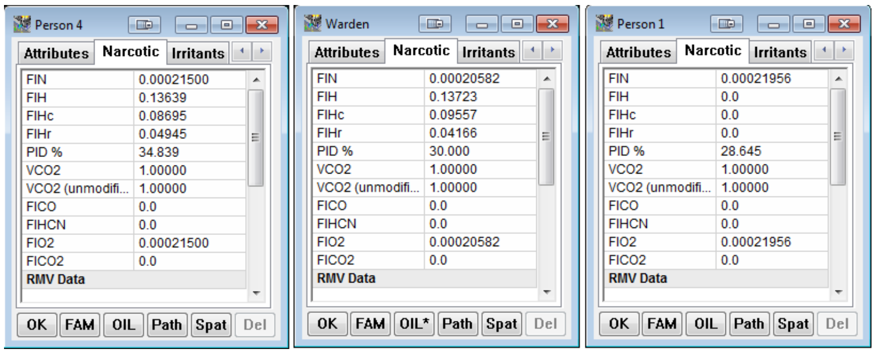
Warden

FIN	0.00025738
FIH	0.06020
FIHc	0.06020
FIHr	0.0
PID %	30.000
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00025738
FICO2	0.0
RMV Data	

Person 1

FIN	0.00037717
FIH	0.03867
FIHc	0.03867
FIHr	0.0
PID %	28.645
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00037717
FICO2	0.0
RMV Data	

(b) Person one's, four's and the warden's properties after evacuation for scenario 2.



Person 4

FIN	0.00021500
FIH	0.13639
FIHc	0.08695
FIHr	0.04945
PID %	34.839
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00021500
FICO2	0.0
RMV Data	

Warden

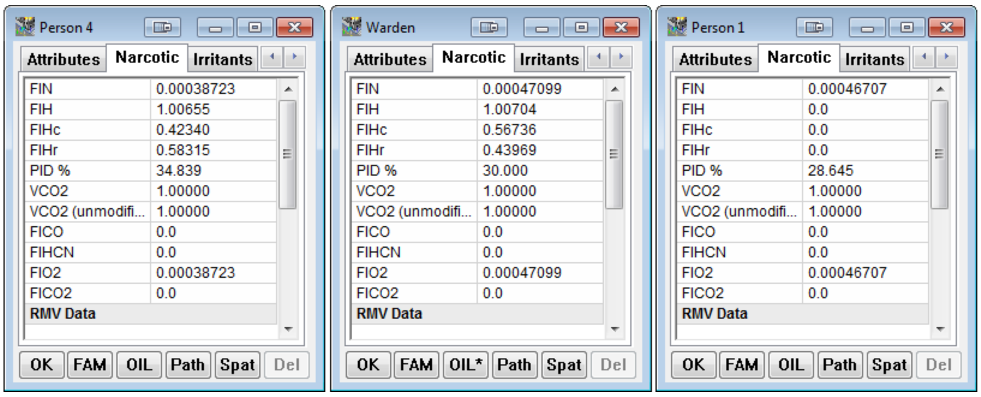
FIN	0.00020582
FIH	0.13723
FIHc	0.09557
FIHr	0.04166
PID %	30.000
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00020582
FICO2	0.0
RMV Data	

Person 1

FIN	0.00021956
FIH	0.0
FIHc	0.0
FIHr	0.0
PID %	28.645
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00021956
FICO2	0.0
RMV Data	

(c) Person one's, four's and the warden's properties after evacuation for scenario 3.

Figure A.9: Narcotic properties for person 4, warden and person 1 for scenario one to three.

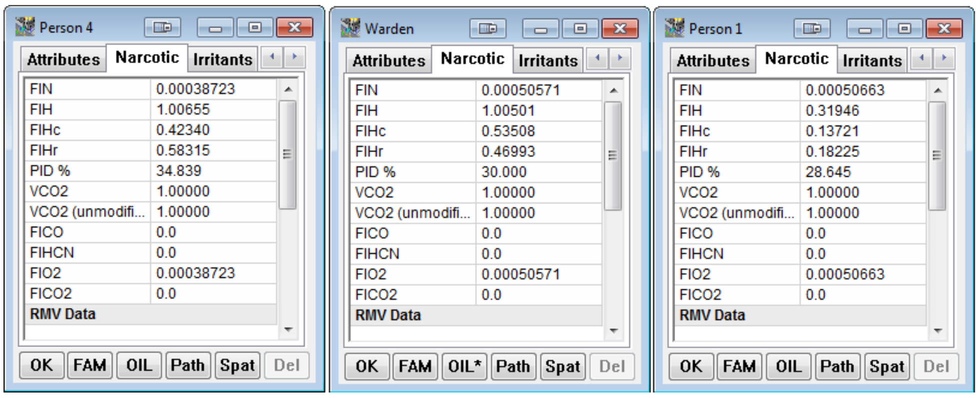


FIN	0.00038723
FIH	1.00655
FIHc	0.42340
FIHr	0.58315
PID %	34.839
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00038723
FICO2	0.0
RMV Data	

FIN	0.00047099
FIH	1.00704
FIHc	0.56736
FIHr	0.43969
PID %	30.000
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00047099
FICO2	0.0
RMV Data	

FIN	0.00046707
FIH	0.0
FIHc	0.0
FIHr	0.0
PID %	28.645
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00046707
FICO2	0.0
RMV Data	

(a) Scenario 1 with 75s delay for warden.



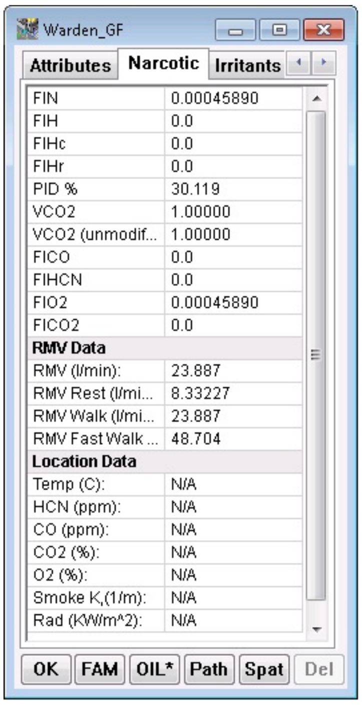
FIN	0.00038723
FIH	1.00655
FIHc	0.42340
FIHr	0.58315
PID %	34.839
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00038723
FICO2	0.0
RMV Data	

FIN	0.00050571
FIH	1.00501
FIHc	0.53508
FIHr	0.46993
PID %	30.000
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00050571
FICO2	0.0
RMV Data	

FIN	0.00050663
FIH	0.31946
FIHc	0.13721
FIHr	0.18225
PID %	28.645
VCO2	1.00000
VCO2 (unmodifi...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00050663
FICO2	0.0
RMV Data	

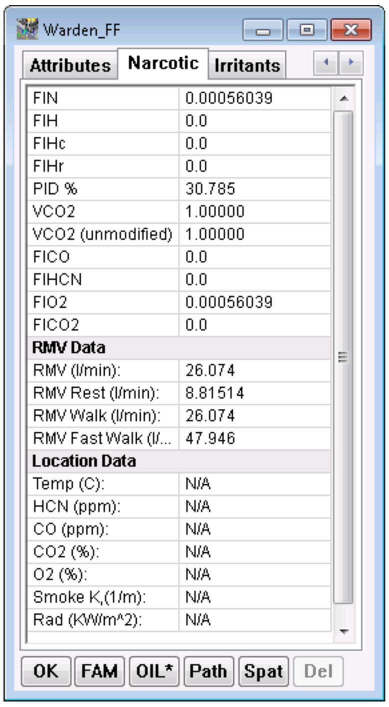
(b) Scenario 3 with 75s delay for the warden.

Figure A.10: Properties for person 4, warden and person 1.



FIN	0.00045890
FIH	0.0
FIHc	0.0
FIHr	0.0
PID %	30.119
VCO2	1.00000
VCO2 (unmodif...	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00045890
FICO2	0.0
RMV Data	
RMV (l/min):	23.887
RMV Rest (l/mi...	8.33227
RMV Walk (l/mi...	23.887
RMV Fast Walk ...	48.704
Location Data	
Temp (C):	N/A
HCN (ppm):	N/A
CO (ppm):	N/A
CO2 (%):	N/A
O2 (%):	N/A
Smoke K,(1/m):	N/A
Rad (KW/m^2):	N/A

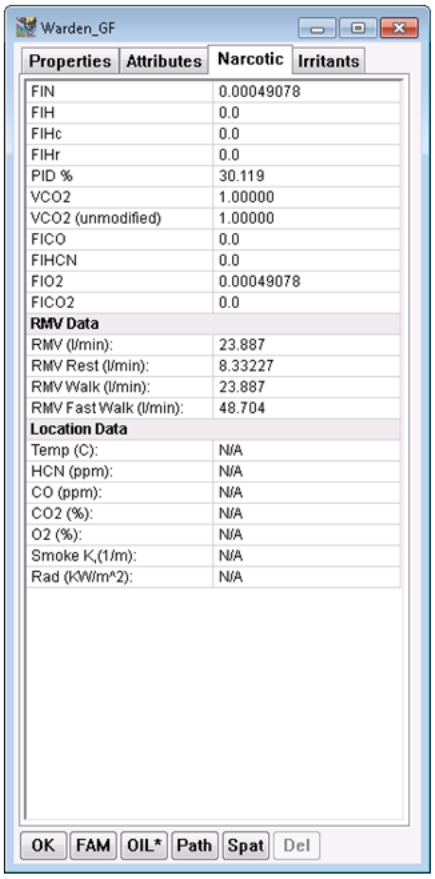
(a) Warden evacuating the ground floor property outputs.



FIN	0.00056039
FIH	0.0
FIHc	0.0
FIHr	0.0
PID %	30.785
VCO2	1.00000
VCO2 (unmodified)	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00056039
FICO2	0.0
RMV Data	
RMV (l/min):	26.074
RMV Rest (l/min):	8.81514
RMV Walk (l/min):	26.074
RMV Fast Walk (l/...	47.946
Location Data	
Temp (C):	N/A
HCN (ppm):	N/A
CO (ppm):	N/A
CO2 (%):	N/A
O2 (%):	N/A
Smoke K,(1/m):	N/A
Rad (KW/m^2):	N/A

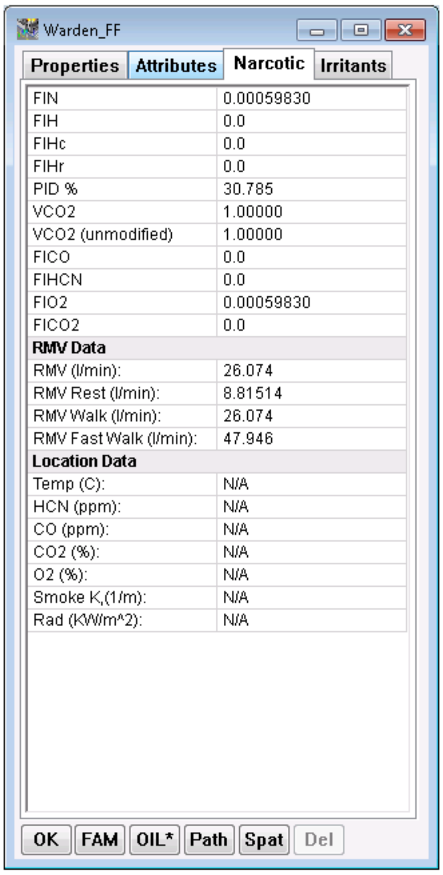
(b) Warden evacuating the first floor property outputs.

Figure A.11: Property outputs for both wardens, ground floor and first floor for scenario 3.



FIN	0.00049078
FIH	0.0
FIHc	0.0
FIHr	0.0
PID %	30.119
VCO2	1.00000
VCO2 (unmodified)	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00049078
FICO2	0.0
RMV Data	
RMV (l/min):	23.887
RMV Rest (l/min):	8.33227
RMV Walk (l/min):	23.887
RMV Fast Walk (l/min):	48.704
Location Data	
Temp (C):	N/A
HCN (ppm):	N/A
CO (ppm):	N/A
CO2 (%):	N/A
O2 (%):	N/A
Smoke K,(1/m):	N/A
Rad (KW/m^2):	N/A

(a) Warden evacuating the ground floor property outputs.

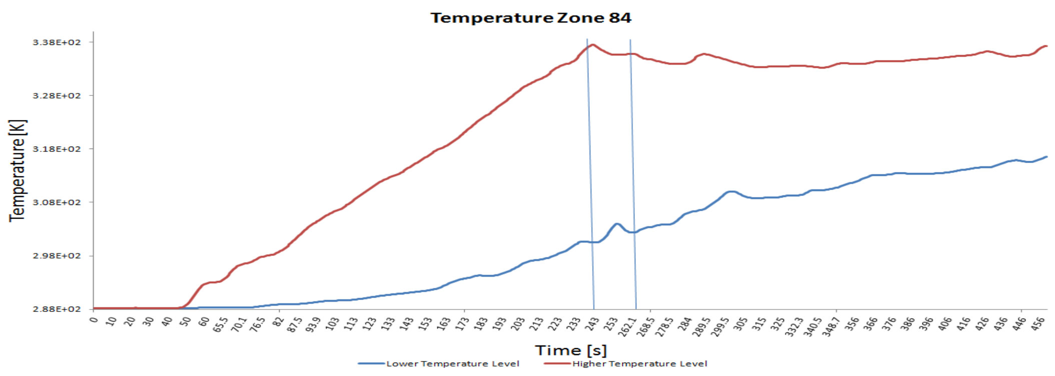


FIN	0.00059830
FIH	0.0
FIHc	0.0
FIHr	0.0
PID %	30.785
VCO2	1.00000
VCO2 (unmodified)	1.00000
FICO	0.0
FIHCN	0.0
FIO2	0.00059830
FICO2	0.0
RMV Data	
RMV (l/min):	26.074
RMV Rest (l/min):	8.81514
RMV Walk (l/min):	26.074
RMV Fast Walk (l/min):	47.946
Location Data	
Temp (C):	N/A
HCN (ppm):	N/A
CO (ppm):	N/A
CO2 (%):	N/A
O2 (%):	N/A
Smoke K,(1/m):	N/A
Rad (KW/m^2):	N/A

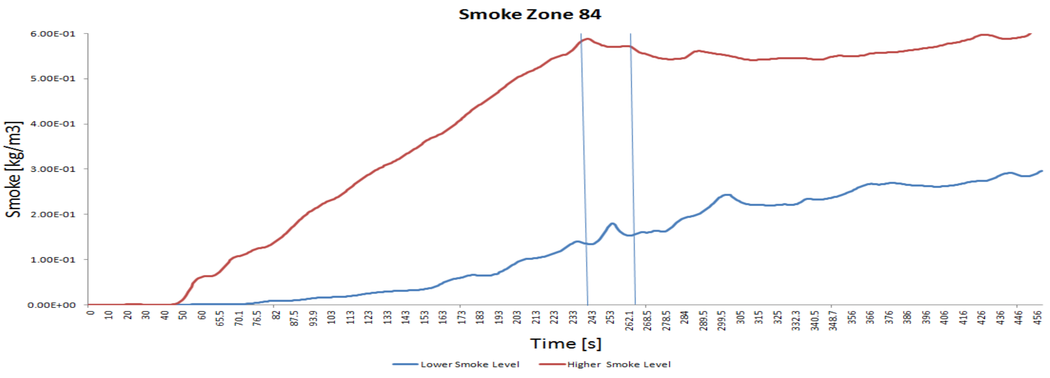
(b) Warden evacuating the first floor property outputs.

Figure A.12: Property outputs for both wardens, ground floor and first floor for scenario 3.

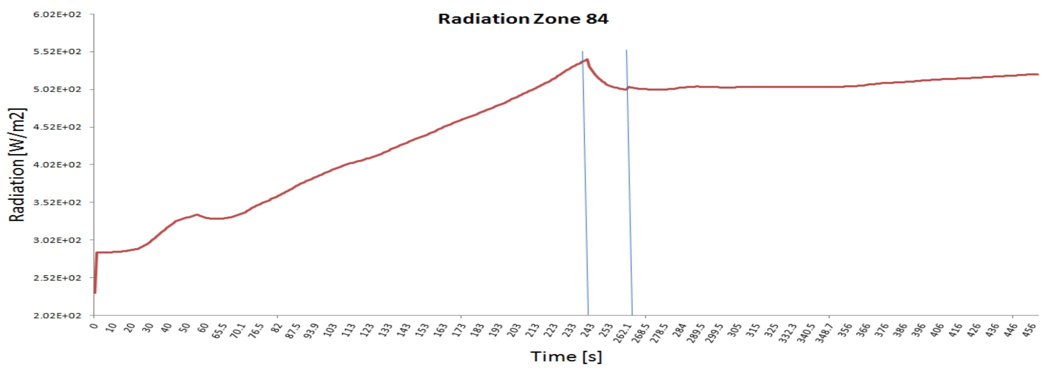
A.7 Scenario 1



(a) Temperature outputs and door events for zone 84 in scenario 1.

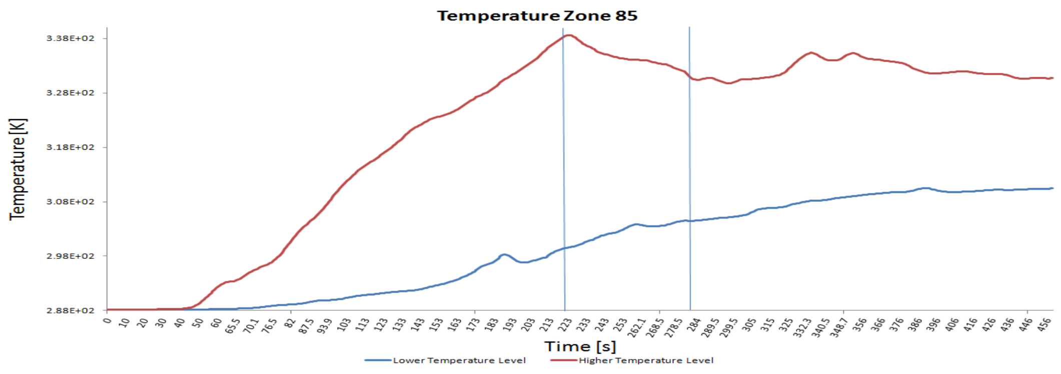


(b) Smoke outputs and door events for zone 84 in scenario 1.

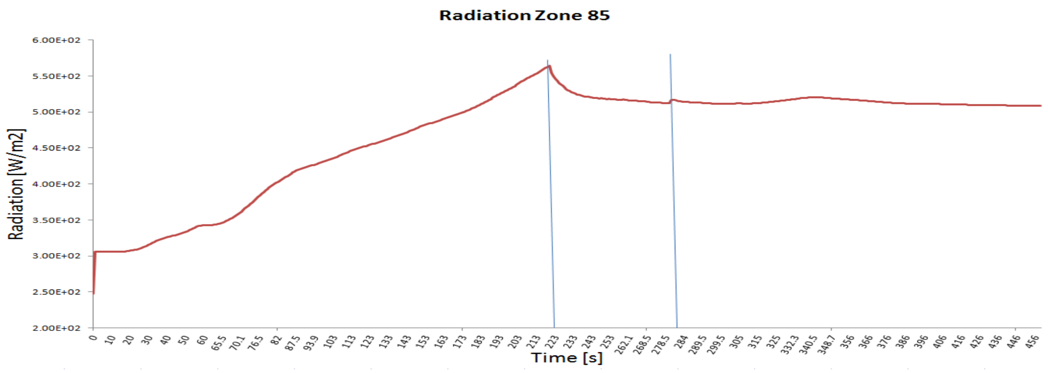


(c) Radiation outputs and door events for zone 84 in scenario 1.

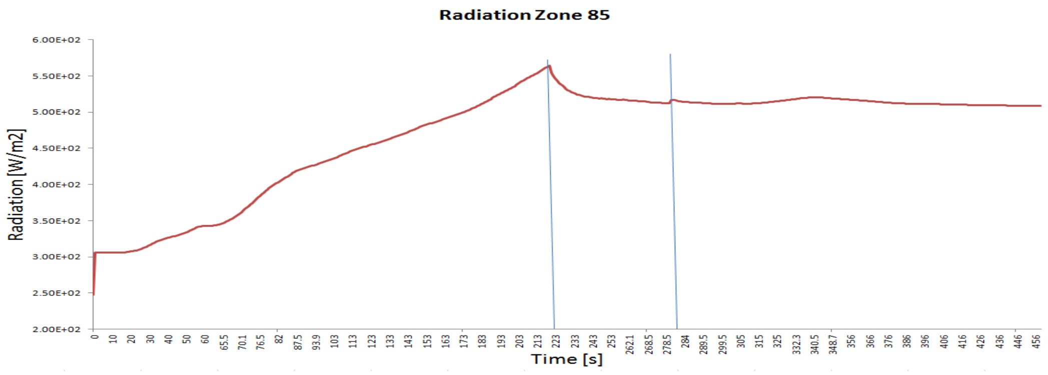
Figure A.13: Temperature, smoke and radiation vs. time for zone 84.



(a) Temperature outputs and door events for zone 85 in scenario 1.

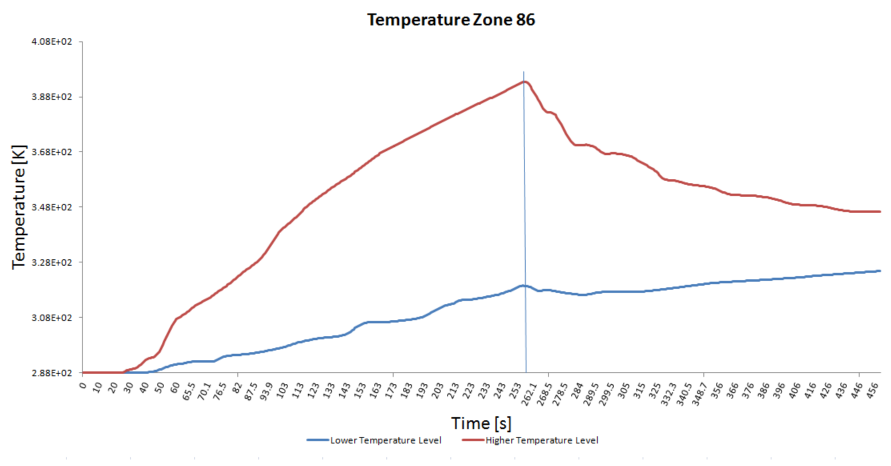


(b) Smoke outputs and door events for zone 85 in scenario 1.

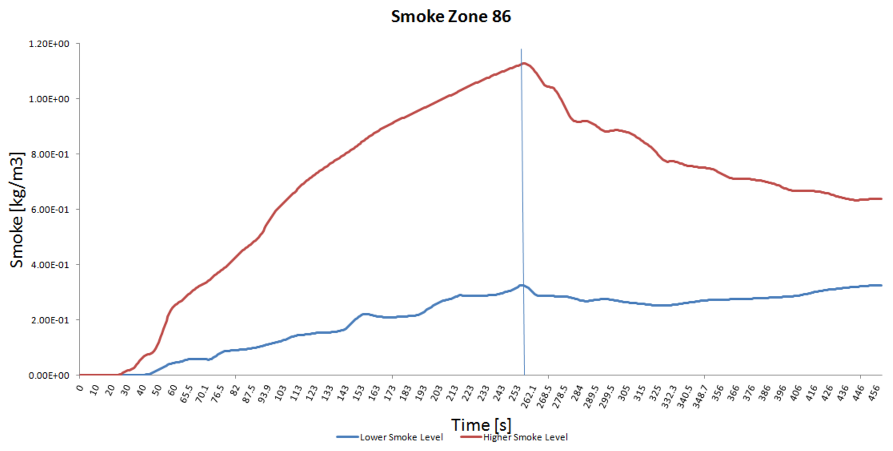


(c) Radiation outputs and door events for zone 85 in scenario 1.

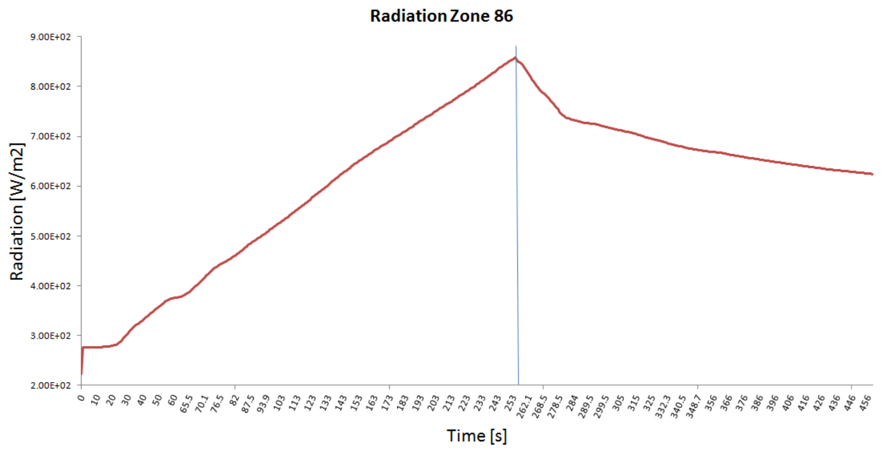
Figure A.14: Temperature, smoke and radiation vs. time for zone 85.



(a) Temperature outputs and door events for zone 86 in scenario 1.

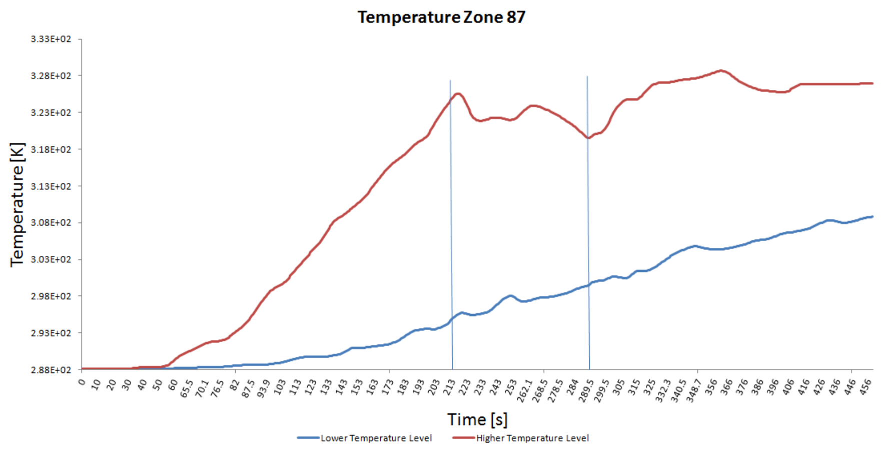


(b) Smoke outputs and door events for zone 86 in scenario 1.

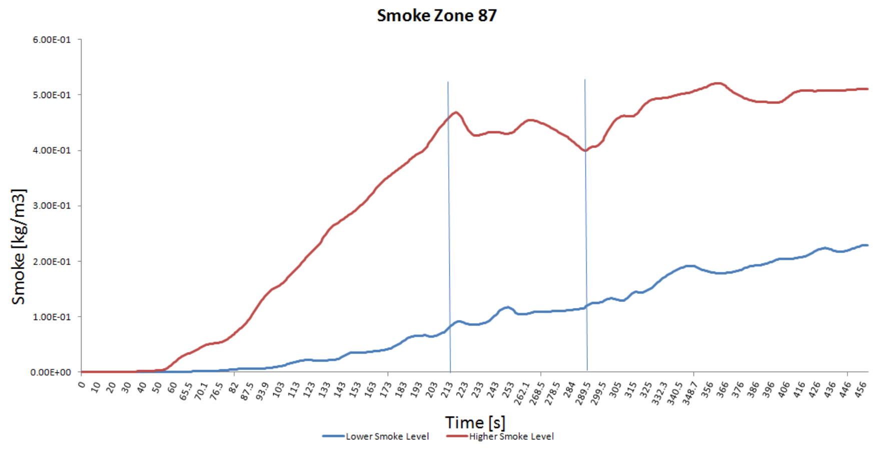


(c) Radiation outputs and door events for zone 86 in scenario 1.

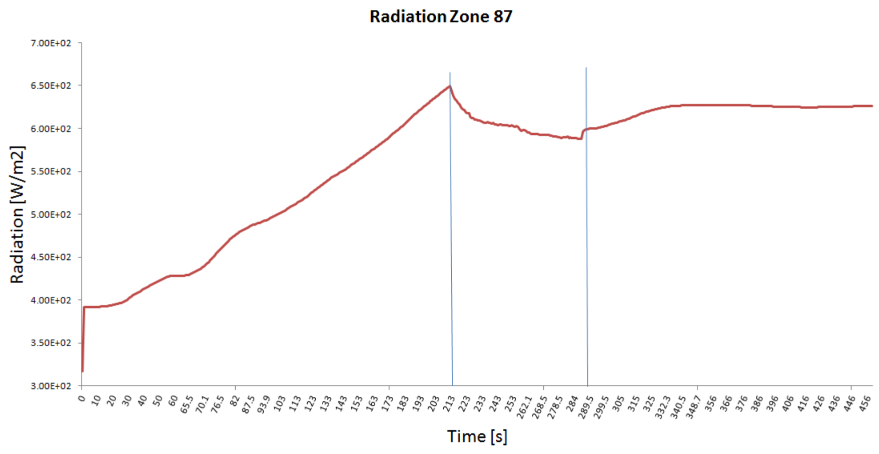
Figure A.15: Temperature, smoke and radiation vs. time for zone 86.



(a) Temperature outputs and door events for zone 87 in scenario 1.

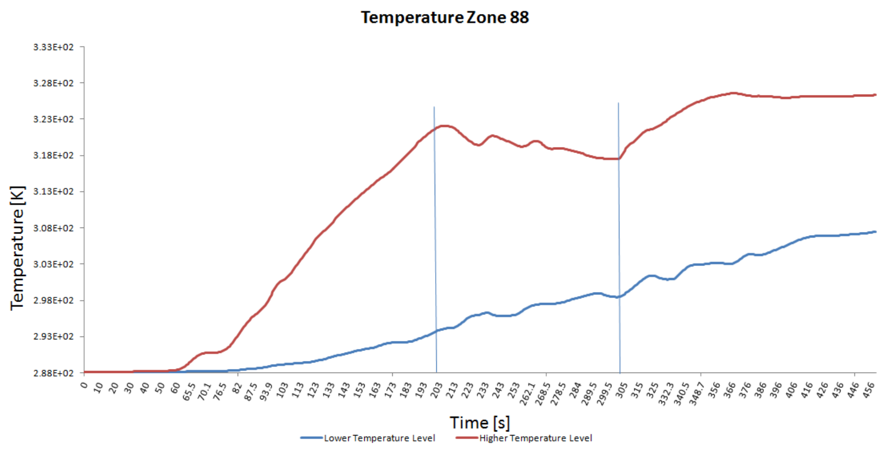


(b) Smoke outputs and door events for zone 87 in scenario 1.

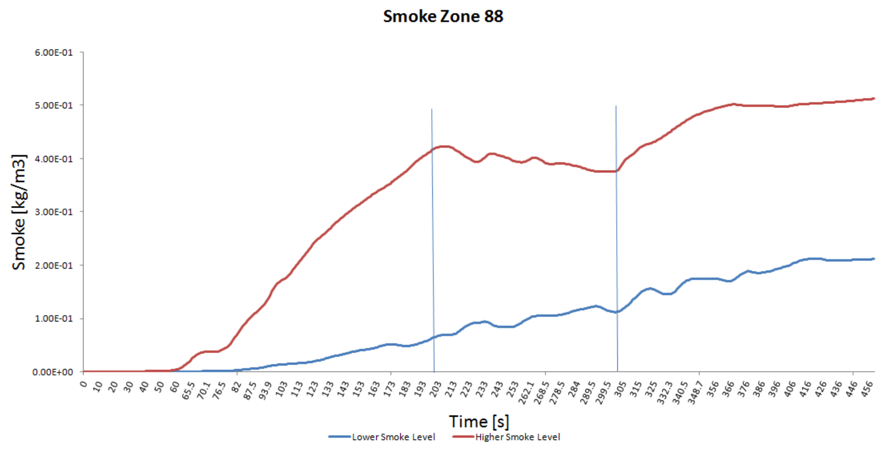


(c) Radiation outputs and door events for zone 87 in scenario 1.

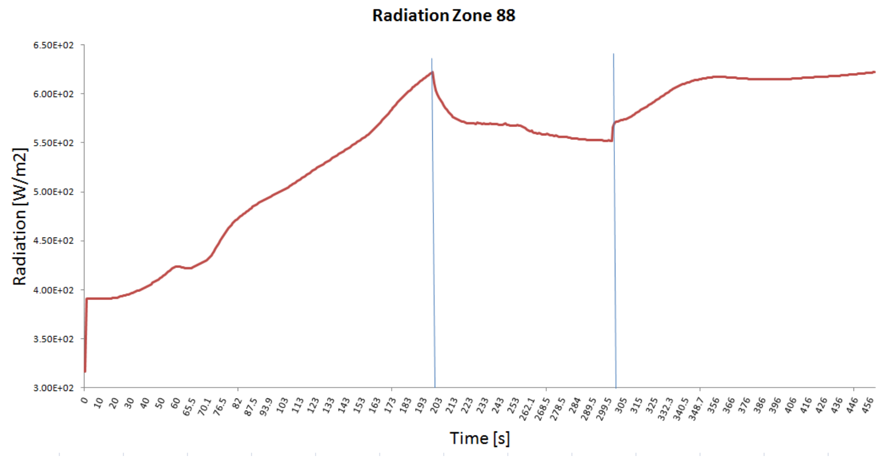
Figure A.16: Temperature, smoke and radiation vs. time for zone 87.



(a) Temperature outputs and door events for zone 88 in scenario 1.

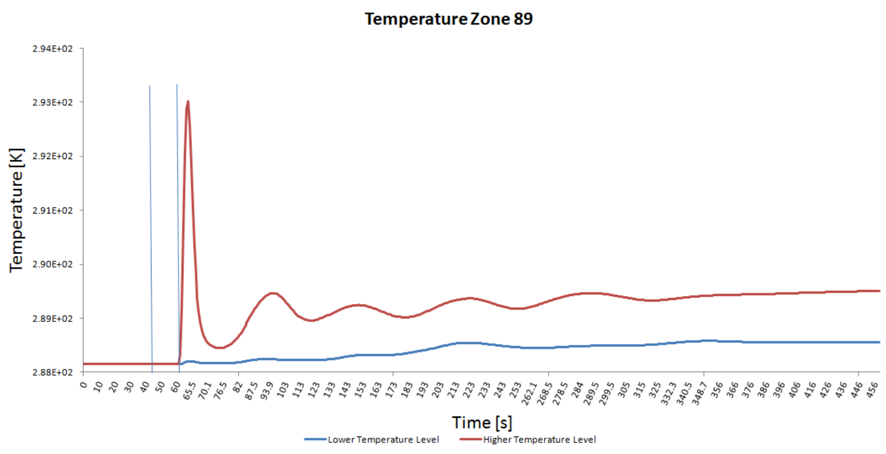


(b) Smoke outputs and door events for zone 88 in scenario 1.

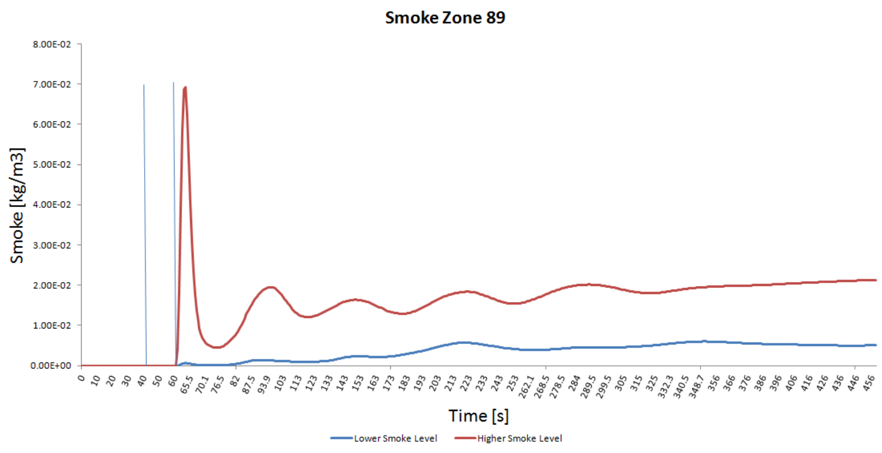


(c) Radiation outputs and door events for zone 88 in scenario 1.

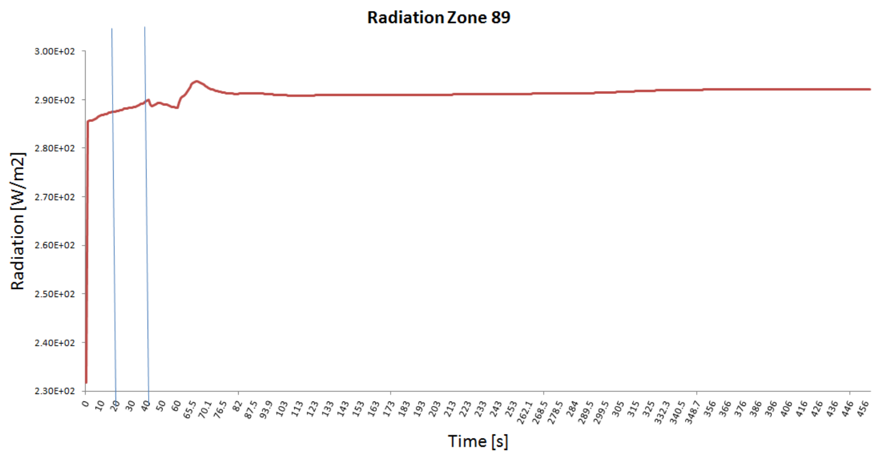
Figure A.17: Temperature, smoke and radiation vs. time for zone 88.



(a) Temperature outputs and door events for zone 89 in scenario 1.

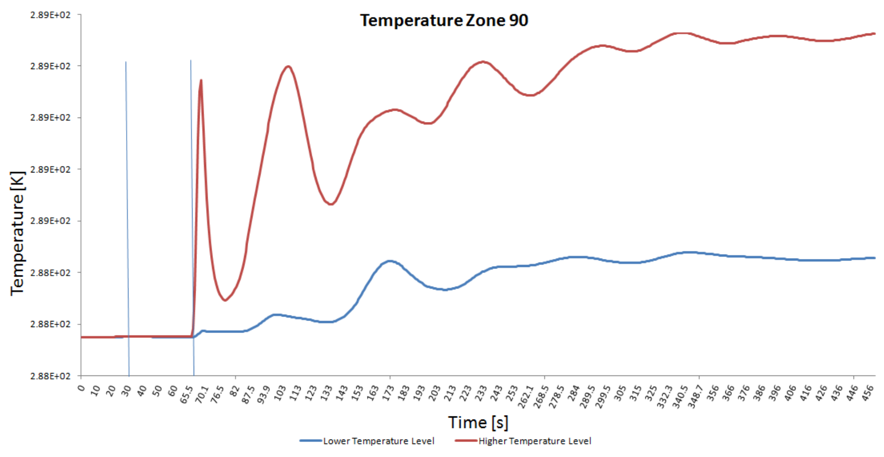


(b) Smoke outputs and door events for zone 89 in scenario 1.

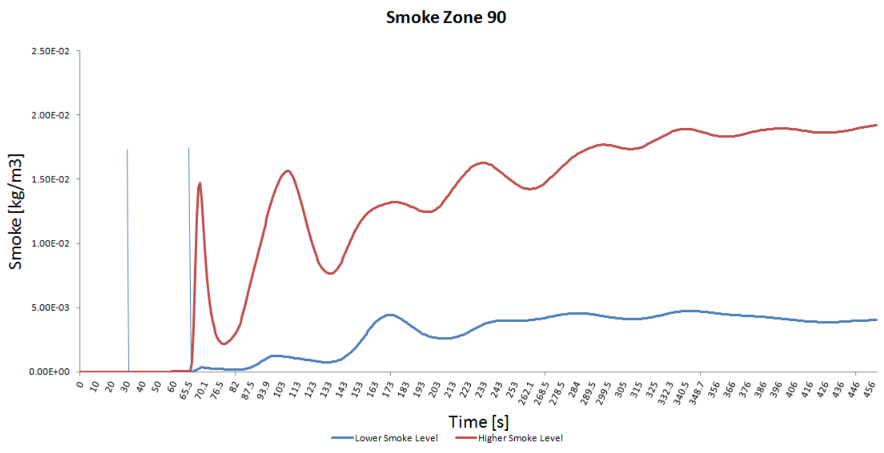


(c) Radiation outputs and door events for zone 89 in scenario 1.

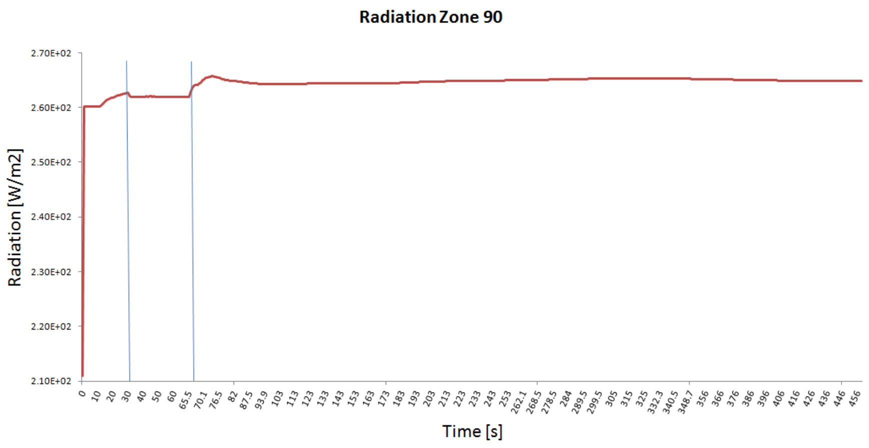
Figure A.18: Temperature, smoke and radiation vs. time for zone 89.



(a) Temperature outputs and door events for zone 90 in scenario 1.

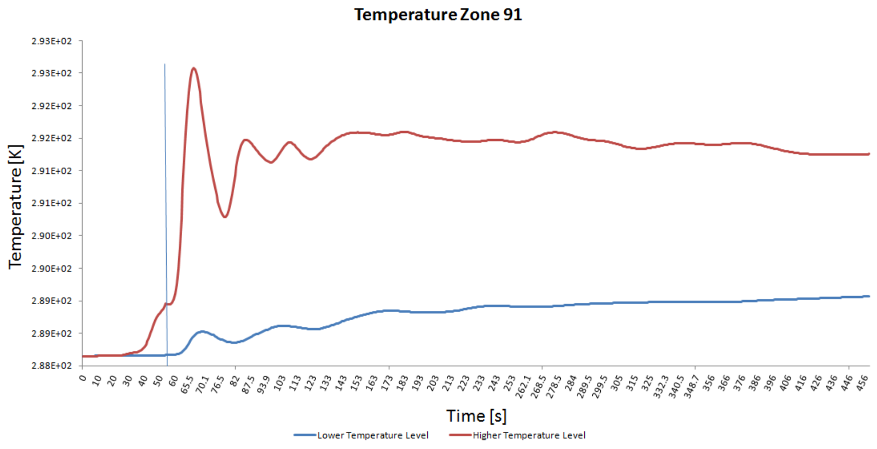


(b) Smoke outputs and door events for zone 90 in scenario 1.

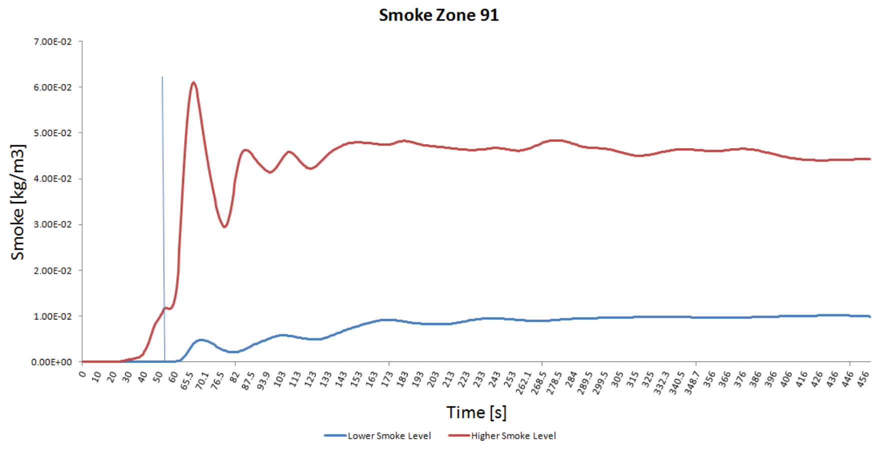


(c) Radiation outputs and door events for zone 90 in scenario 1.

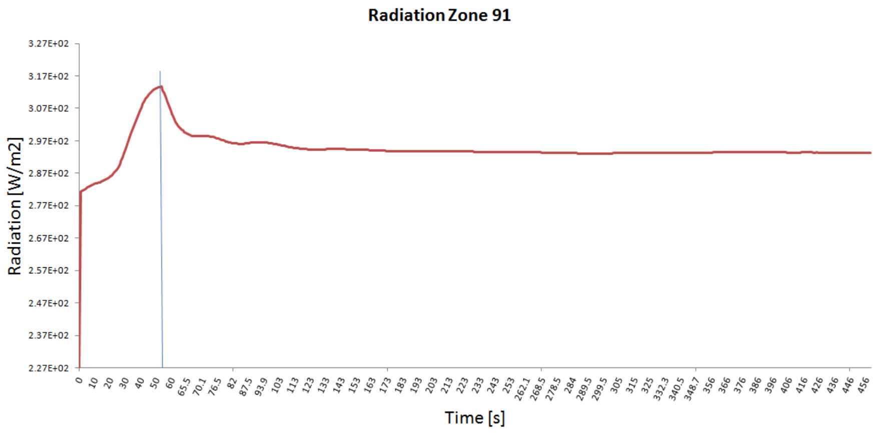
Figure A.19: Temperature, smoke and radiation vs. time for zone 90.



(a) Temperature outputs and door events for zone 91 in scenario 1.

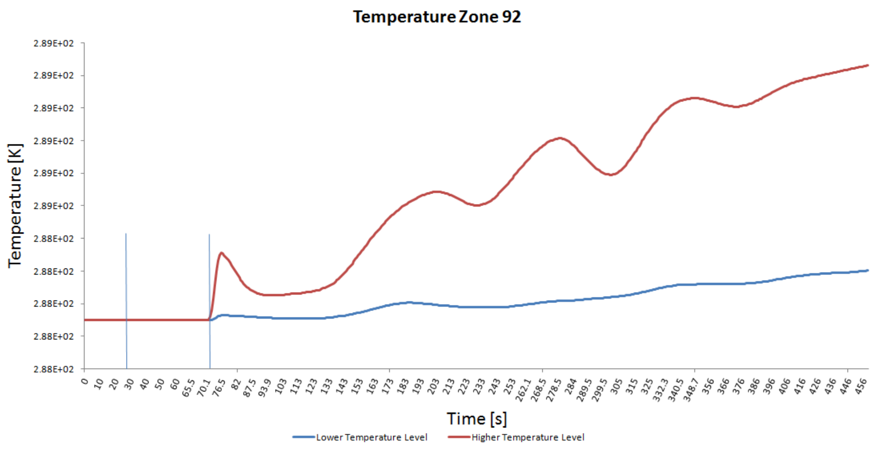


(b) Smoke outputs and door events for zone 91 in scenario 1.

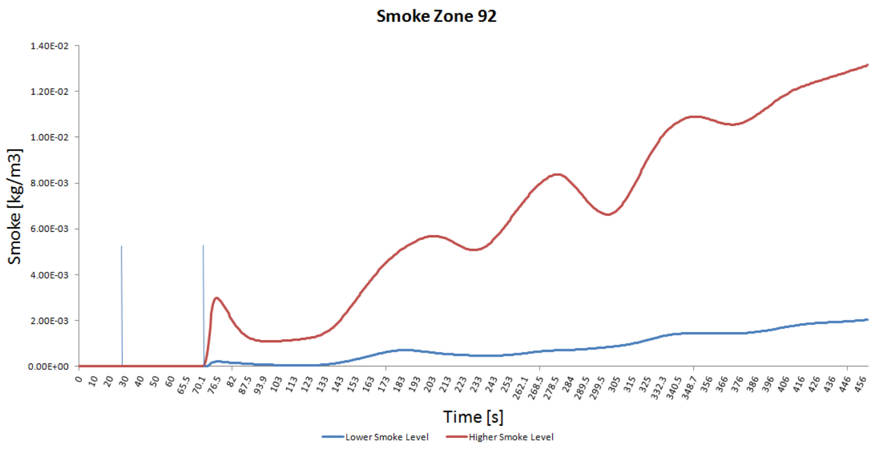


(c) Radiation outputs and door events for zone 91 in scenario 1.

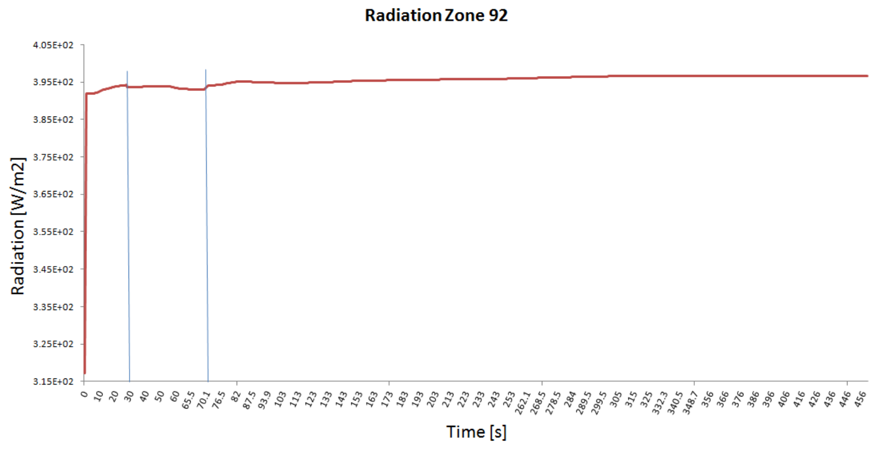
Figure A.20: Temperature, smoke and radiation vs. time for zone 91.



(a) Temperature outputs and door events for zone 92 in scenario 1.

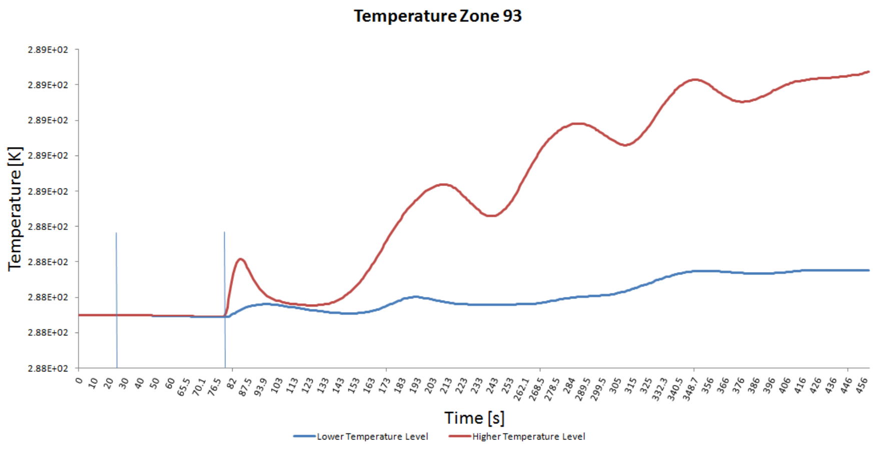


(b) Smoke outputs and door events for zone 92 in scenario 1.

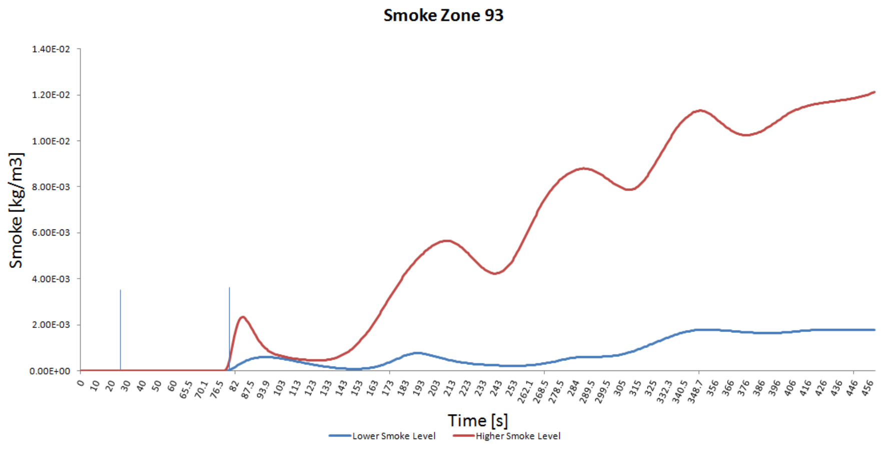


(c) Radiation outputs and door events for zone 92 in scenario 1.

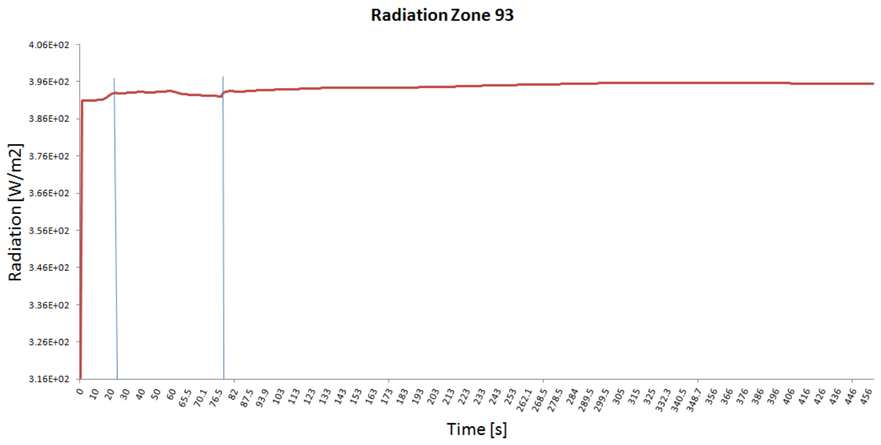
Figure A.21: Temperature, smoke and radiation vs. time for zone 92.



(a) Temperature outputs and door events for zone 93 in scenario 1.



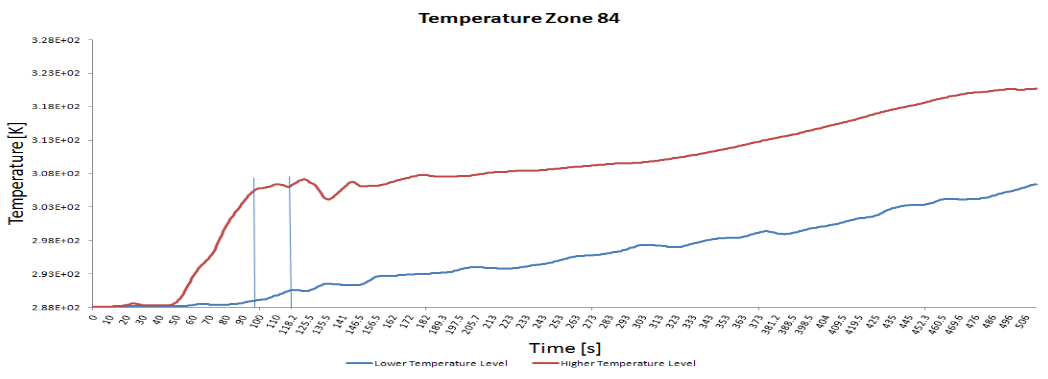
(b) Smoke outputs and door events for zone 93 in scenario 1.



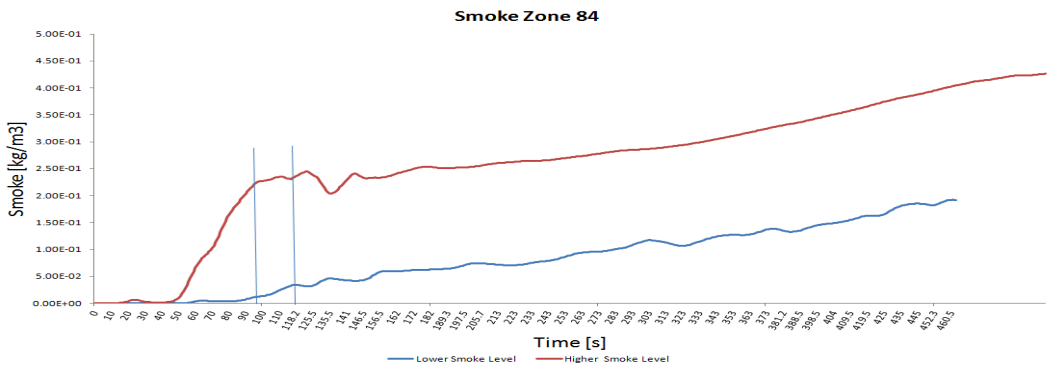
(c) Radiation outputs and door events for zone 93 in scenario 1.

Figure A.22: Temperature, smoke and radiation vs. time for zone 93.

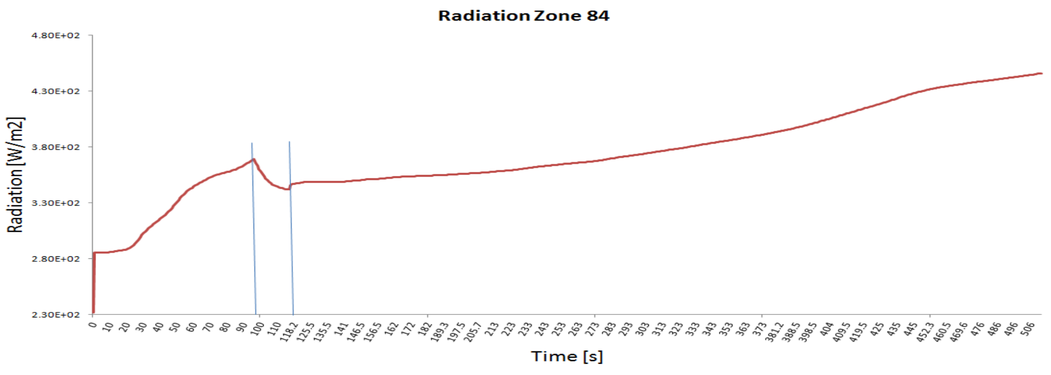
Scenario 2



(a) Temperature outputs and door events for zone 84 in scenario 2.

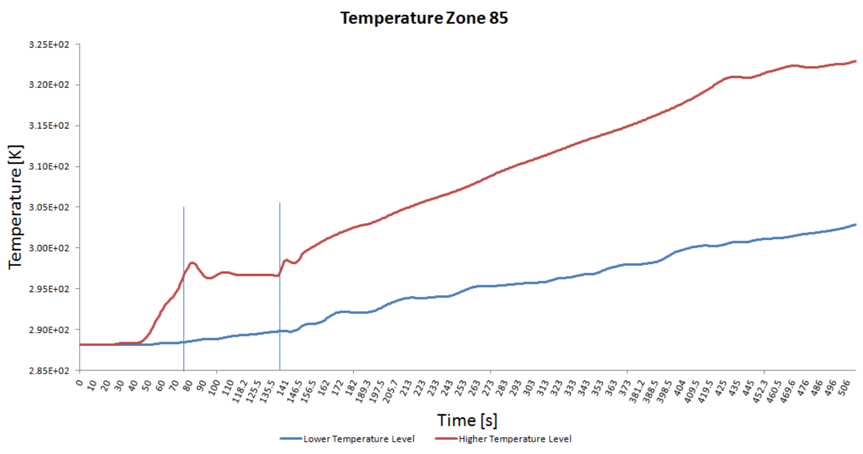


(b) Smoke outputs and door events for zone 84 in scenario 2.

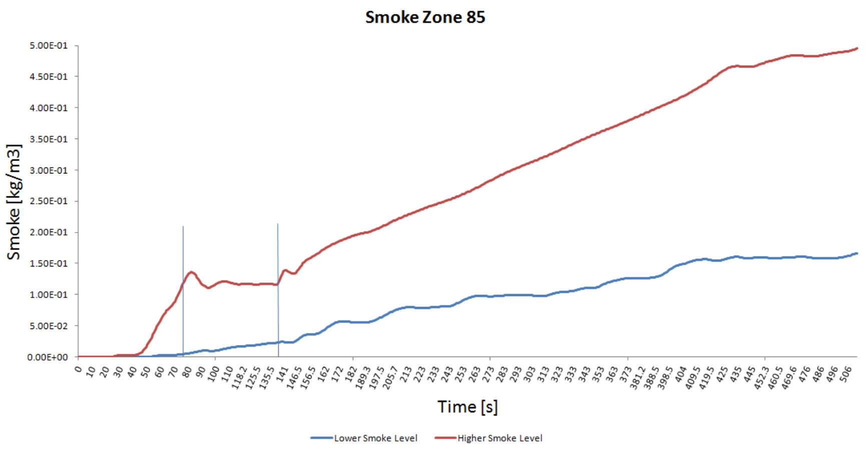


(c) Radiation outputs and door events for zone 84 in scenario 2.

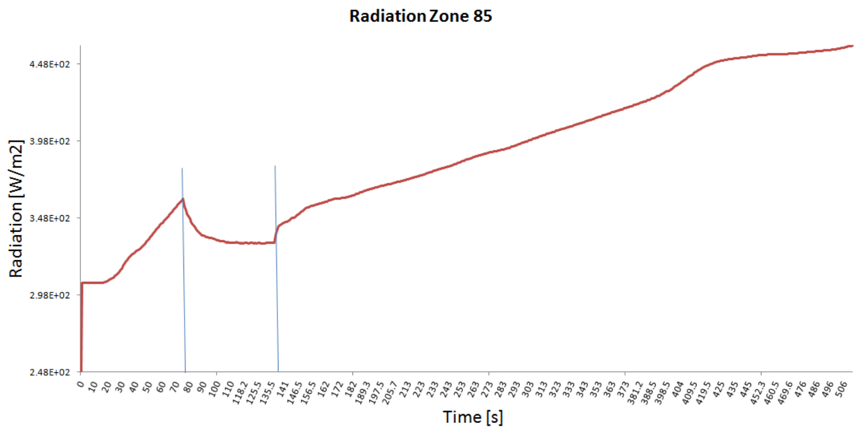
Figure A.23: Temperature, smoke and radiation vs. time for zone 84.



(a) Temperature outputs and door events for zone 85 in scenario 2.

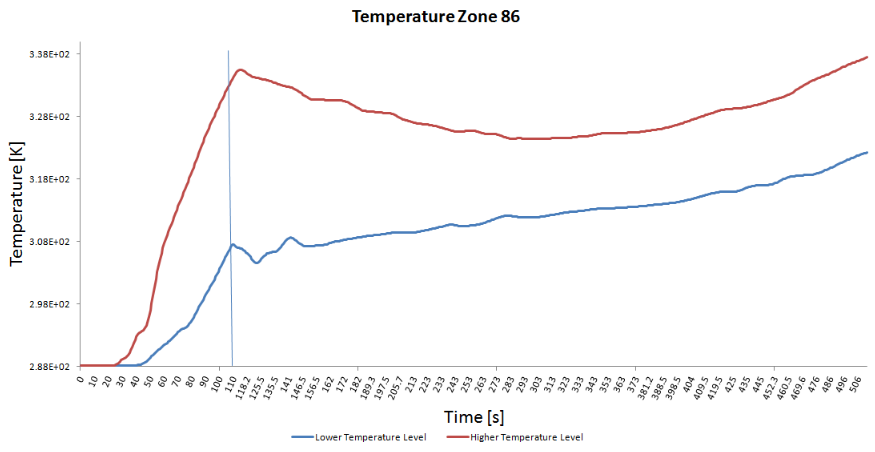


(b) Smoke outputs and door events for zone 85 in scenario 2.

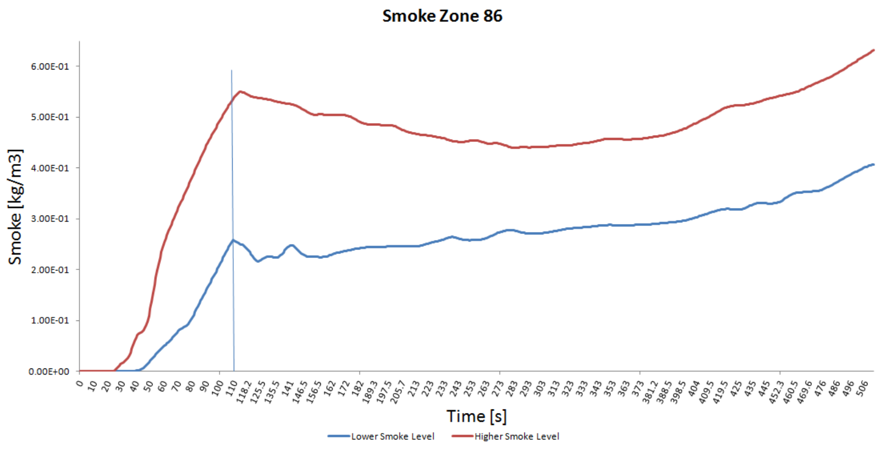


(c) Radiation outputs and door events for zone 85 in scenario 2.

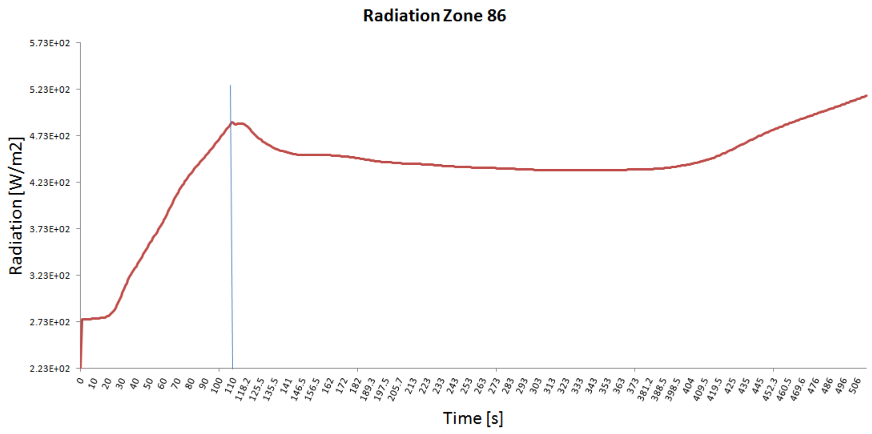
Figure A.24: Temperature, smoke and radiation vs. time for zone 85.



(a) Temperature outputs and door events for zone 86 in scenario 2.

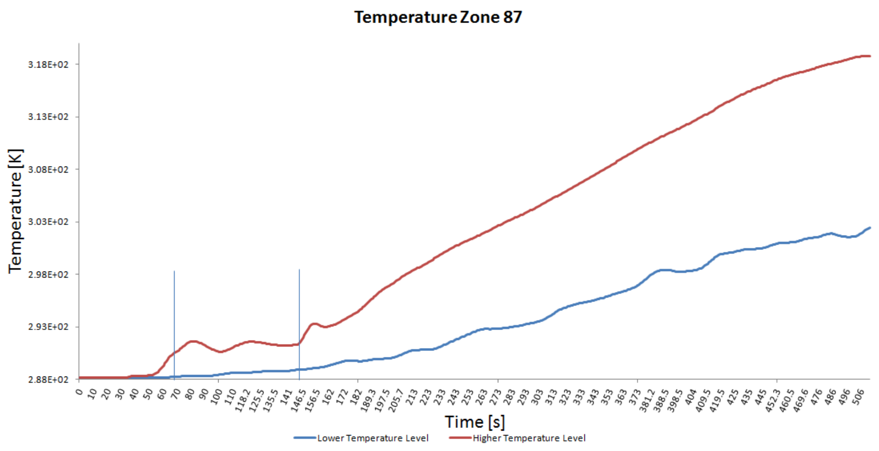


(b) Smoke outputs and door events for zone 86 in scenario 2.

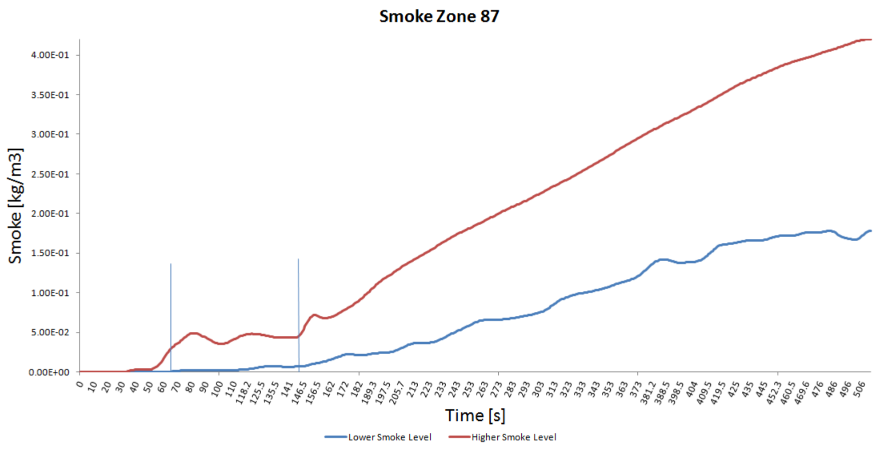


(c) Radiation outputs and door events for zone 86 in scenario 2.

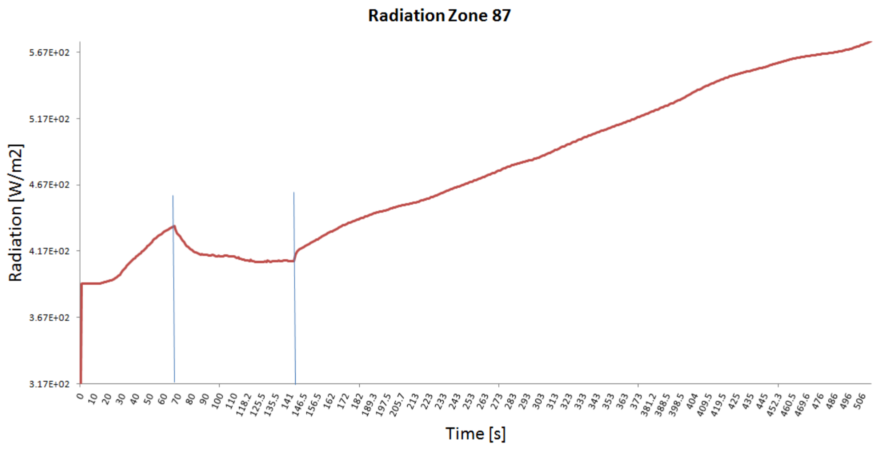
Figure A.25: Temperature, smoke and radiation vs. time for zone 86.



(a) Temperature outputs and door events for zone 87 in scenario 2.

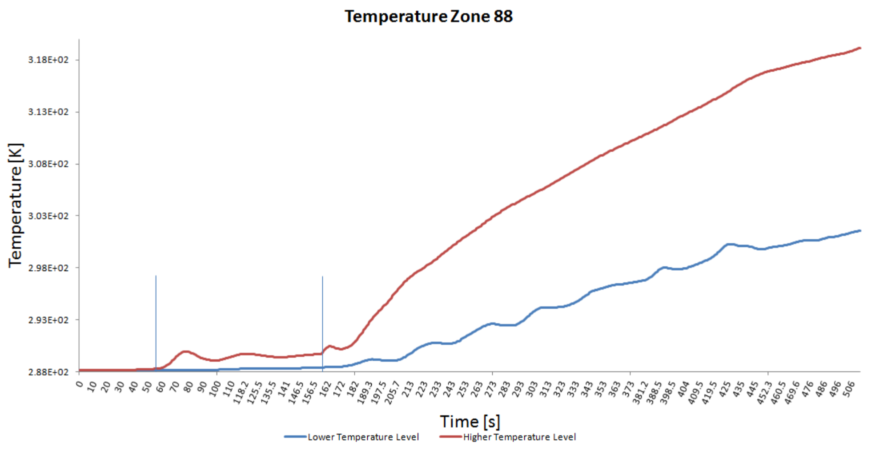


(b) Smoke outputs and door events for zone 87 in scenario 2.

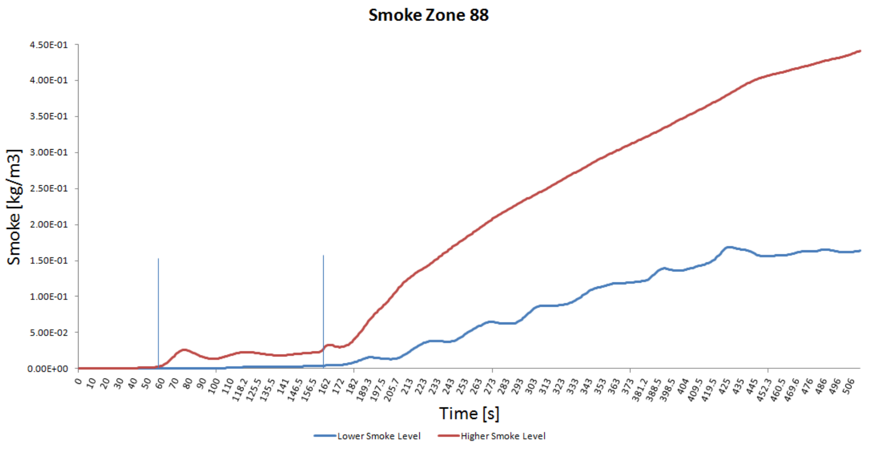


(c) Radiation outputs and door events for zone 87 in scenario 2.

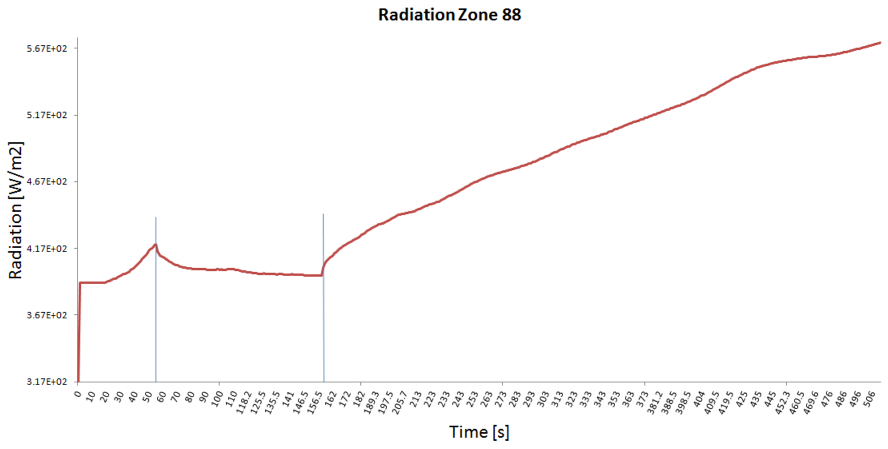
Figure A.26: Temperature, smoke and radiation vs. time for zone 87.



(a) Temperature outputs and door events for zone 88 in scenario 2.

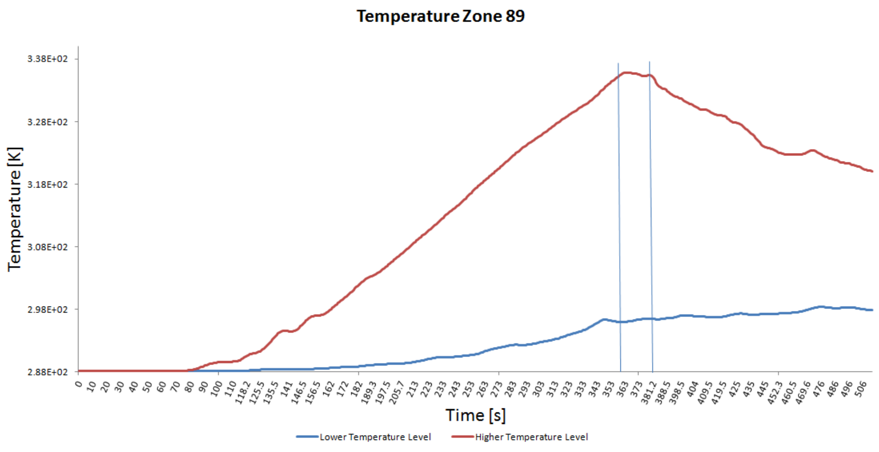


(b) Smoke outputs and door events for zone 88 in scenario 2.

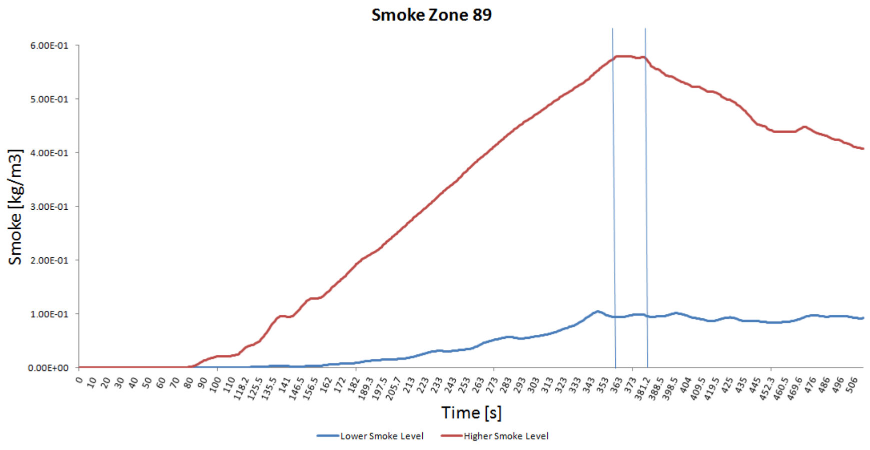


(c) Radiation outputs and door events for zone 88 in scenario 2.

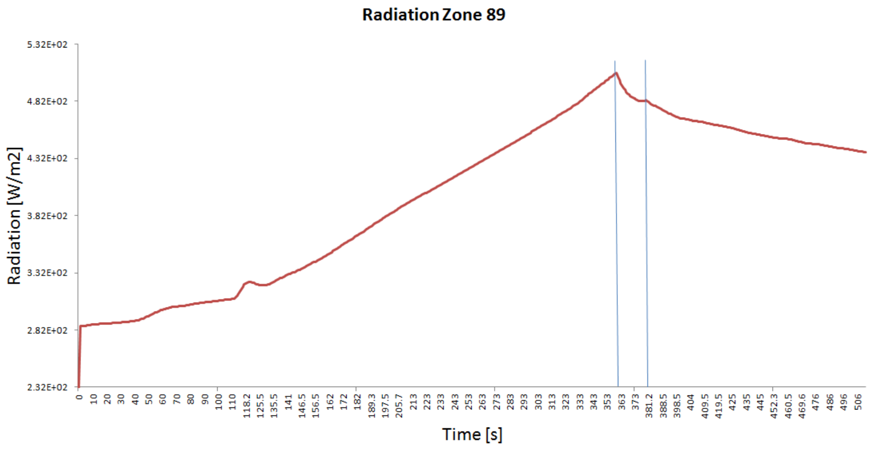
Figure A.27: Temperature, smoke and radiation vs. time for zone 88.



(a) Temperature outputs and door events for zone 89 in scenario 2.

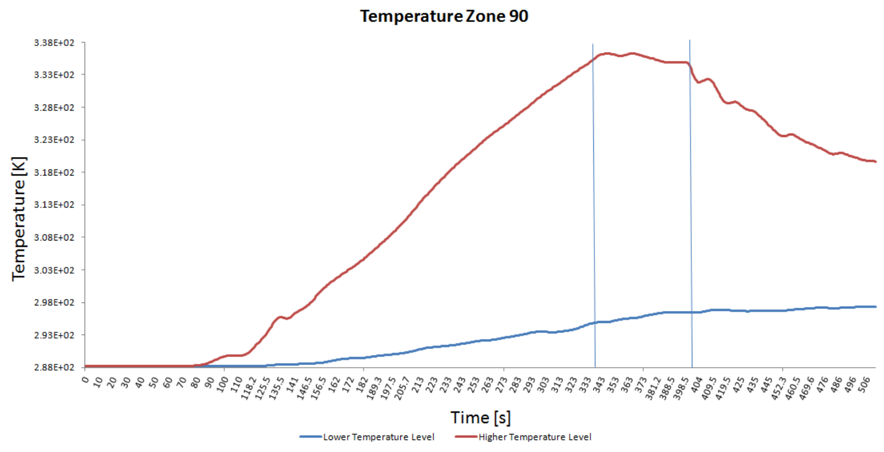


(b) Smoke outputs and door events for zone 89 in scenario 2.

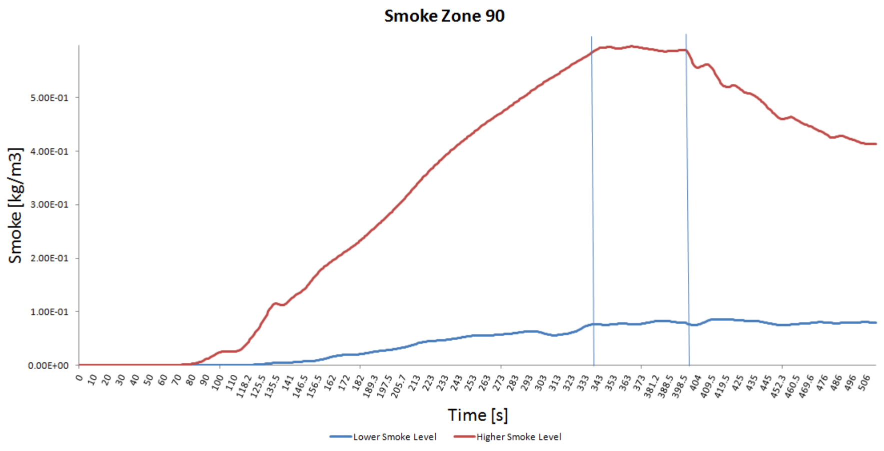


(c) Radiation outputs and door events for zone 89 in scenario 2.

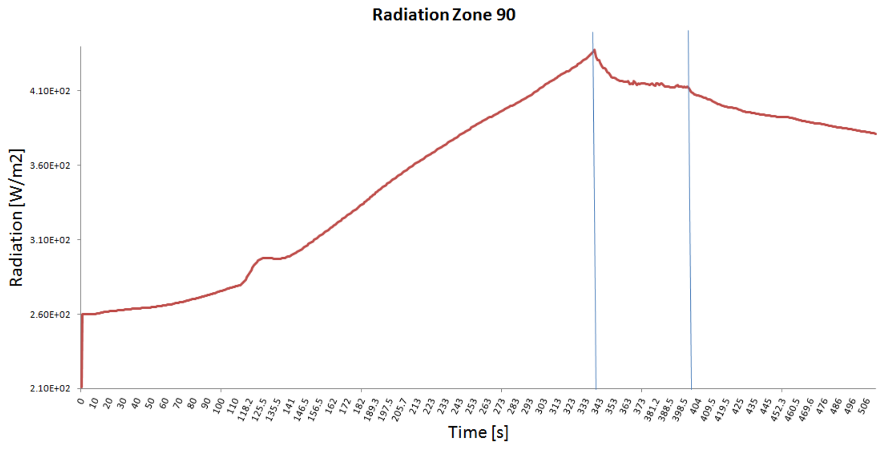
Figure A.28: Temperature, smoke and radiation vs. time for zone 89.



(a) Temperature outputs and door events for zone 90 in scenario 2.

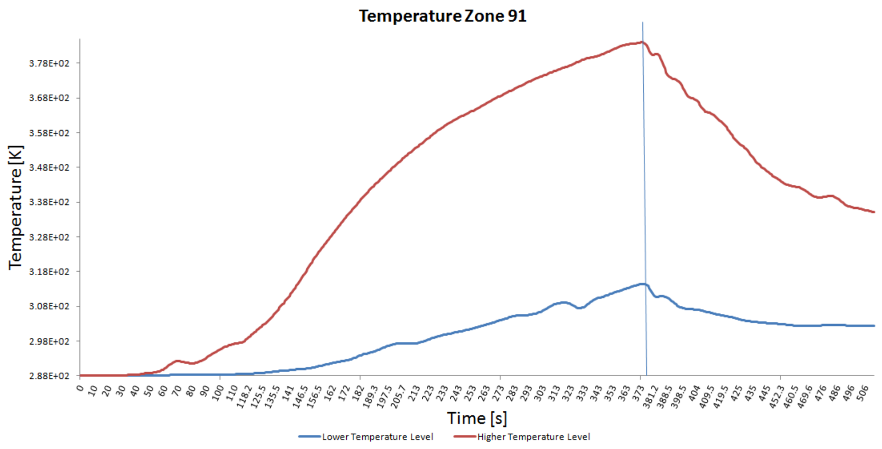


(b) Smoke outputs and door events for zone 90 in scenario 2.

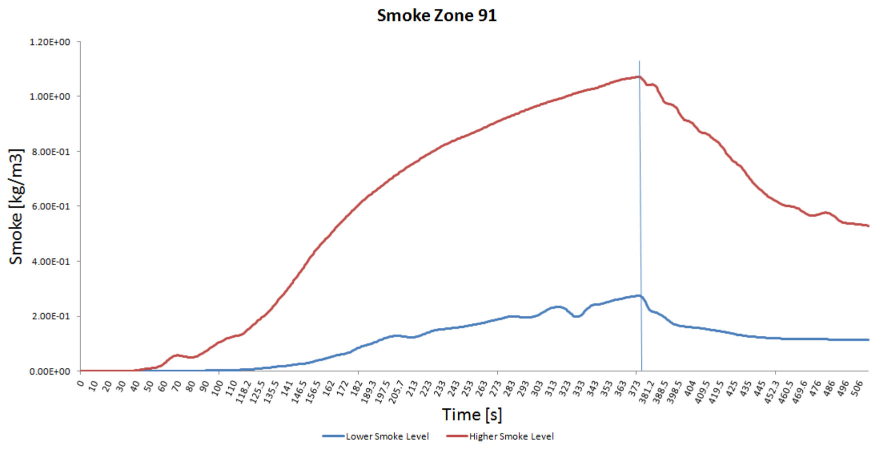


(c) Radiation outputs and door events for zone 90 in scenario 2.

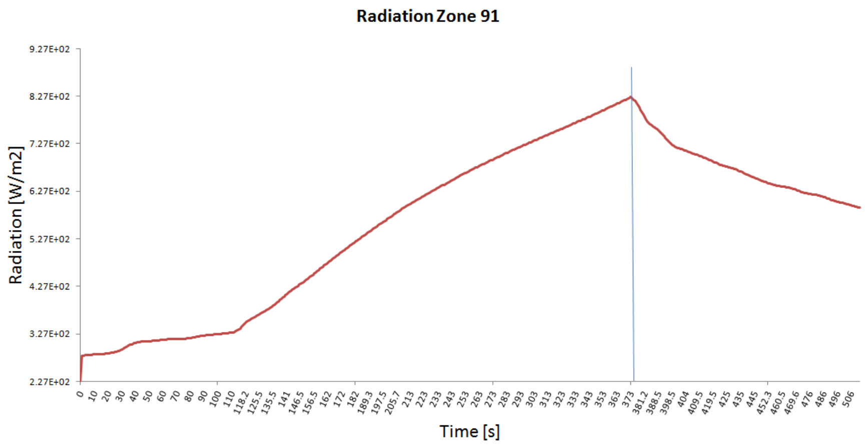
Figure A.29: Temperature, smoke and radiation vs. time for zone 90.



(a) Temperature outputs and door events for zone 91 in scenario 2.

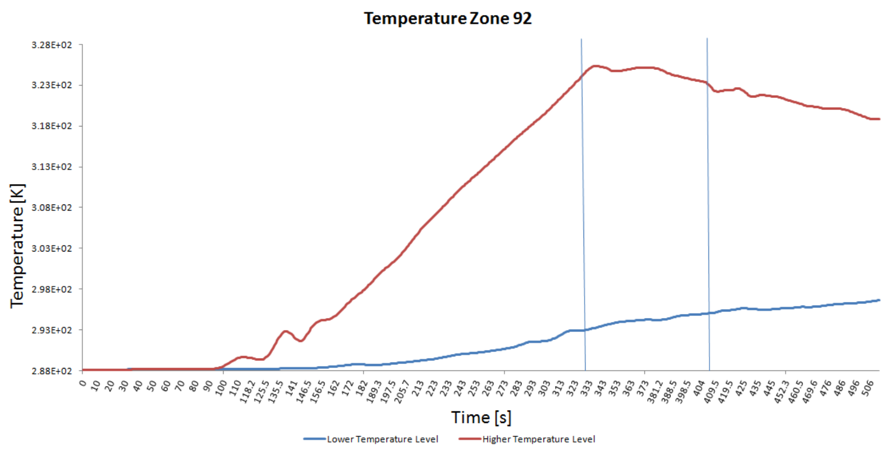


(b) Smoke outputs and door events for zone 91 in scenario 2.

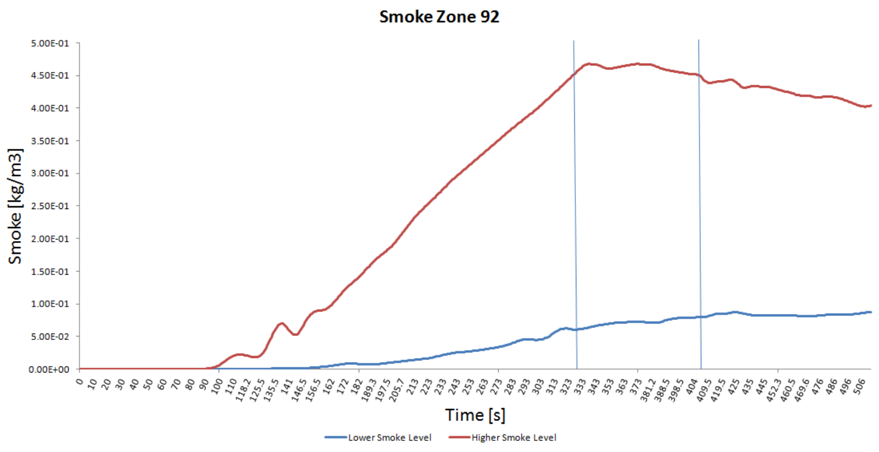


(c) Radiation outputs and door events for zone 91 in scenario 2.

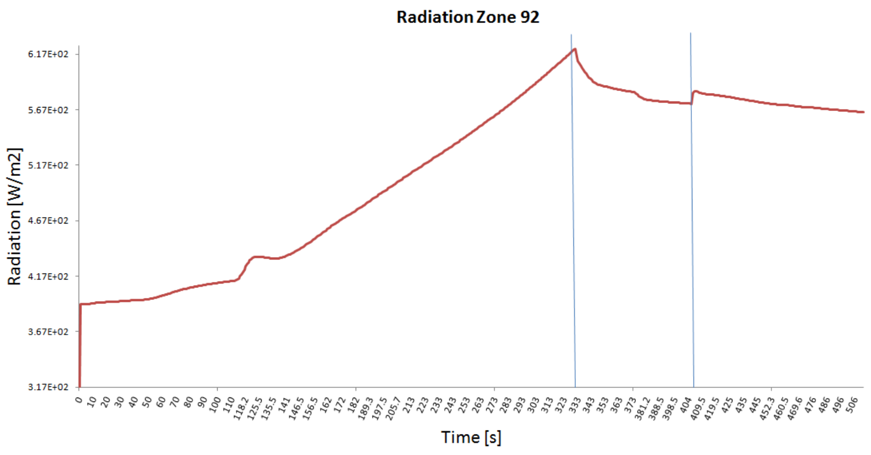
Figure A.30: Temperature, smoke and radiation vs. time for zone 91.



(a) Temperature outputs and door events for zone 92 in scenario 2.

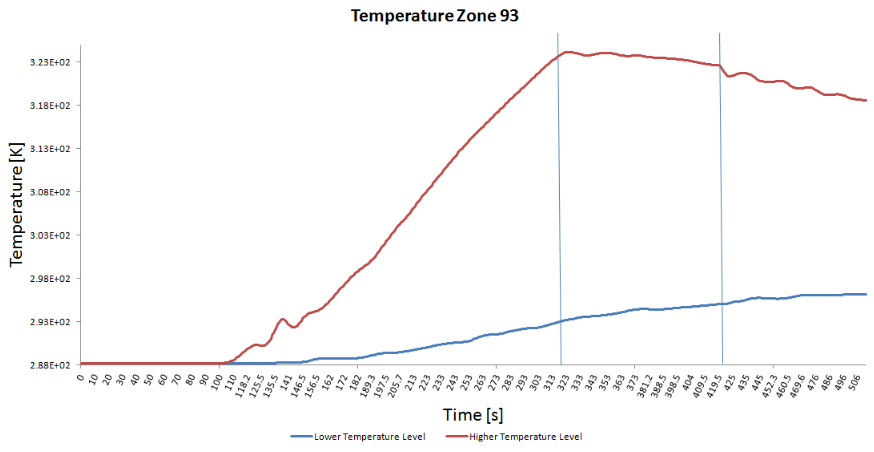


(b) Smoke outputs and door events for zone 92 in scenario 2.

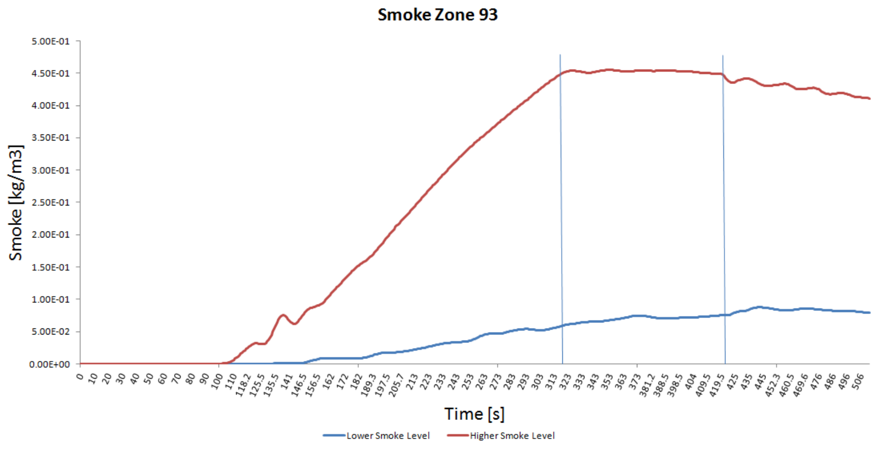


(c) Radiation outputs and door events for zone 92 in scenario 2.

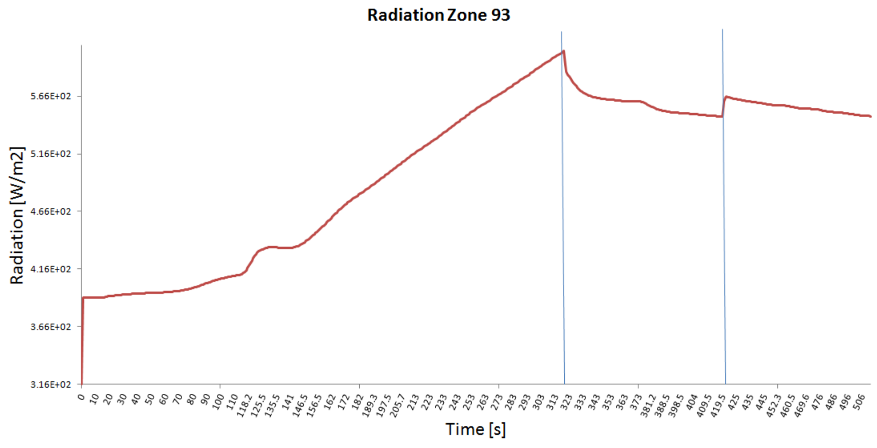
Figure A.31: Temperature, smoke and radiation vs. time for zone 92.



(a) Temperature outputs and door events for zone 93 in scenario 2.



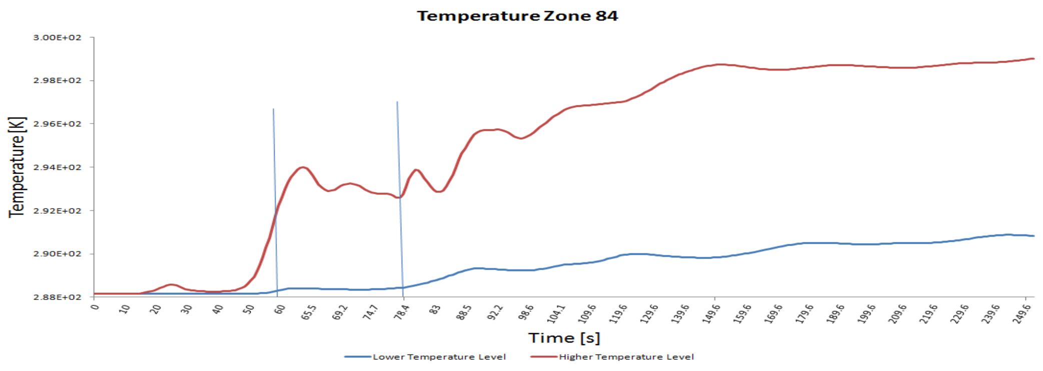
(b) Smoke outputs and door events for zone 93 in scenario 2.



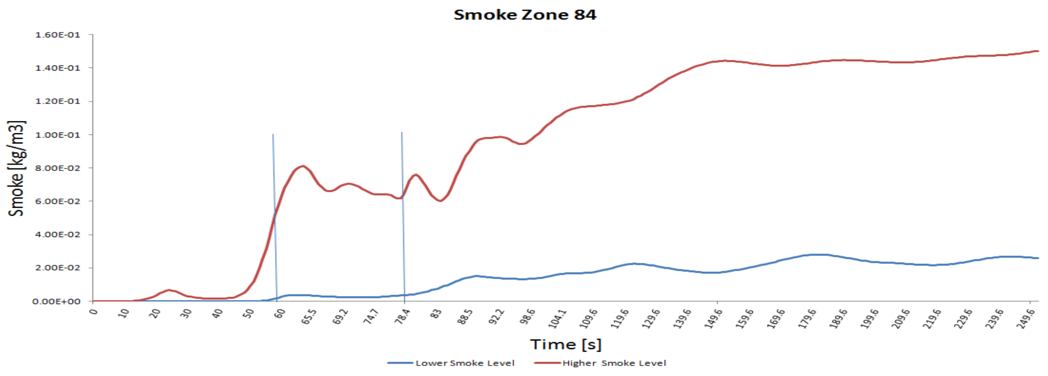
(c) Radiation outputs and door events for zone 93 in scenario 2.

Figure A.32: Temperature, smoke and radiation vs. time for zone 93.

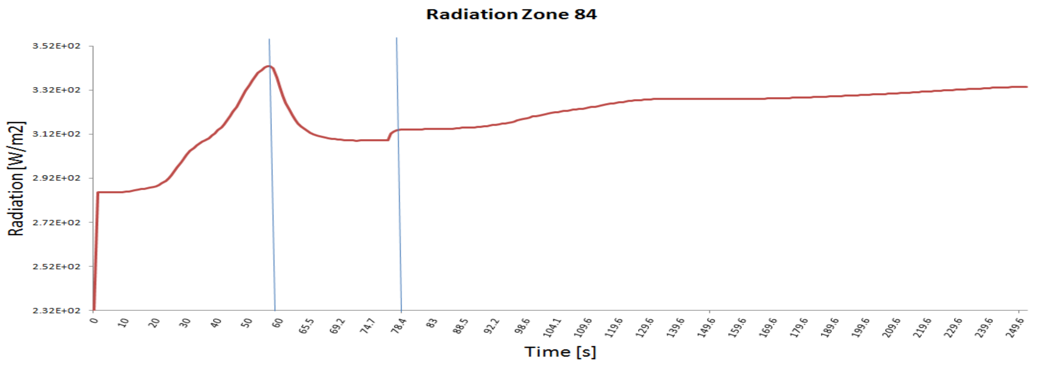
Scenario 3



(a) Temperature outputs and door events for zone 84 in scenario 3.

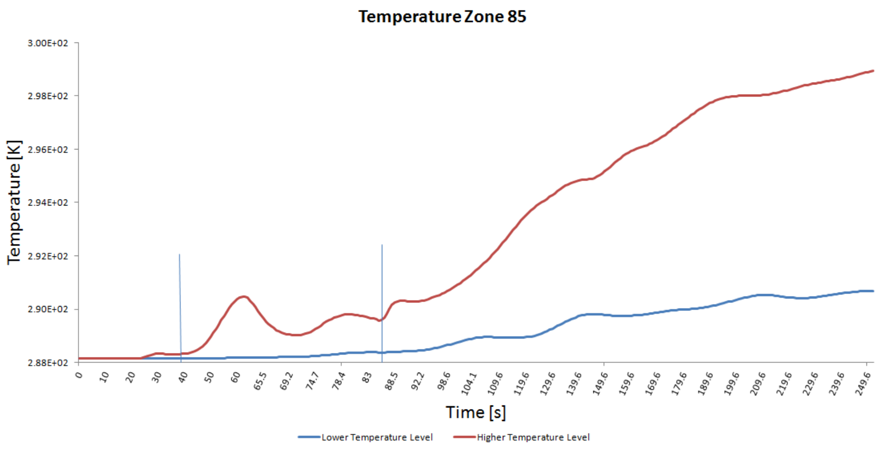


(b) Smoke outputs and door events for zone 84 in scenario 3.

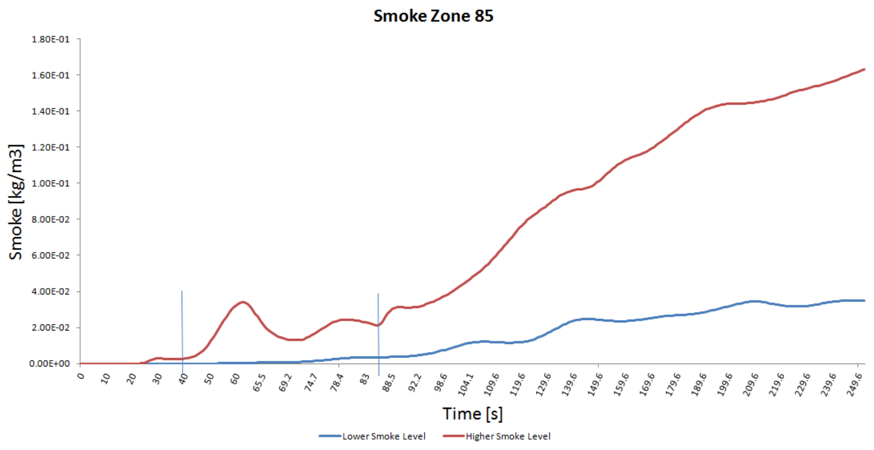


(c) Radiation outputs and door events for zone 84 in scenario 3.

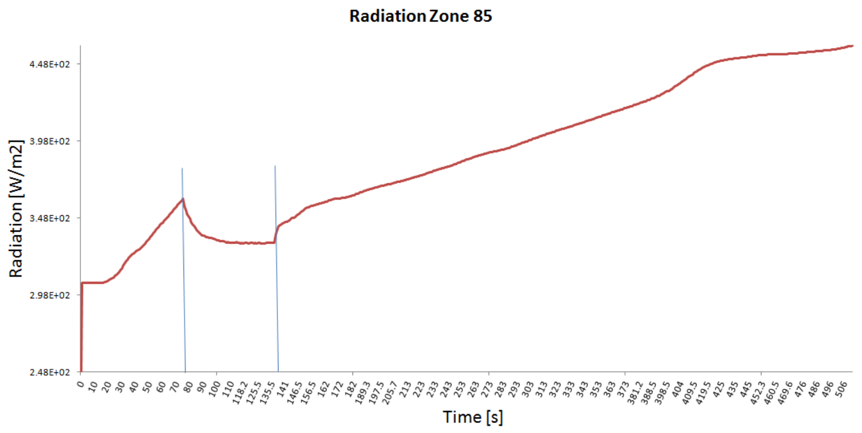
Figure A.33: Temperature, smoke and radiation vs. time for zone 84.



(a) Temperature outputs and door events for zone 85 in scenario 3.

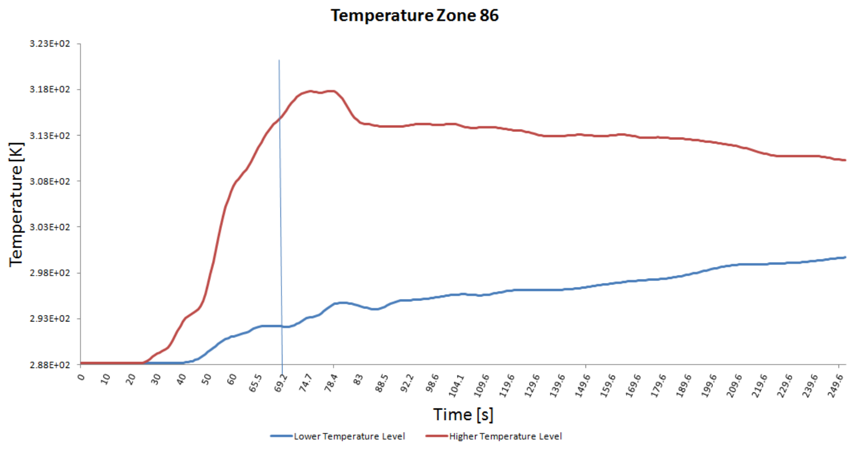


(b) Smoke outputs and door events for zone 85 in scenario 3.

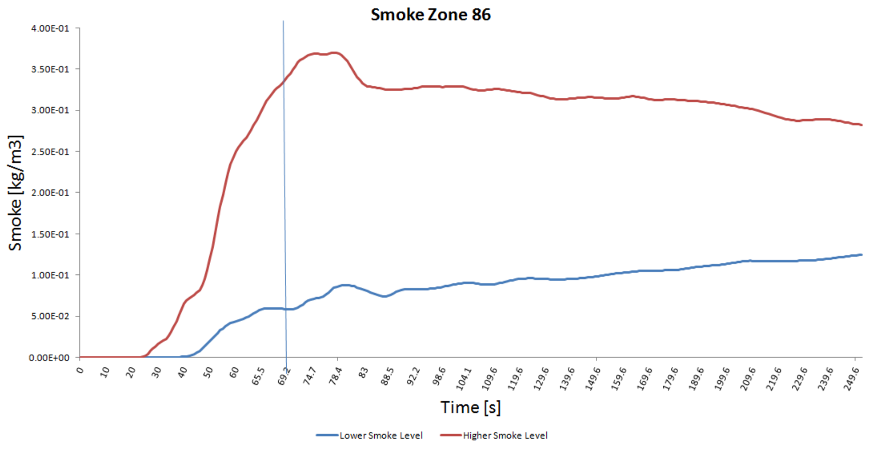


(c) Radiation outputs and door events for zone 85 in scenario 3.

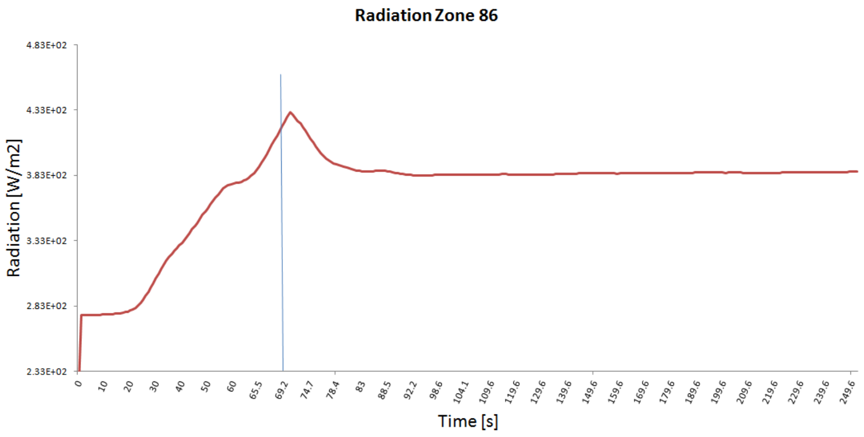
Figure A.34: Temperature, smoke and radiation vs. time for zone 85.



(a) Temperature outputs and door events for zone 86 in scenario 3.

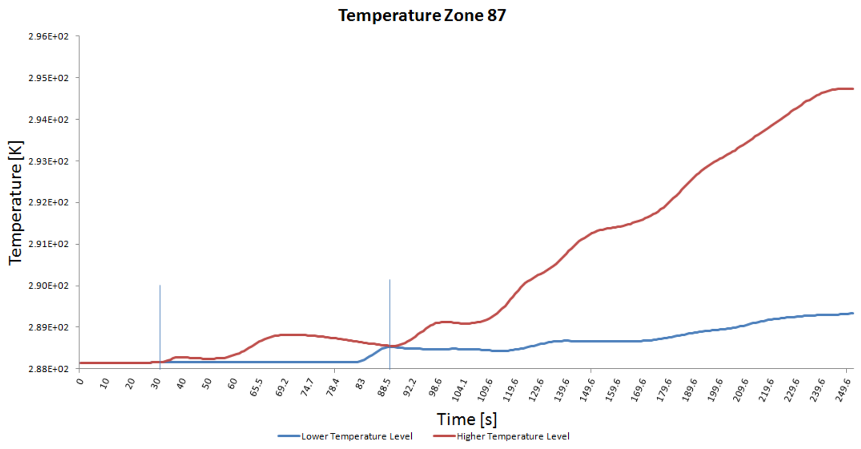


(b) Smoke outputs and door events for zone 86 in scenario 3.

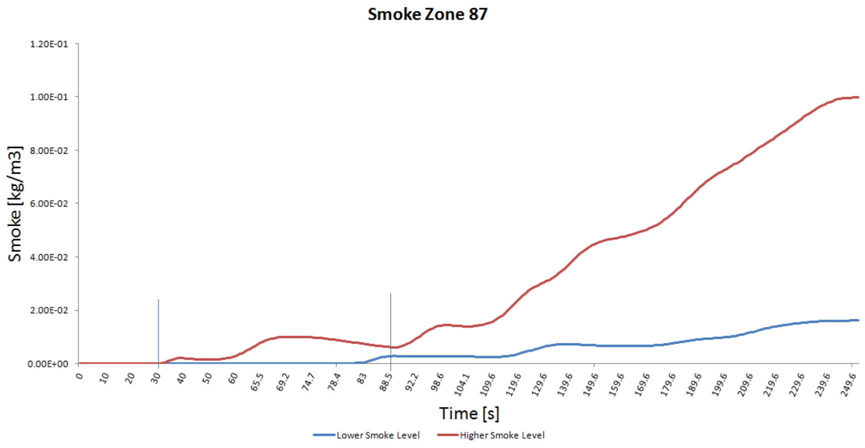


(c) Radiation outputs and door events for zone 86 in scenario 3.

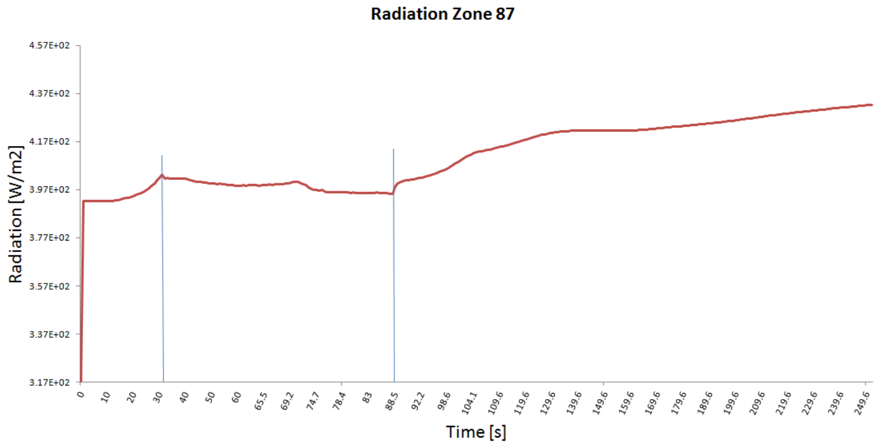
Figure A.35: Temperature, smoke and radiation vs. time for zone 86.



(a) Temperature outputs and door events for zone 87 in scenario 3.

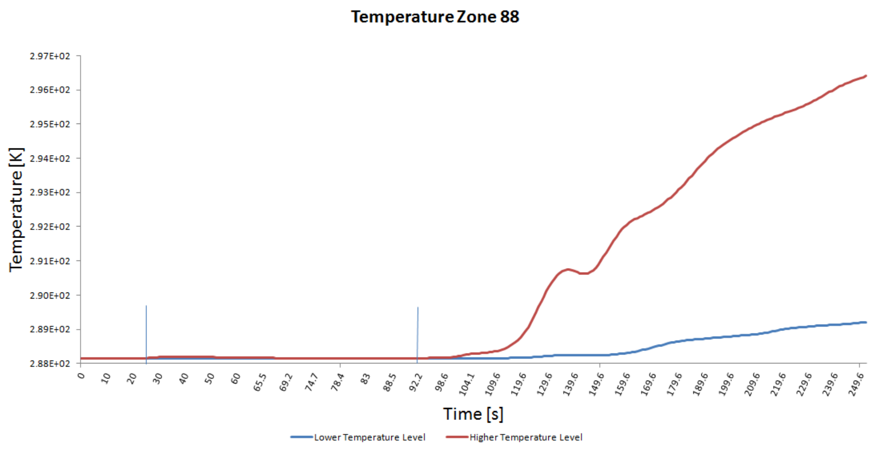


(b) Smoke outputs and door events for zone 87 in scenario 3.

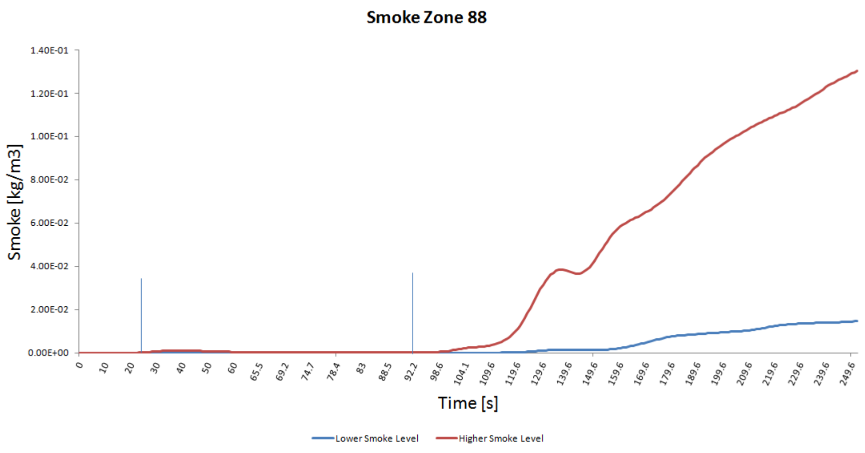


(c) Radiation outputs and door events for zone 87 in scenario 3.

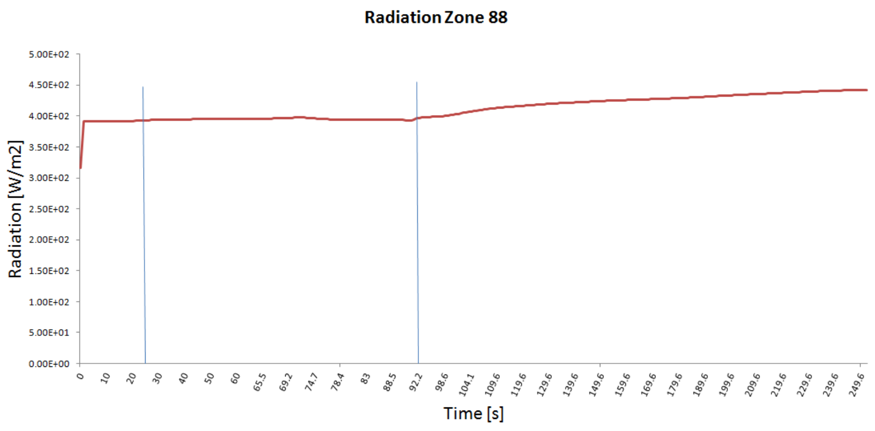
Figure A.36: Temperature, smoke and radiation vs. time for zone 87.



(a) Temperature outputs and door events for zone 88 in scenario 3.

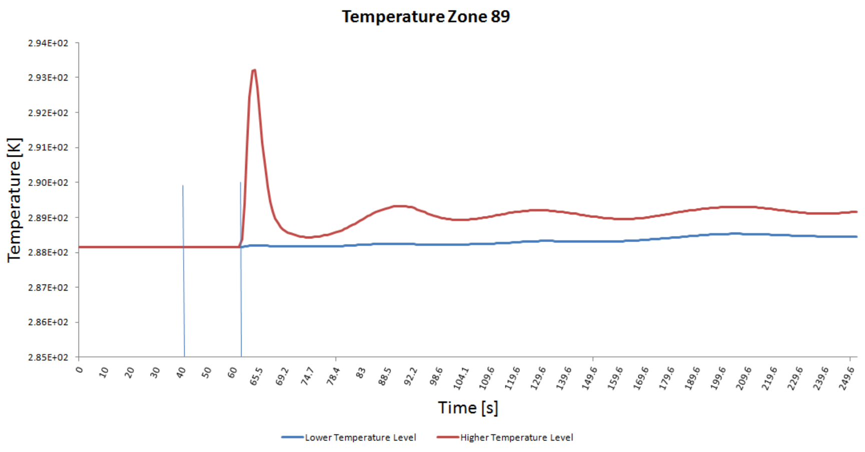


(b) Smoke outputs and door events for zone 88 in scenario 3.

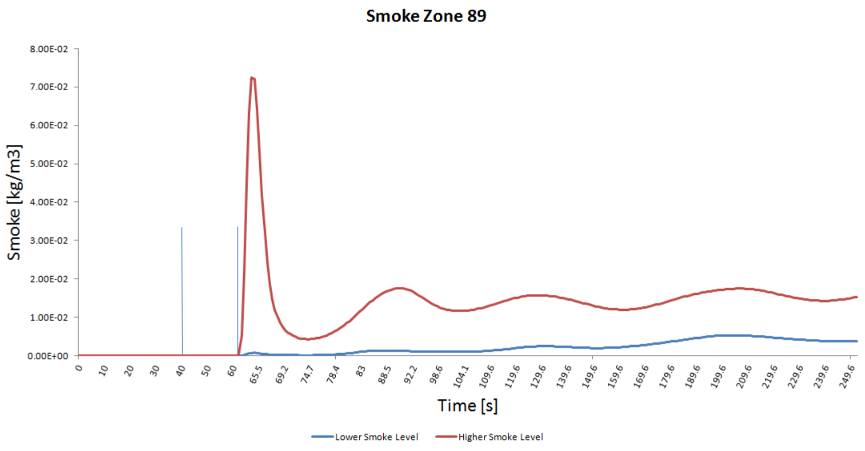


(c) Radiation outputs and door events for zone 88 in scenario 3.

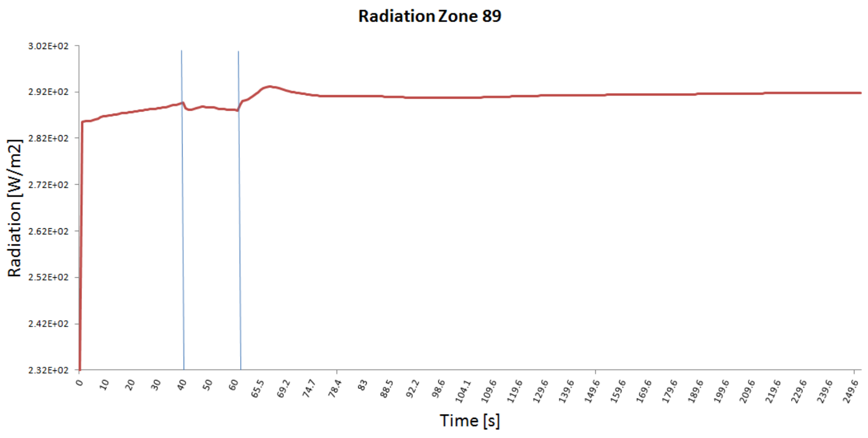
Figure A.37: Temperature, smoke and radiation vs. time for zone 88.



(a) Temperature outputs and door events for zone 89 in scenario 3.

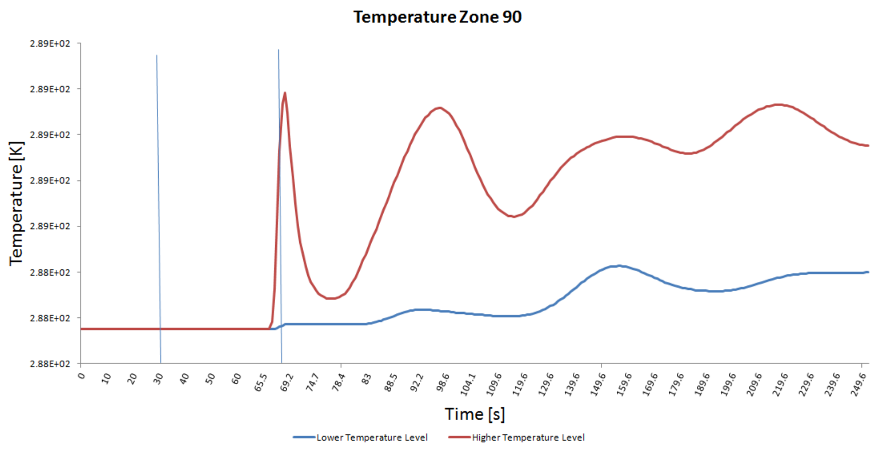


(b) Smoke outputs and door events for zone 89 in scenario 3.

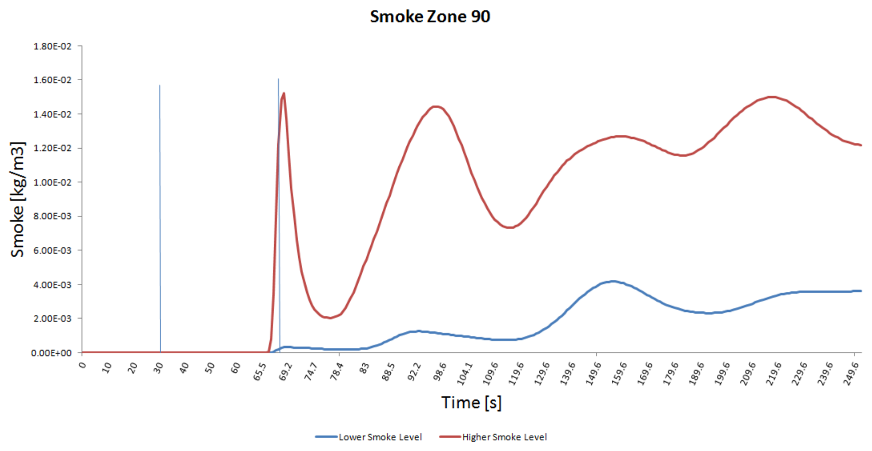


(c) Radiation outputs and door events for zone 89 in scenario 3.

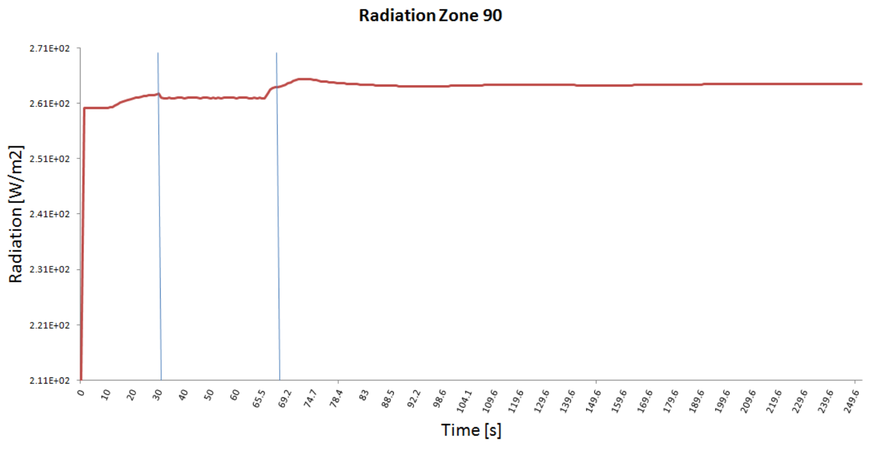
Figure A.38: Temperature, smoke and radiation vs. time for zone 89.



(a) Temperature outputs and door events for zone 90 in scenario 3.

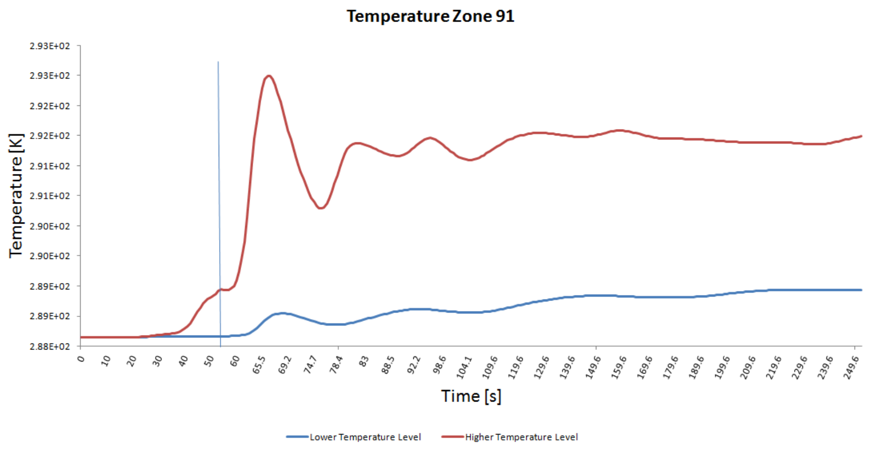


(b) Smoke outputs and door events for zone 90 in scenario 3.

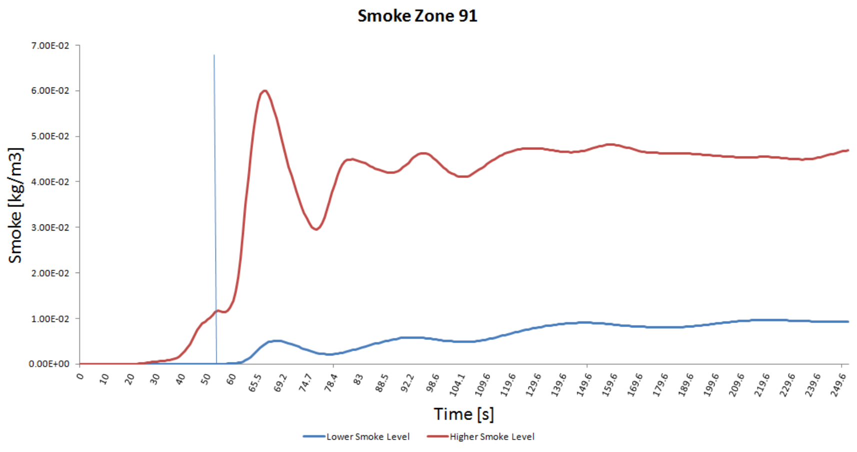


(c) Radiation outputs and door events for zone 90 in scenario 3.

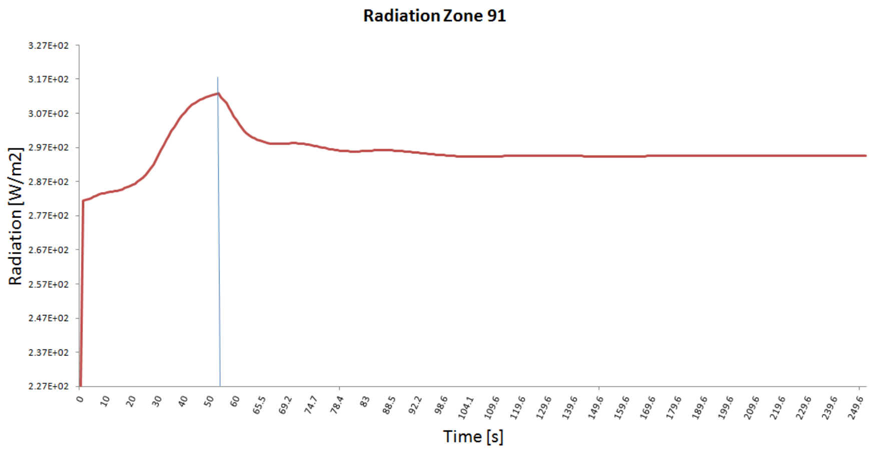
Figure A.39: Temperature, smoke and radiation vs. time for zone 90.



(a) Temperature outputs and door events for zone 91 in scenario 3.

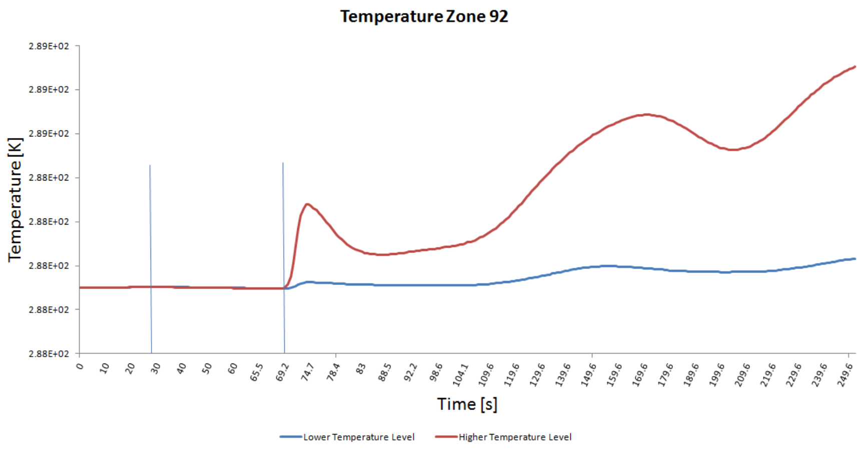


(b) Smoke outputs and door events for zone 91 in scenario 3.

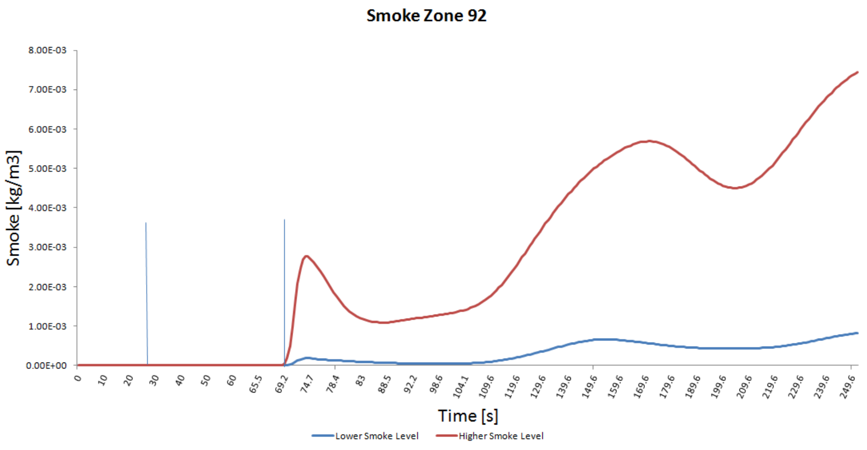


(c) Radiation outputs and door events for zone 91 in scenario 3.

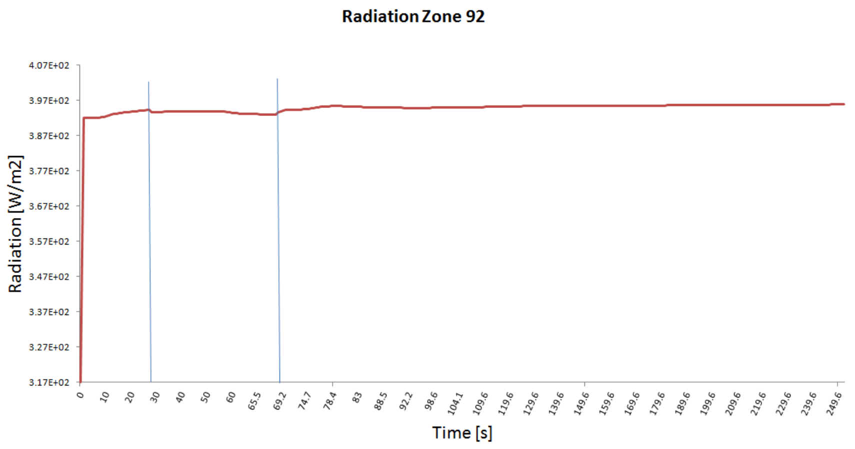
Figure A.40: Temperature, smoke and radiation vs. time for zone 91.



(a) Temperature outputs and door events for zone 92 in scenario 3.

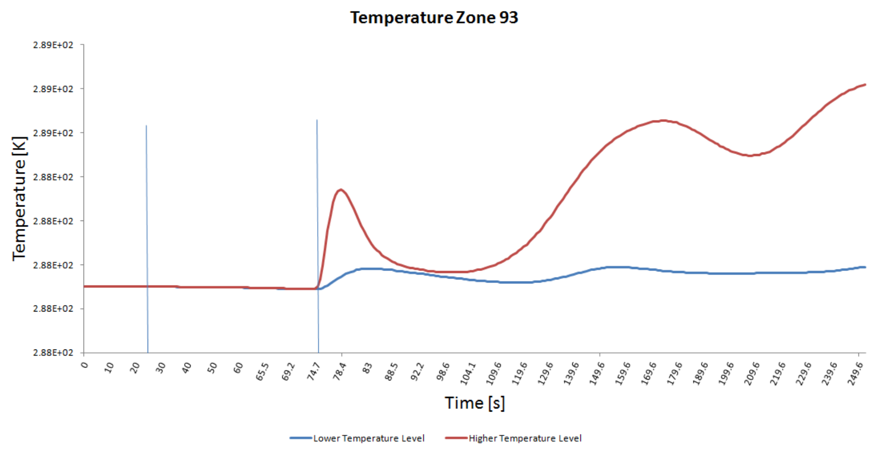


(b) Smoke outputs and door events for zone 92 in scenario 3.

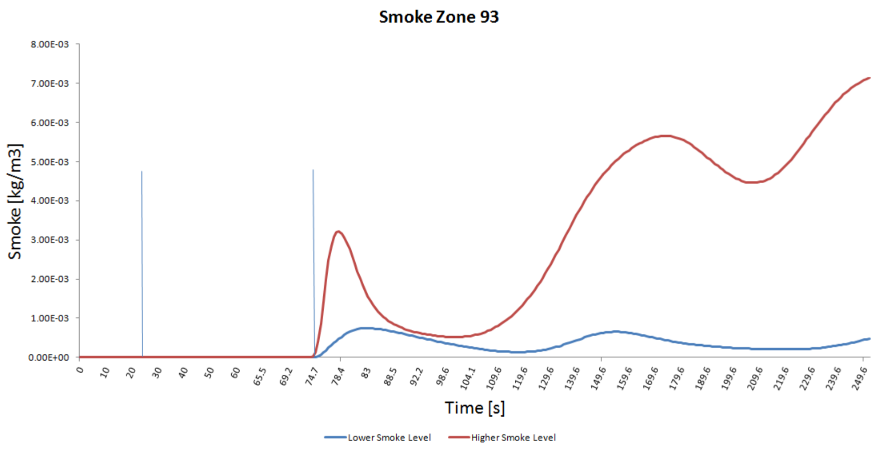


(c) Radiation outputs and door events for zone 92 in scenario 3.

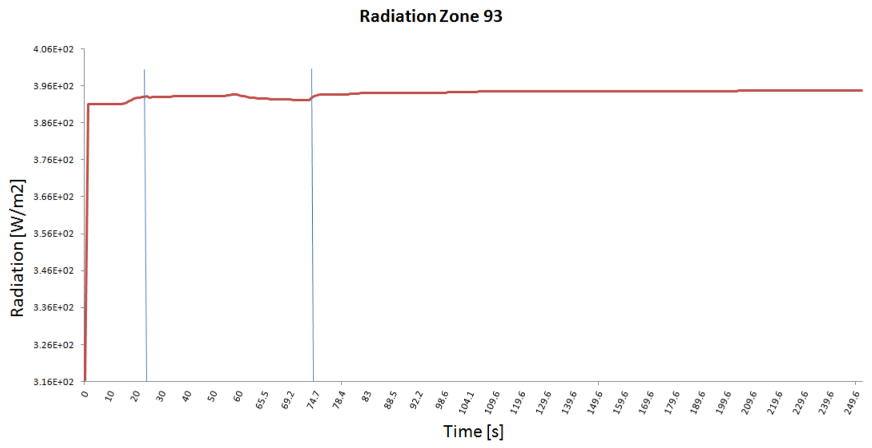
Figure A.41: Temperature, smoke and radiation vs. time for zone 92.



(a) Temperature outputs and door events for zone 93 in scenario 3.

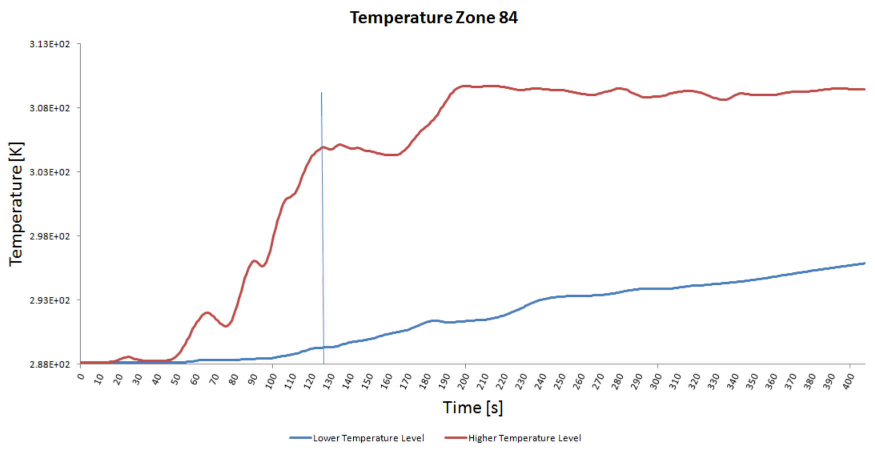


(b) Smoke outputs and door events for zone 93 in scenario 3.

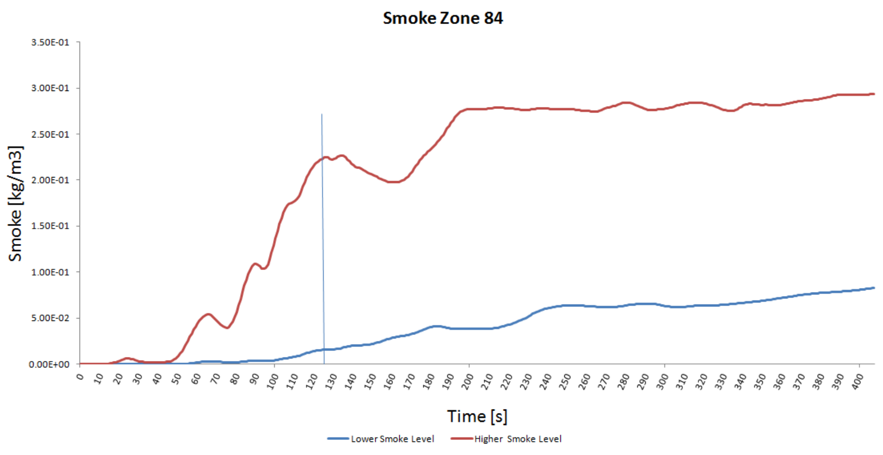


(c) Radiation outputs and door events for zone 93 in scenario 3.

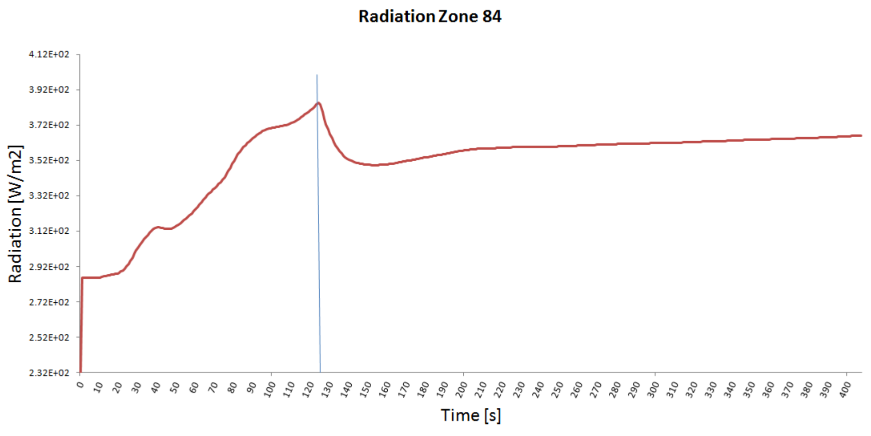
Figure A.42: Temperature, smoke and radiation vs. time for zone 93.



(a) Temperature outputs for zone 84 non-coupled.

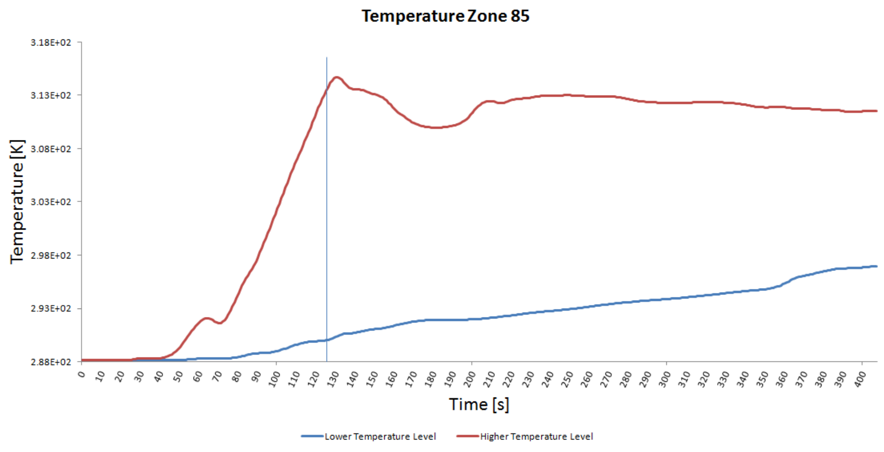


(b) Smoke outputs for zone 84 non-coupled.

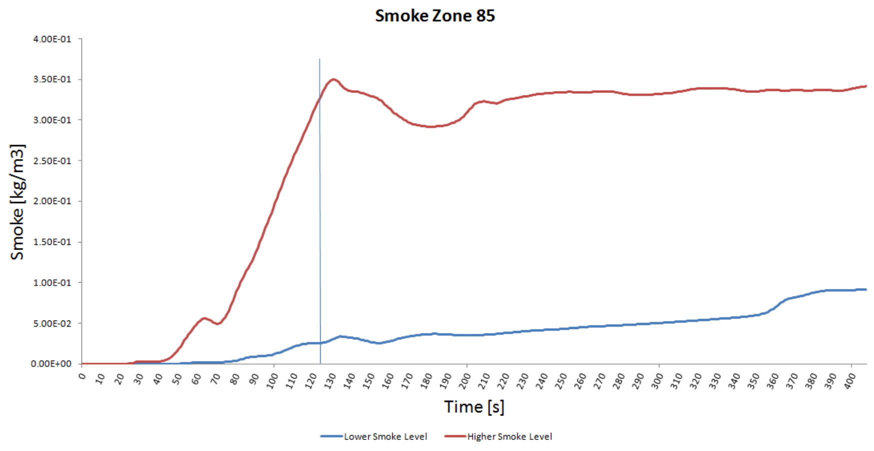


(c) Radiation outputs for zone 84 non-coupled.

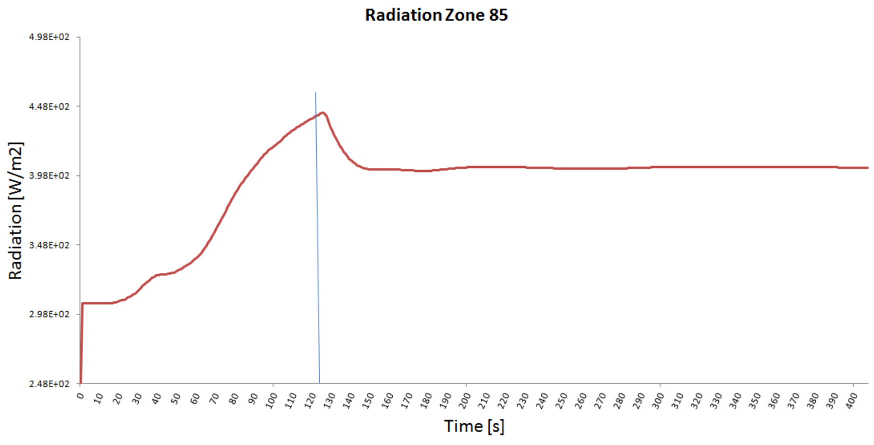
Figure A.43: Temperature, smoke and radiation vs. time for zone 84.



(a) Temperature outputs for zone 85 non-coupled.

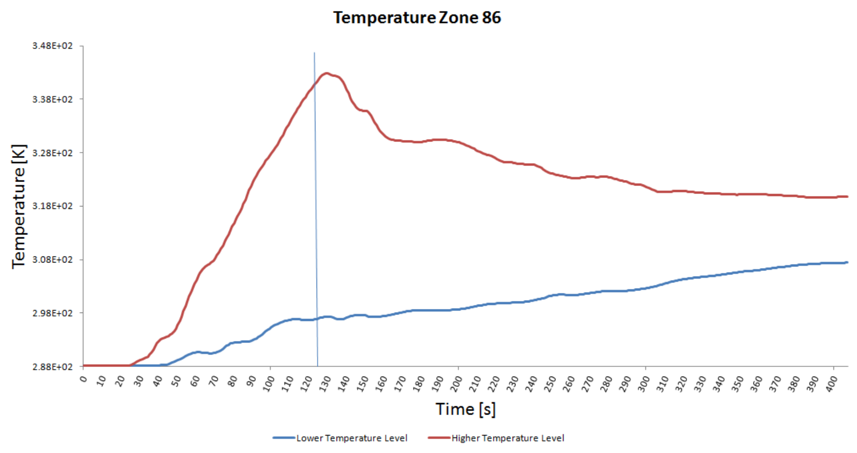


(b) Smoke outputs for zone 85 non-coupled.

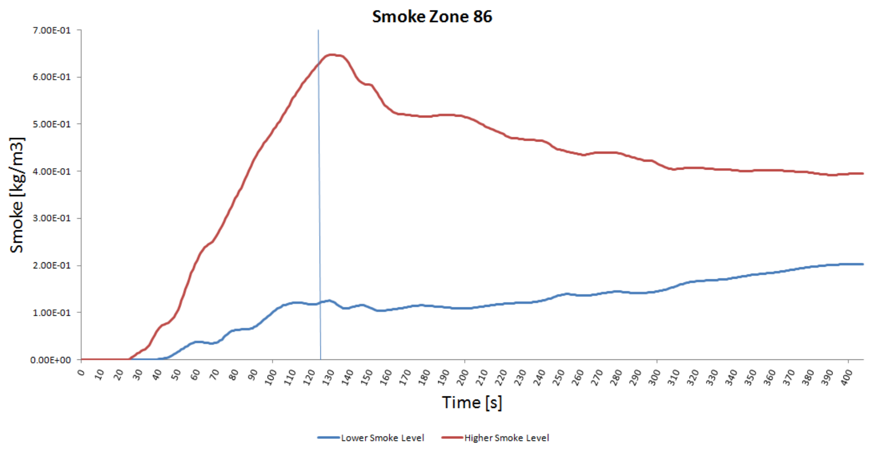


(c) Radiation outputs for zone 85 non-coupled.

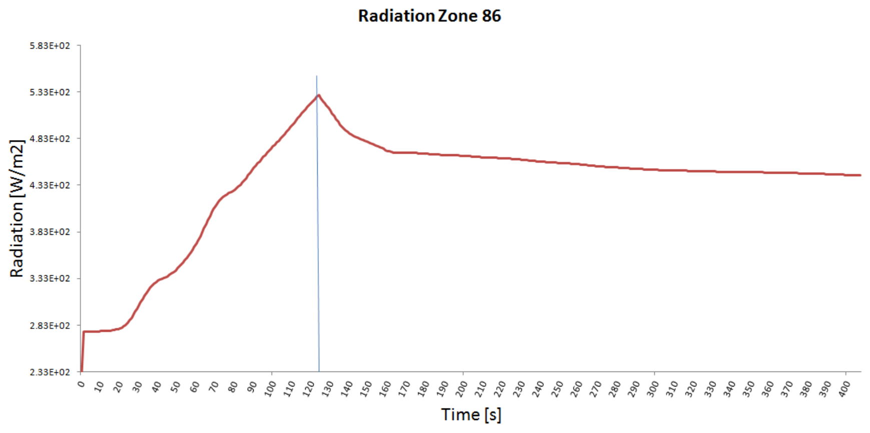
Figure A.44: Temperature, smoke and radiation vs. time for zone 85.



(a) Temperature outputs for zone 86 non-coupled.

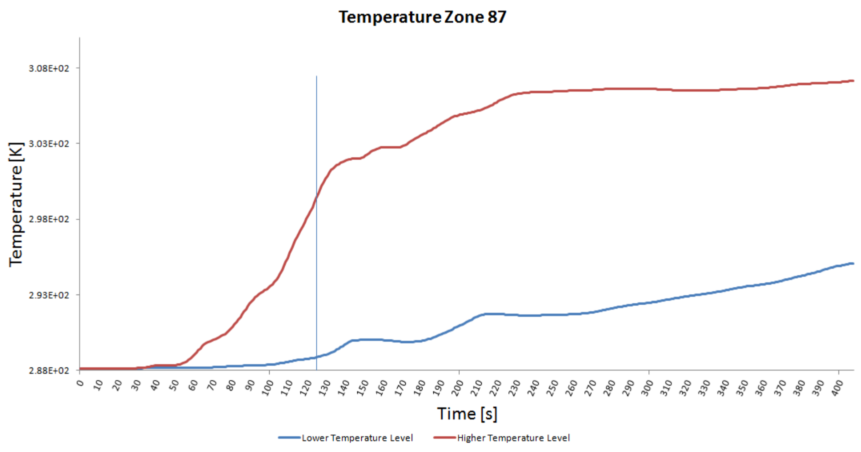


(b) Smoke outputs for zone 86 non-coupled.

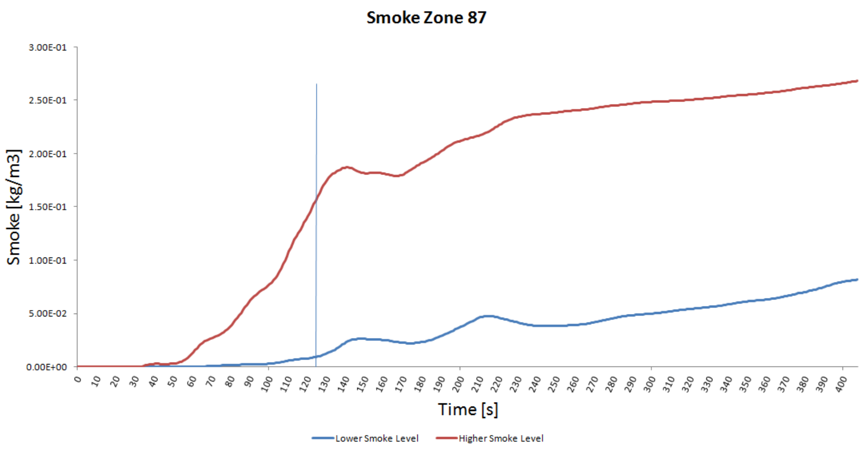


(c) Radiation outputs for zone 86 non-coupled.

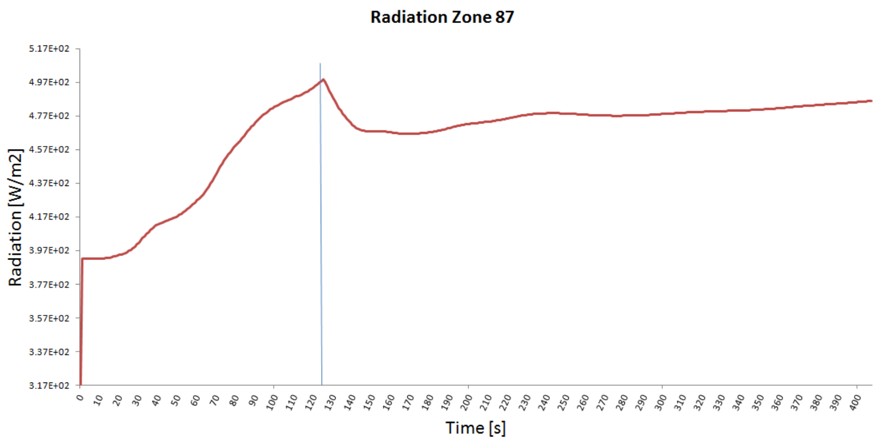
Figure A.45: Temperature, smoke and radiation vs. time for zone 86.



(a) Temperature outputs for zone 87 non-coupled.

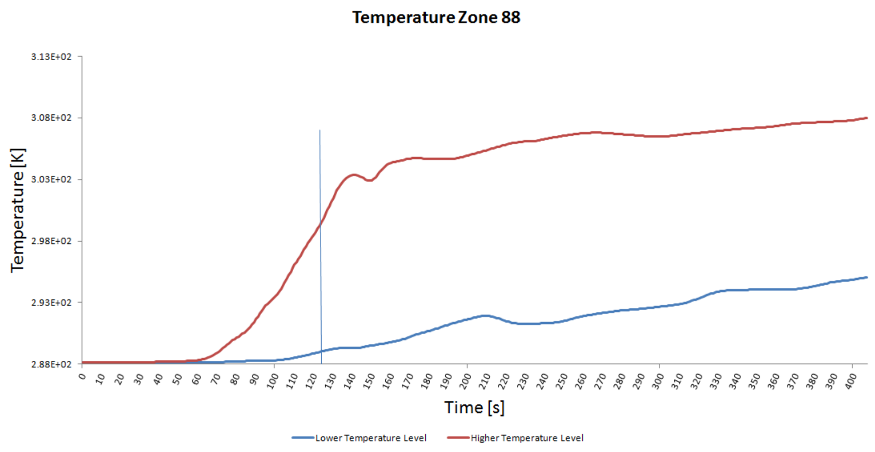


(b) Smoke outputs for zone 87 non-coupled.

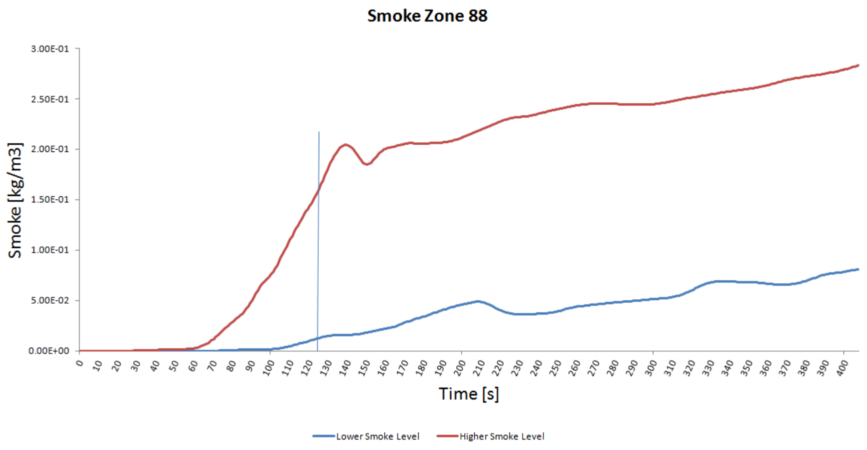


(c) Radiation outputs for zone 87 non-coupled.

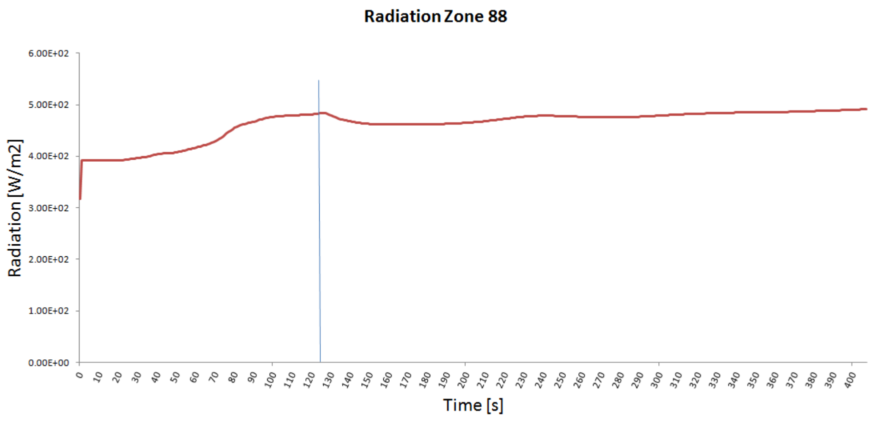
Figure A.46: Temperature, smoke and radiation vs. time for zone 87.



(a) Temperature outputs for zone 88 non-coupled.

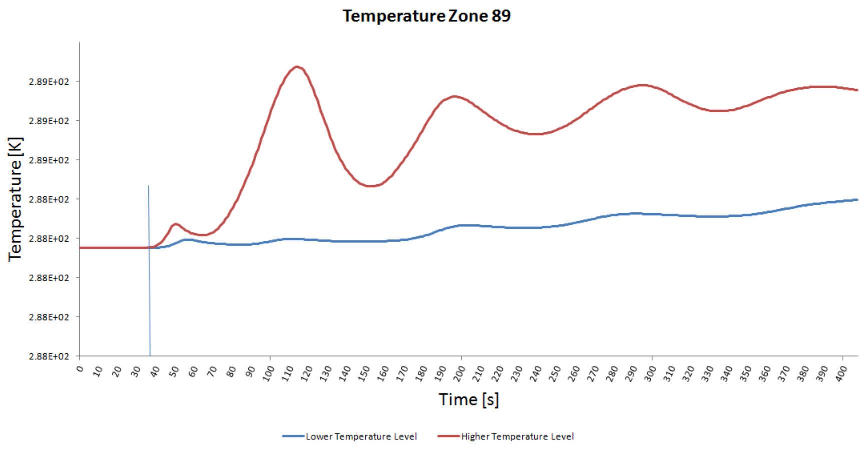


(b) Smoke outputs for zone 88 non-coupled.

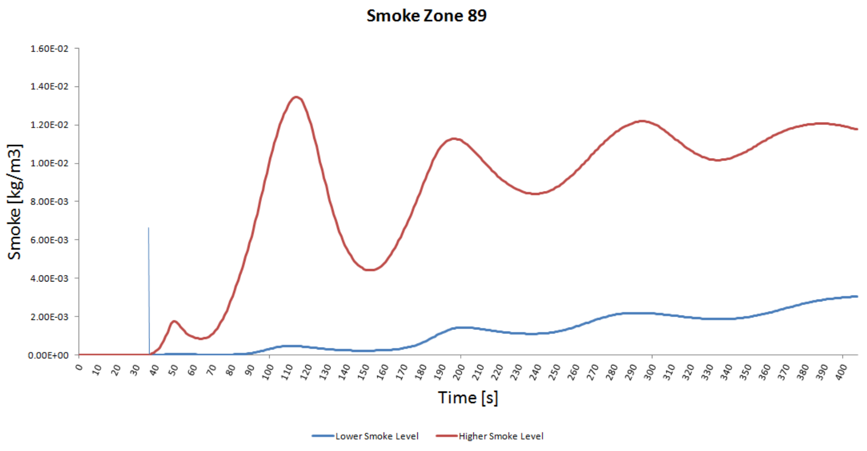


(c) Radiation outputs for zone 88 non-coupled.

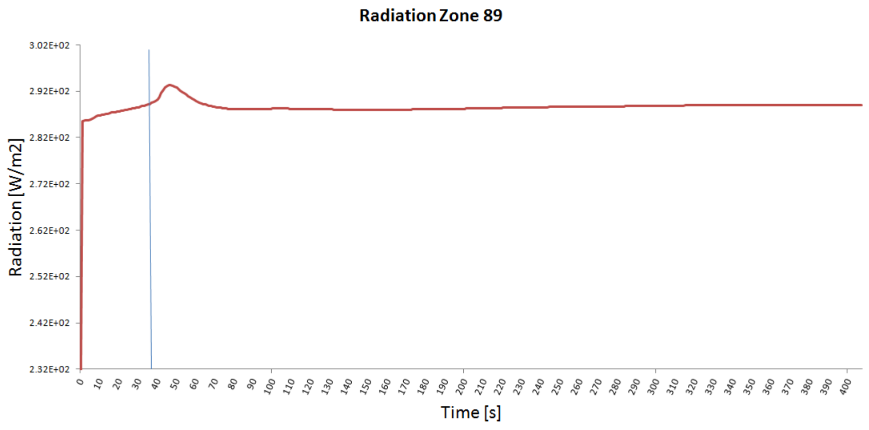
Figure A.47: Temperature, smoke and radiation vs. time for zone 88.



(a) Temperature outputs for zone 89 non-coupled.

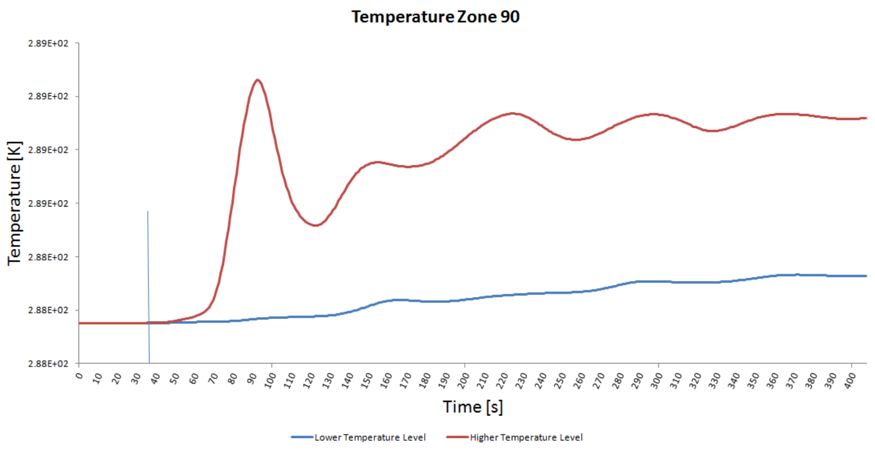


(b) Smoke outputs for zone 89 non-coupled.

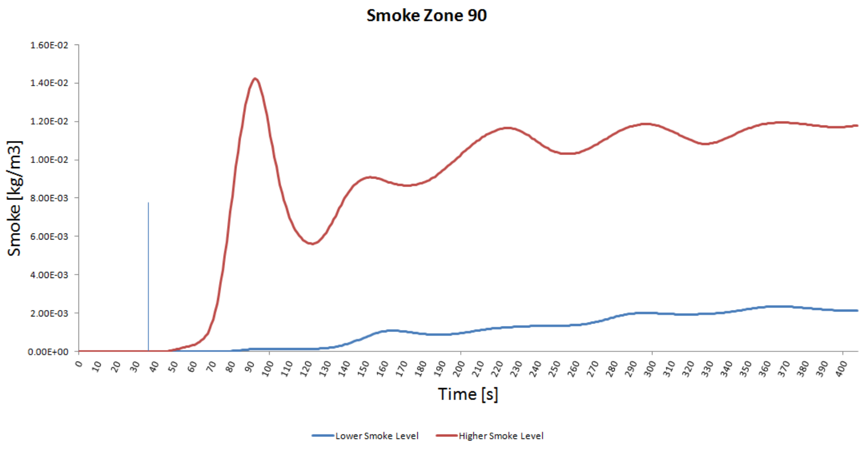


(c) Radiation outputs for zone 89 non-coupled.

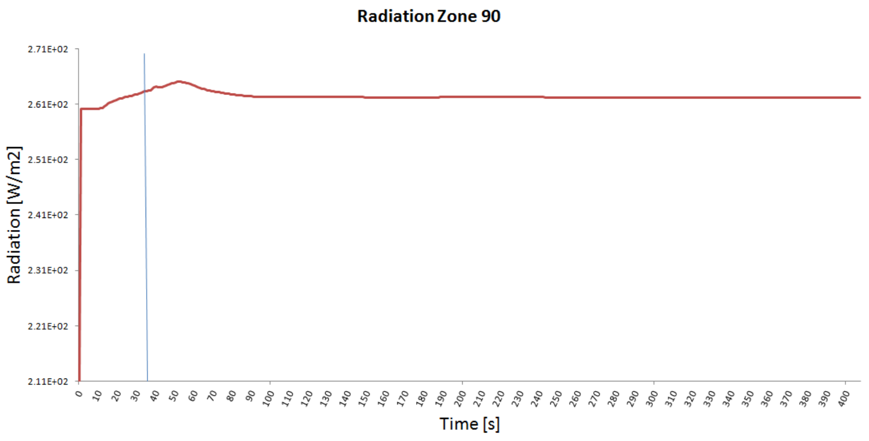
Figure A.48: Temperature, smoke and radiation vs. time for zone 89.



(a) Temperature outputs for zone 90 non-coupled.

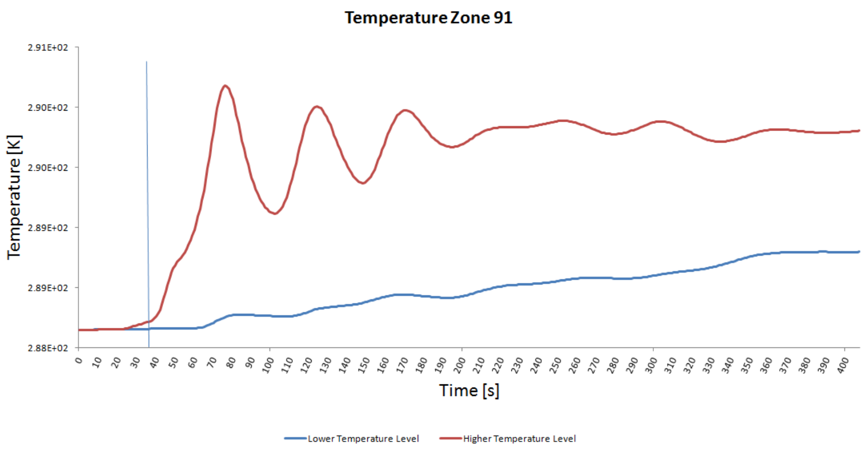


(b) Smoke outputs for zone 90 non-coupled.

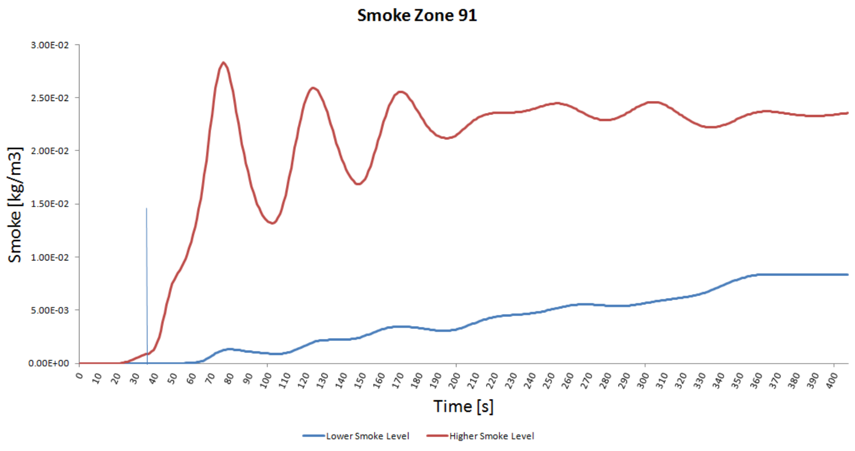


(c) Radiation outputs for zone 90 non-coupled.

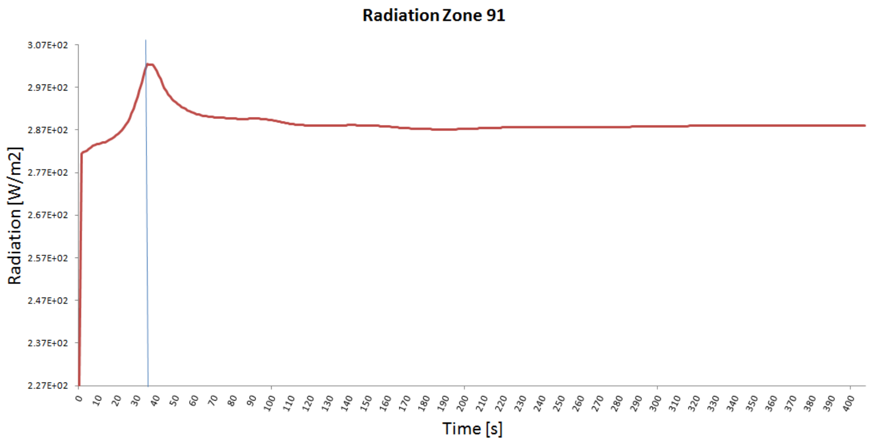
Figure A.49: Temperature, smoke and radiation vs. time for zone 90.



(a) Temperature outputs for zone 91 non-coupled.

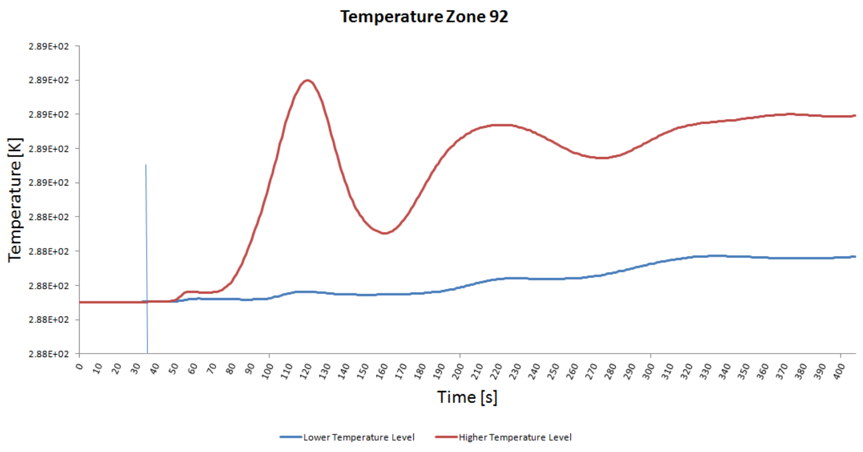


(b) Smoke outputs for zone 91 non-coupled.

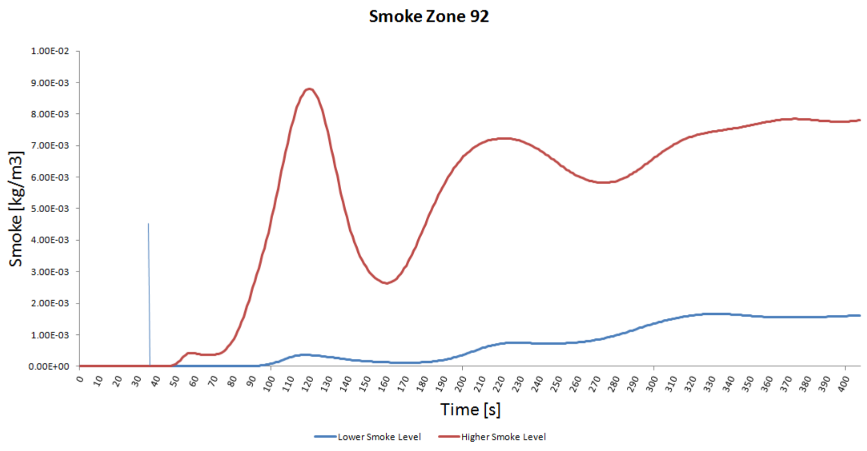


(c) Radiation outputs for zone 91 non-coupled.

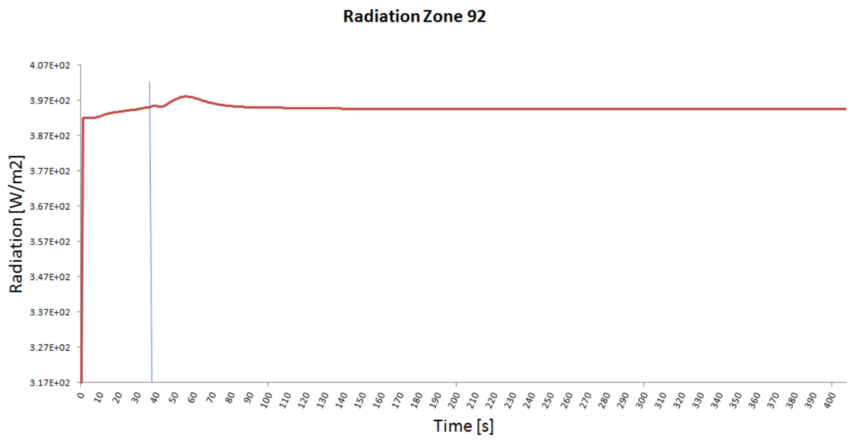
Figure A.50: Temperature, smoke and radiation vs. time for zone 91.



(a) Temperature outputs for zone 92 non-coupled.

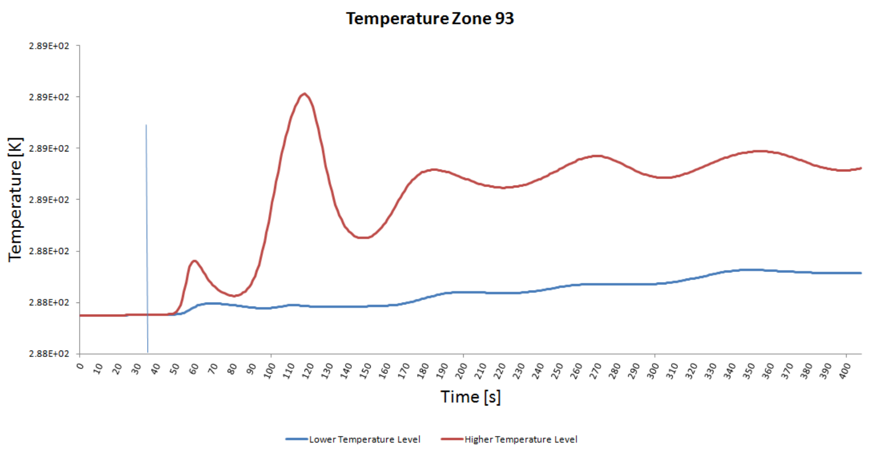


(b) Smoke outputs for zone 92 non-coupled.

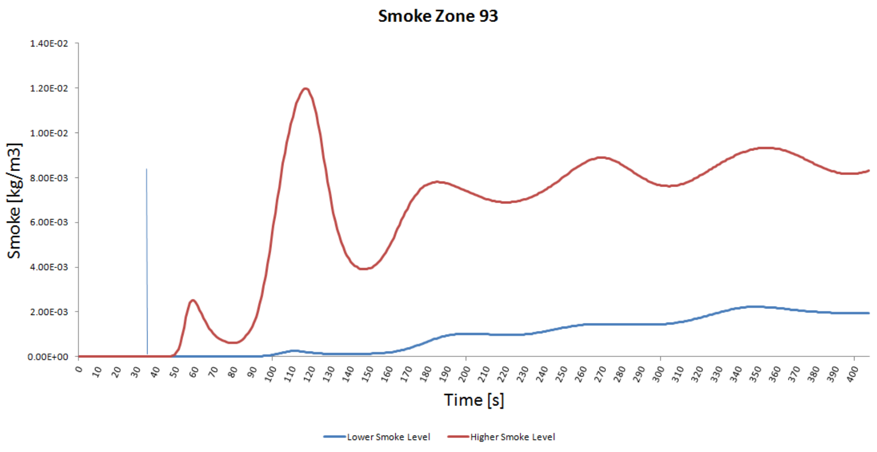


(c) Radiation outputs for zone 92 non-coupled.

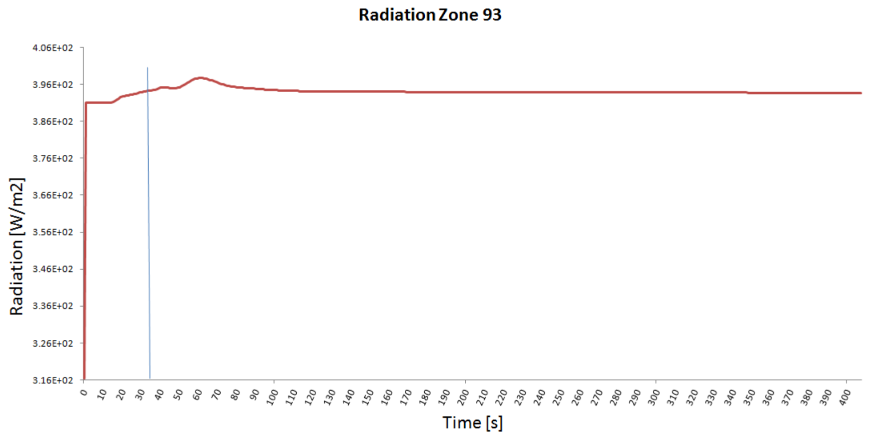
Figure A.51: Temperature, smoke and radiation vs. time for zone 92.



(a) Temperature outputs for zone 93 non-coupled.

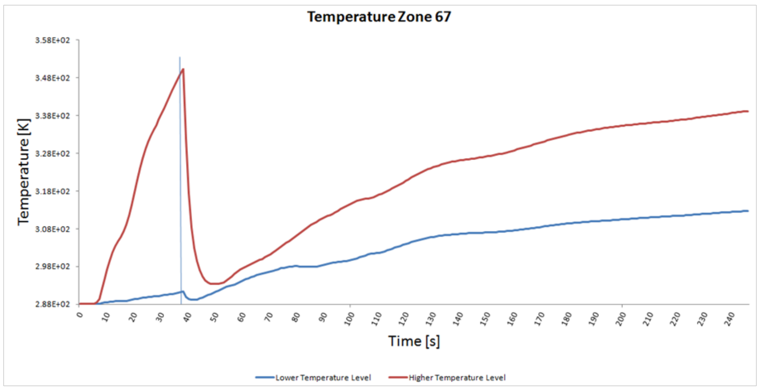


(b) Smoke outputs for zone 93 non-coupled.

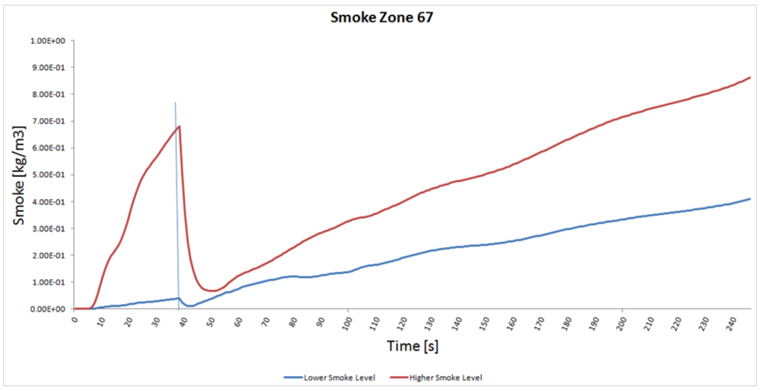


(c) Radiation outputs for zone 93 non-coupled.

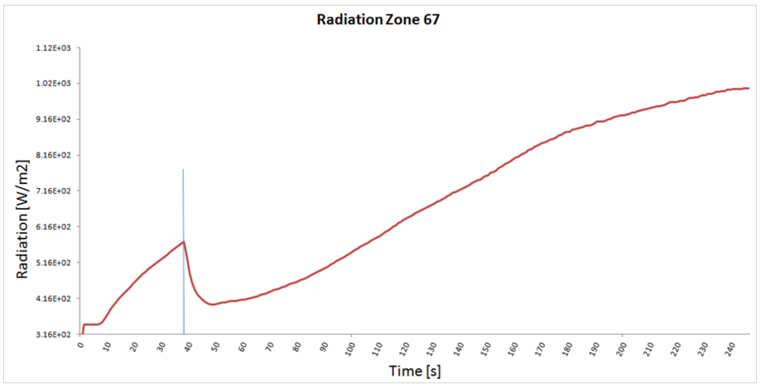
Figure A.52: Temperature, smoke and radiation vs. time for zone 93.



(a) Temperature outputs for zone 84 non-coupled.

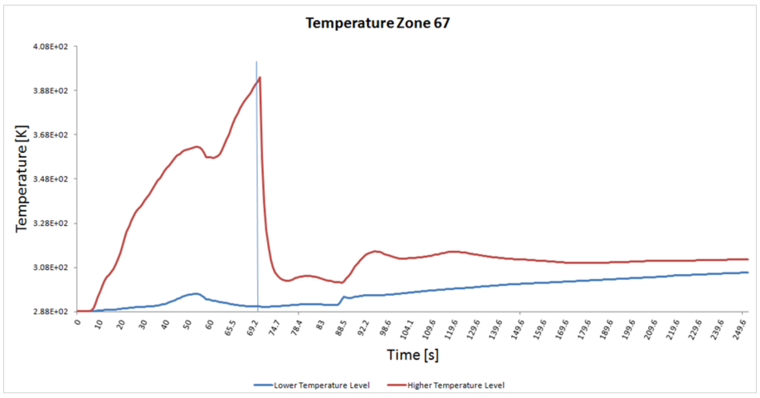


(b) Smoke outputs for zone 84 non-coupled.

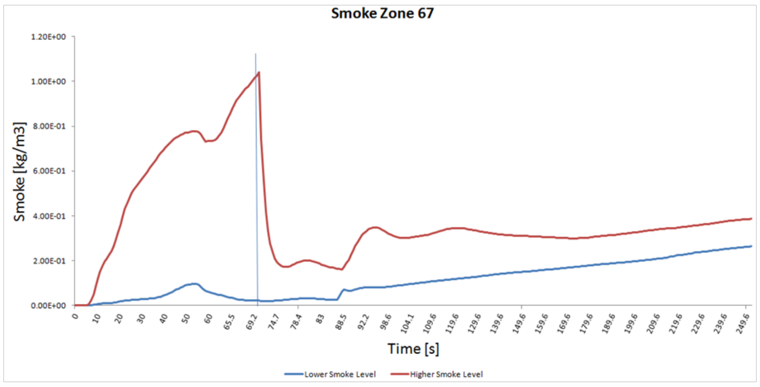


(c) Radiation outputs for zone 84 non-coupled.

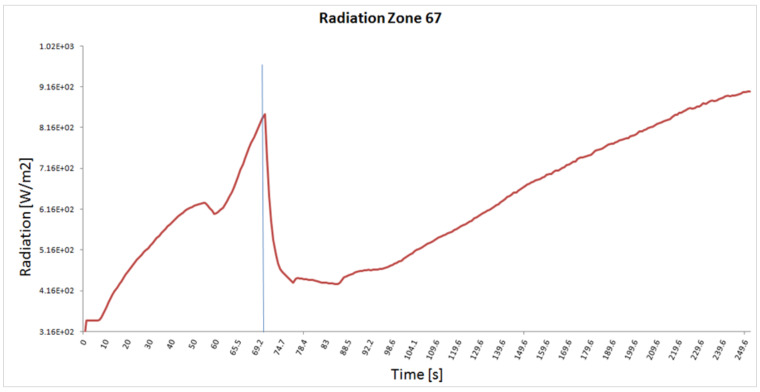
Figure A.53: Temperature, smoke and radiation vs. time for zone 67 serial.



(a) Temperature outputs for zone 84 non-coupled.

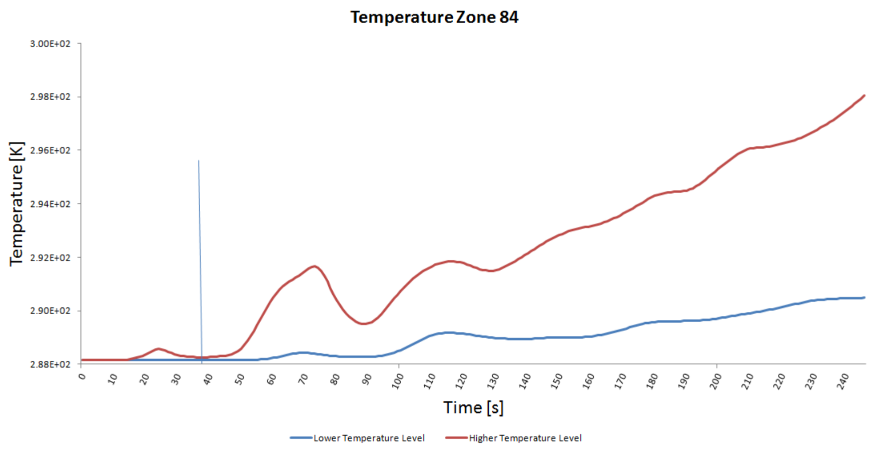


(b) Smoke outputs for zone 84 non-coupled.

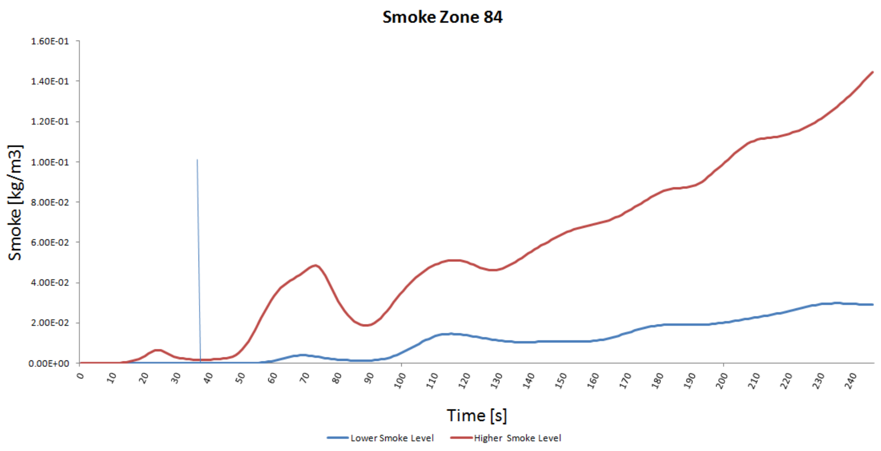


(c) Radiation outputs for zone 84 non-coupled.

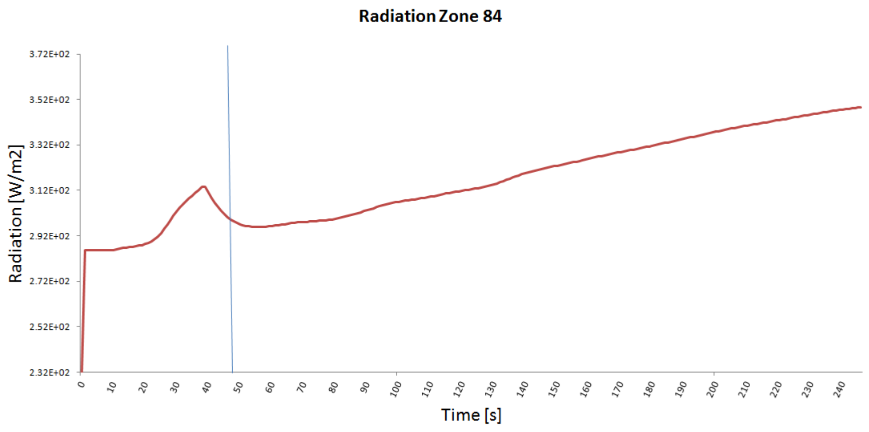
Figure A.54: Temperature, smoke and radiation vs. time for zone 67 coupled.



(a) Temperature outputs for zone 84 non-coupled.

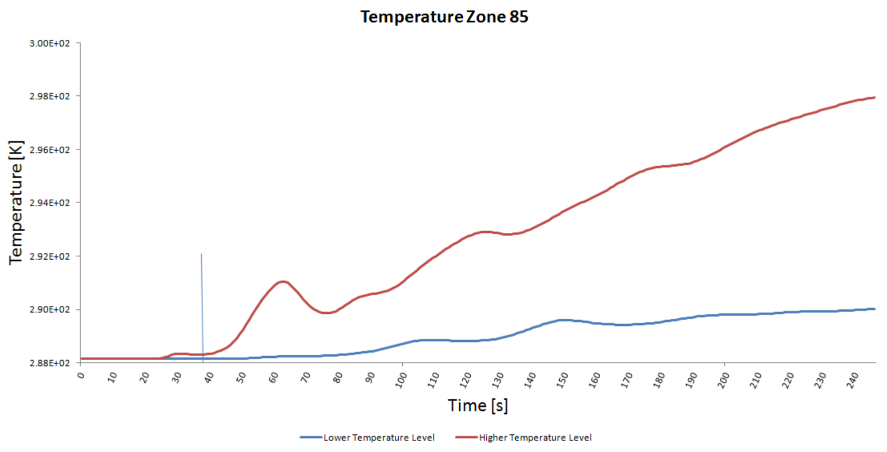


(b) Smoke outputs for zone 84 non-coupled.

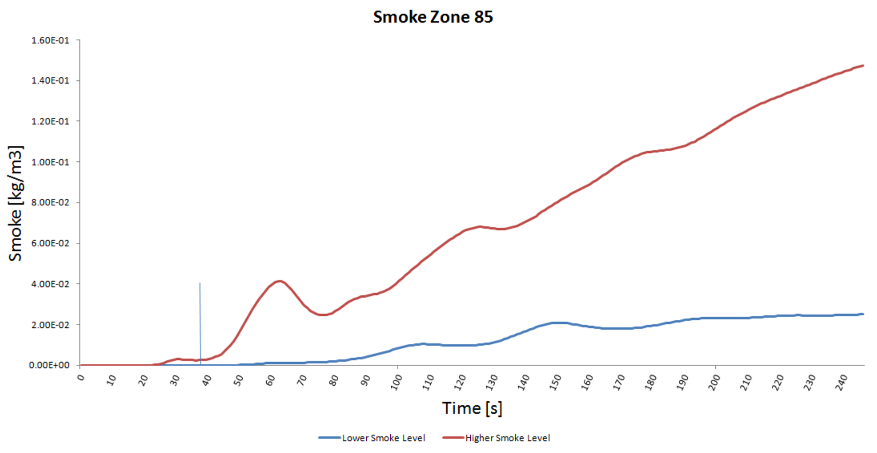


(c) Radiation outputs for zone 84 non-coupled.

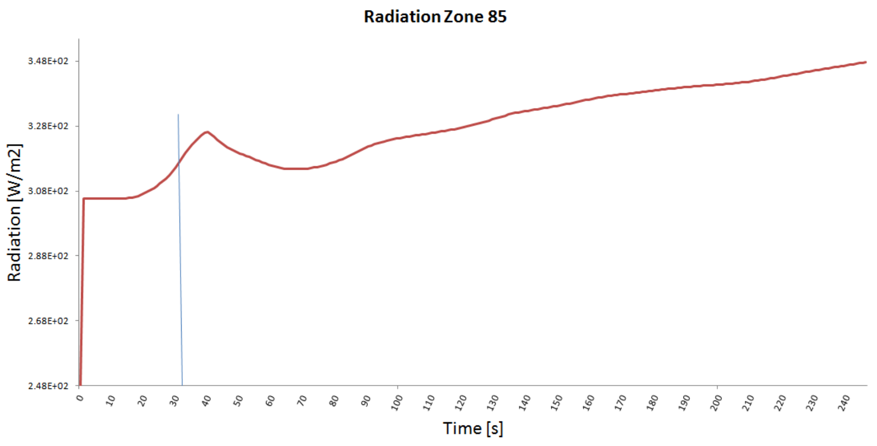
Figure A.55: Temperature, smoke and radiation vs. time for zone 84.



(a) Temperature outputs for zone 85 non-coupled.

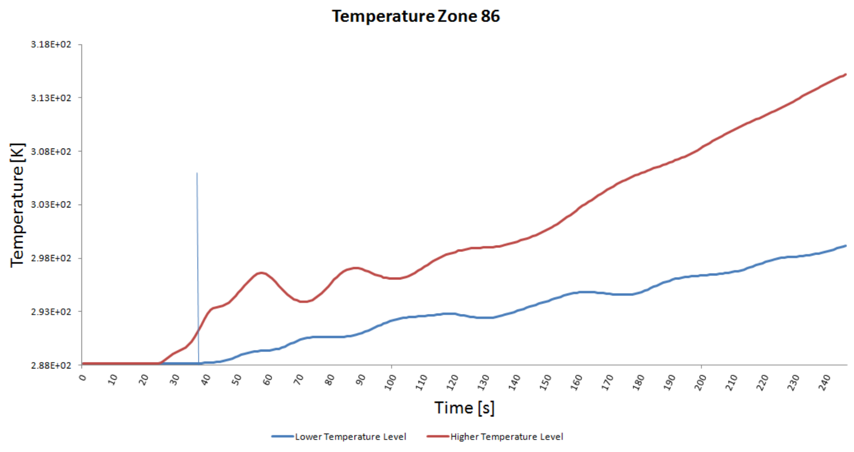


(b) Smoke outputs for zone 85 non-coupled.

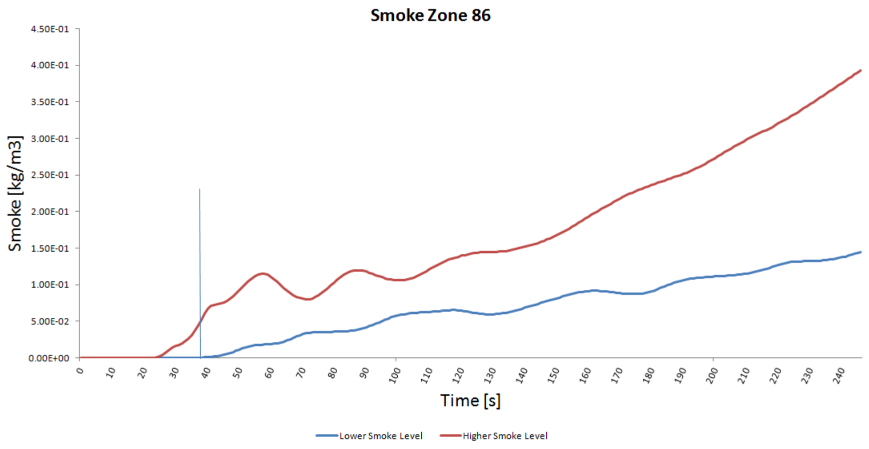


(c) Radiation outputs for zone 85 non-coupled.

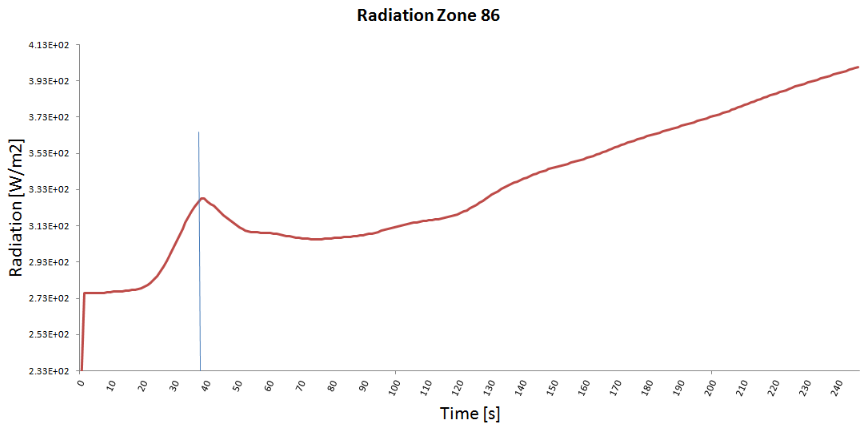
Figure A.56: Temperature, smoke and radiation vs. time for zone 85.



(a) Temperature outputs for zone 86 non-coupled.

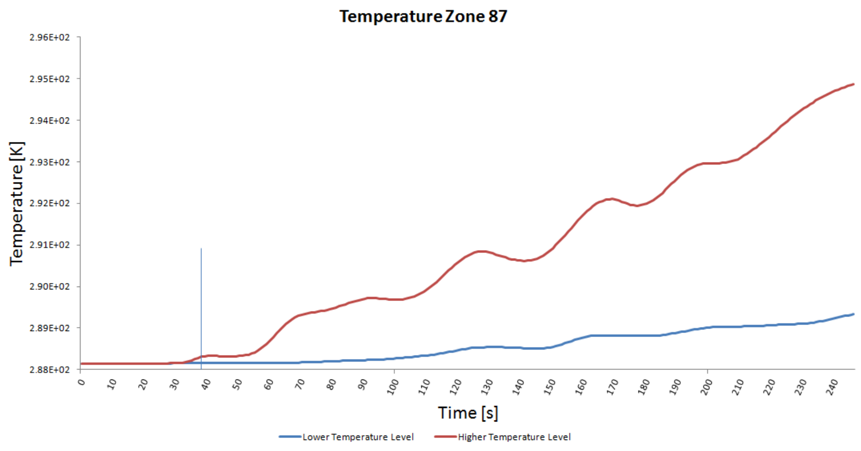


(b) Smoke outputs for zone 86 non-coupled.

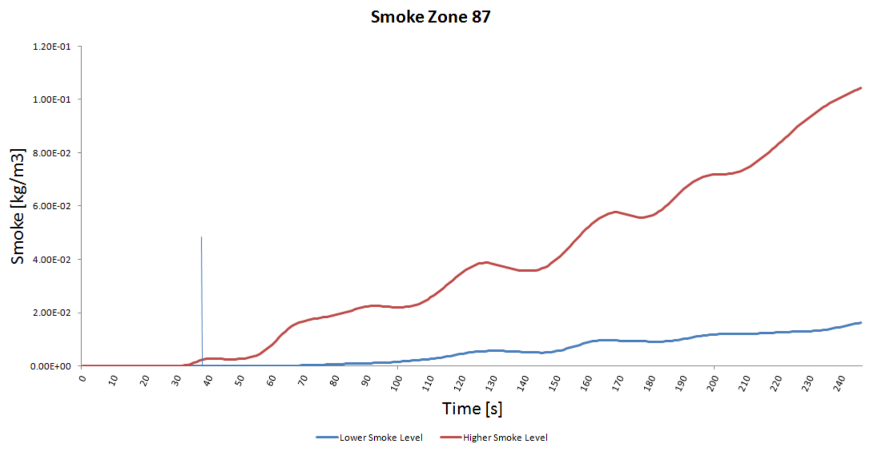


(c) Radiation outputs for zone 86 non-coupled.

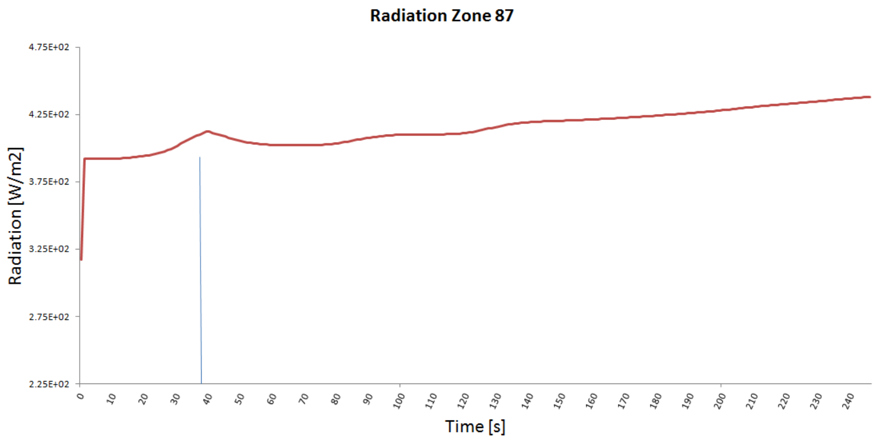
Figure A.57: Temperature, smoke and radiation vs. time for zone 86.



(a) Temperature outputs for zone 87 non-coupled.

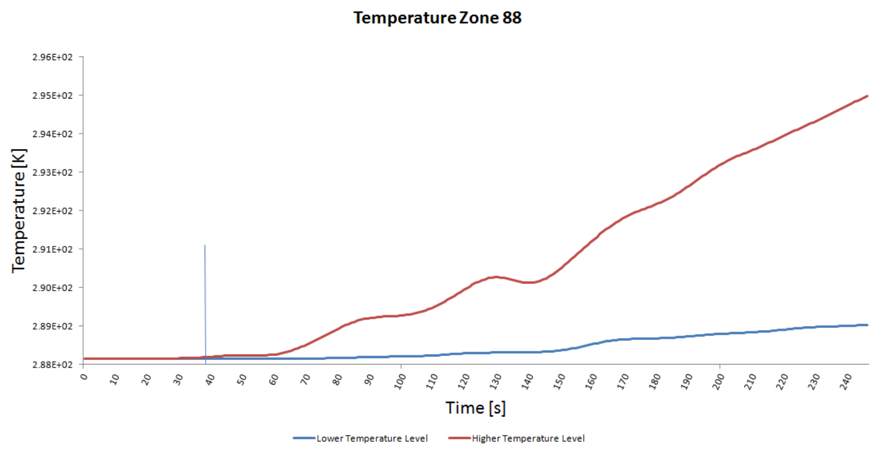


(b) Smoke outputs for zone 87 non-coupled.

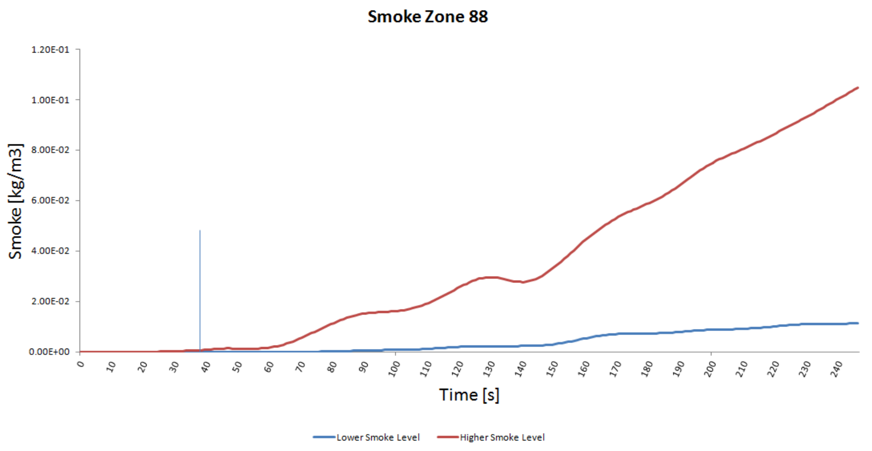


(c) Radiation outputs for zone 87 non-coupled.

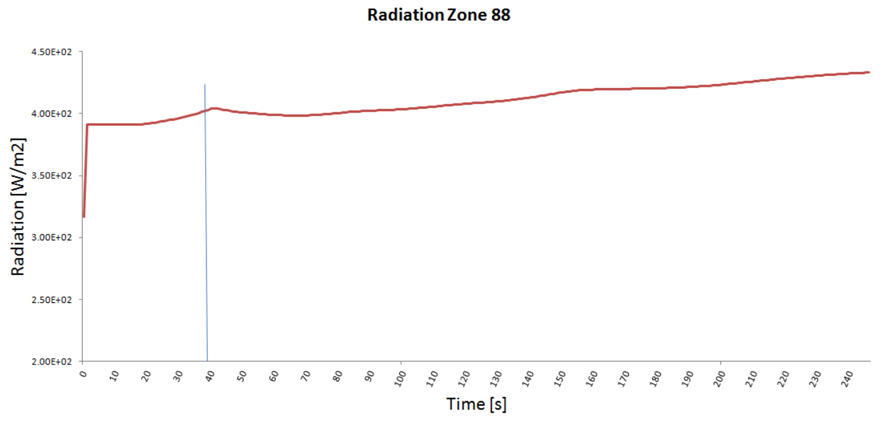
Figure A.58: Temperature, smoke and radiation vs. time for zone 87.



(a) Temperature outputs for zone 88 non-coupled.

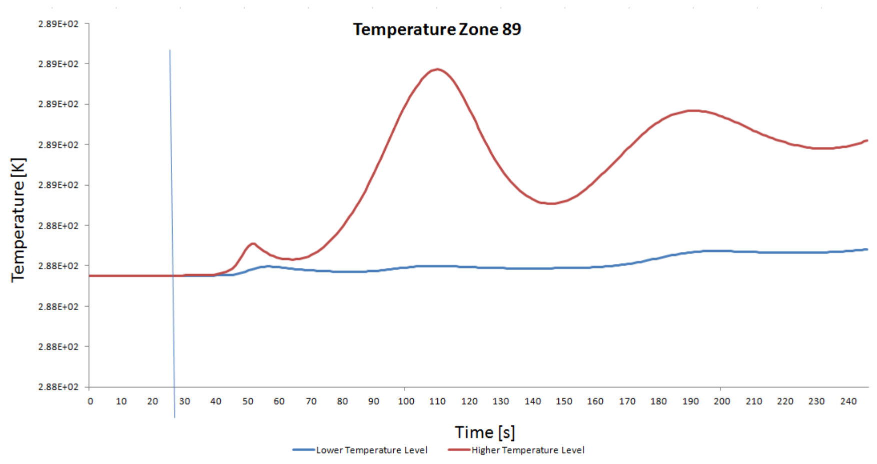


(b) Smoke outputs for zone 88 non-coupled.

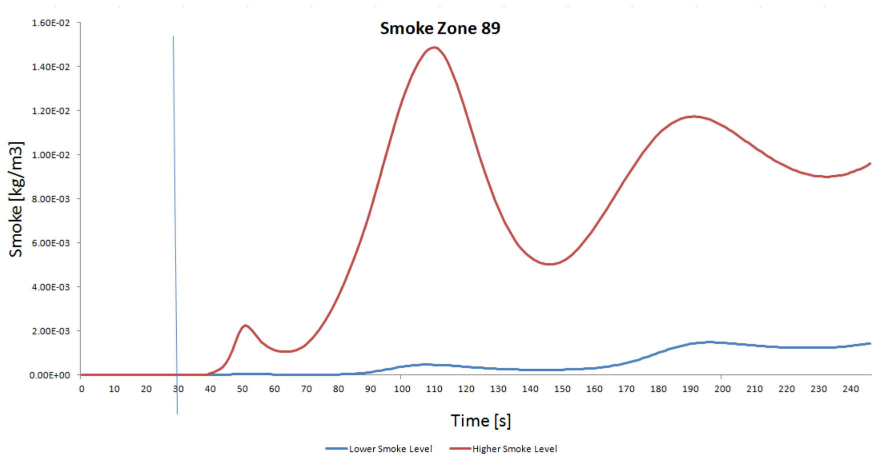


(c) Radiation outputs for zone 88 non-coupled.

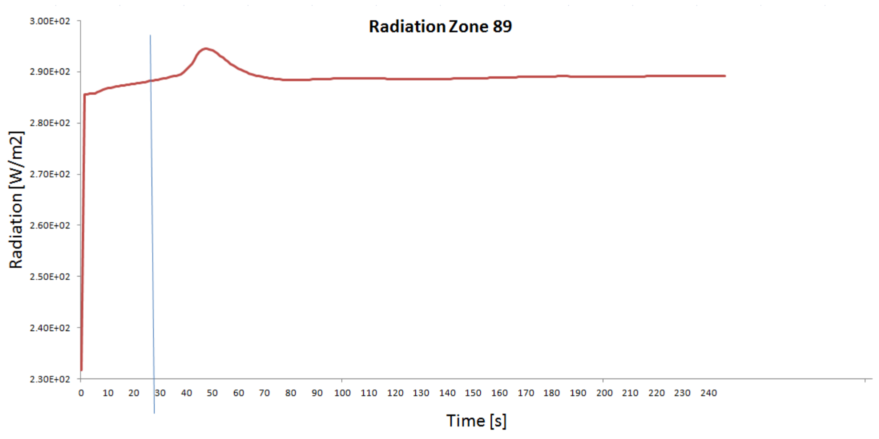
Figure A.59: Temperature, smoke and radiation vs. time for zone 88.



(a) Temperature outputs for zone 89 non-coupled.

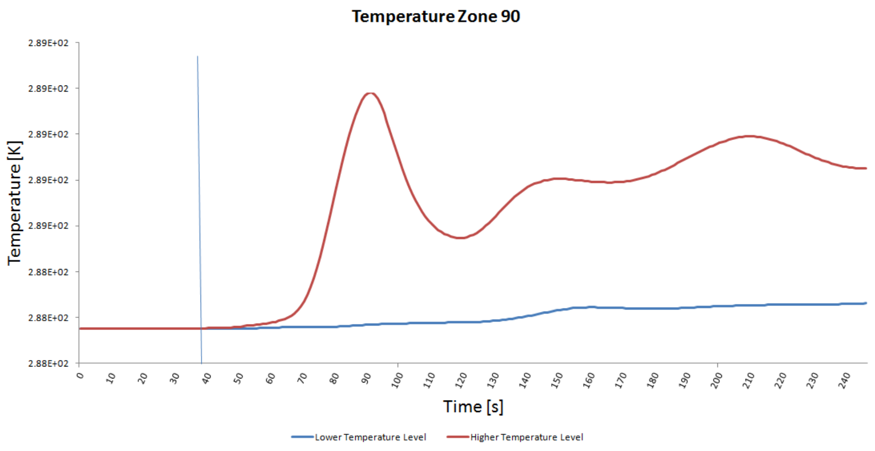


(b) Smoke outputs for zone 89 non-coupled.

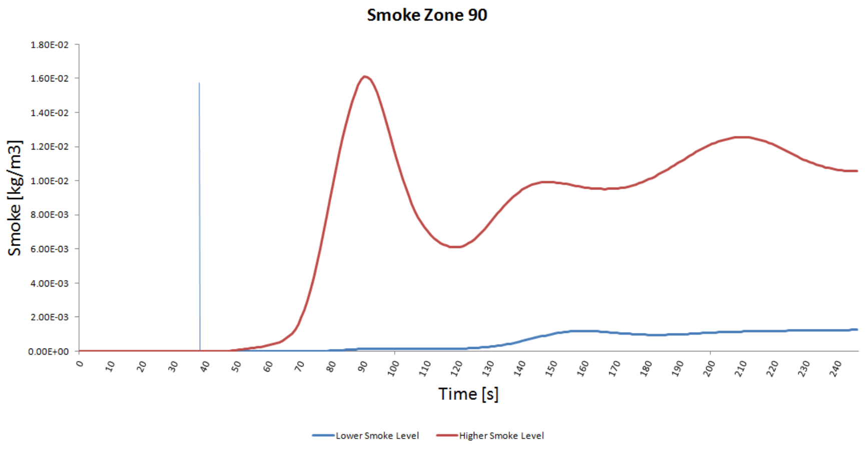


(c) Radiation outputs for zone 89 non-coupled.

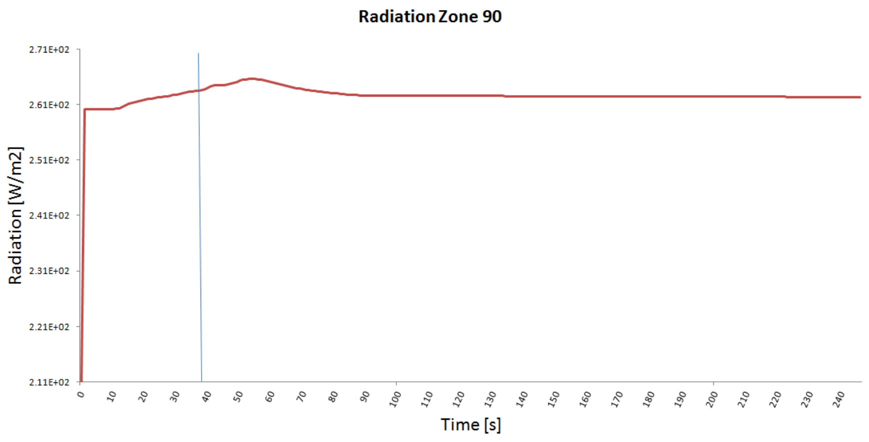
Figure A.60: Temperature, smoke and radiation vs. time for zone 89.



(a) Temperature outputs for zone 90 non-coupled.

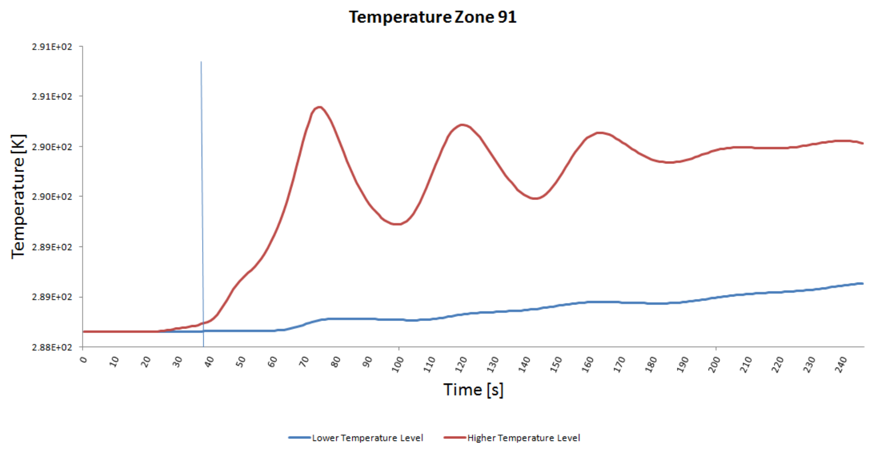


(b) Smoke outputs for zone 90 non-coupled.

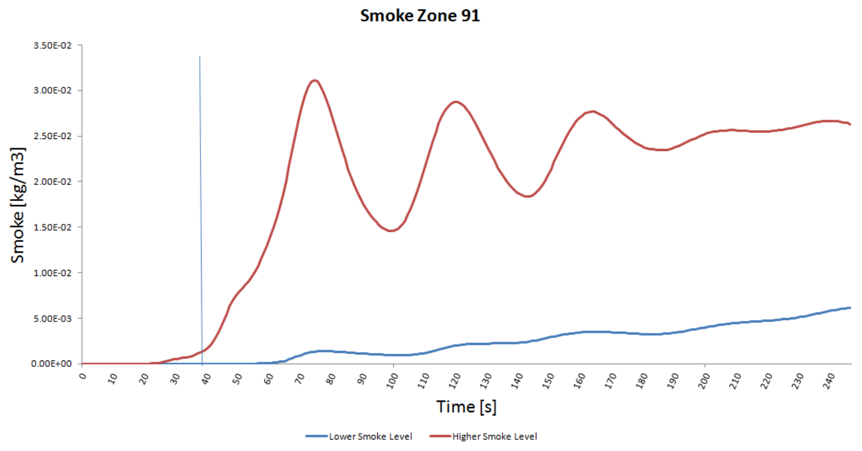


(c) Radiation outputs for zone 90 non-coupled.

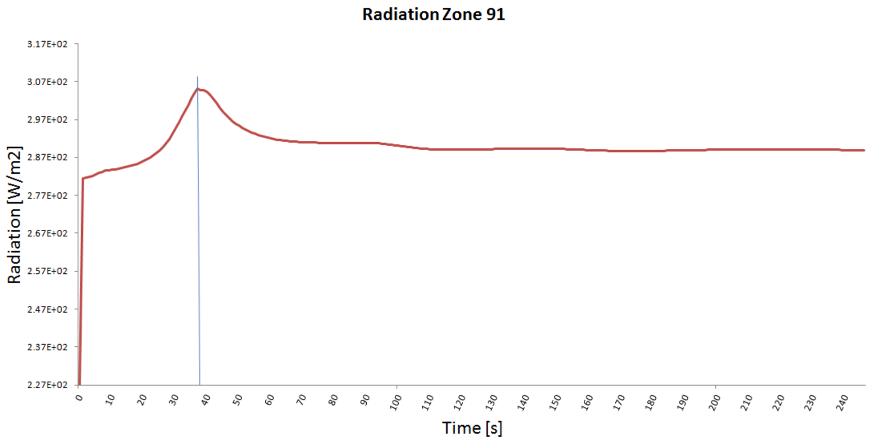
Figure A.61: Temperature, smoke and radiation vs. time for zone 90.



(a) Temperature outputs for zone 91 non-coupled.

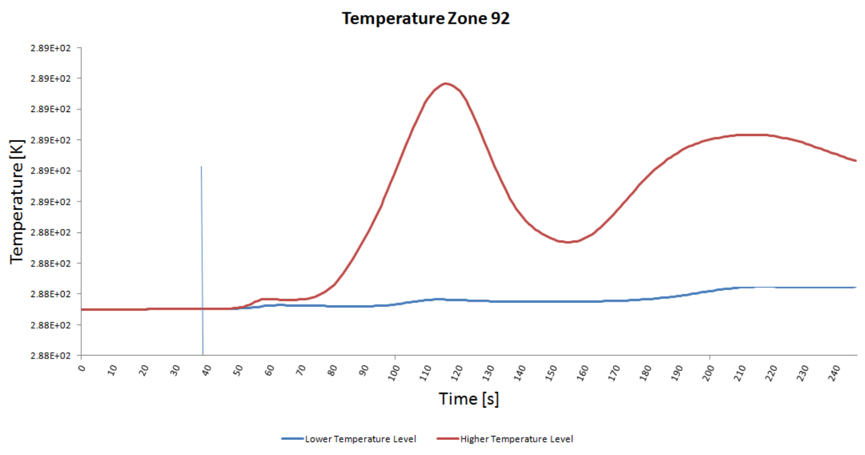


(b) Smoke outputs for zone 91 non-coupled.

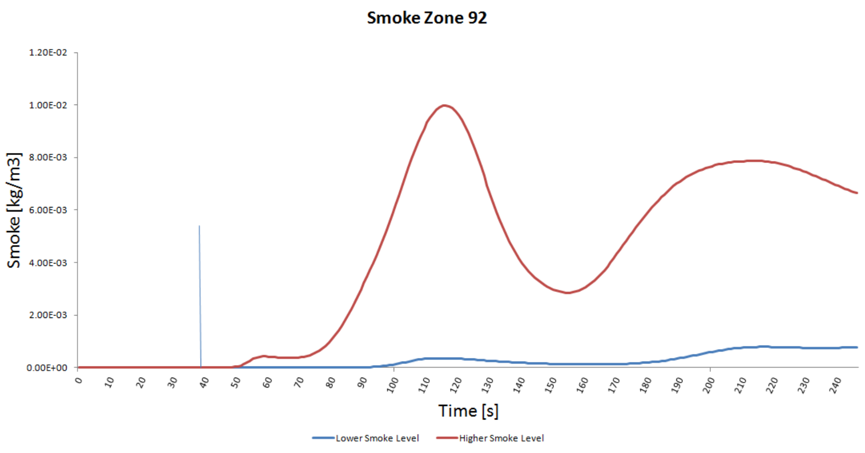


(c) Radiation outputs for zone 91 non-coupled.

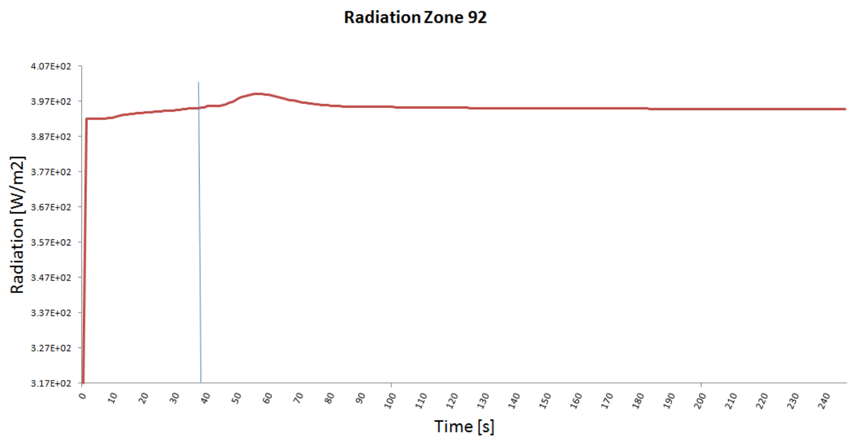
Figure A.62: Temperature, smoke and radiation vs. time for zone 91.



(a) Temperature outputs for zone 92 non-coupled.

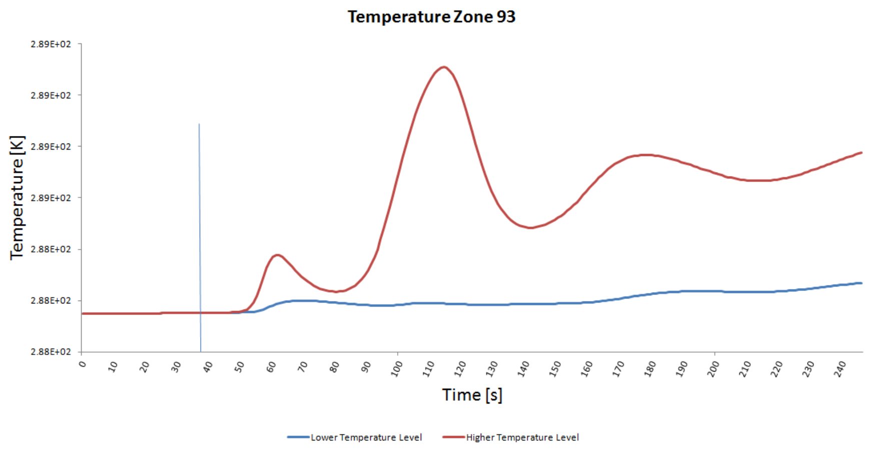


(b) Smoke outputs for zone 92 non-coupled.

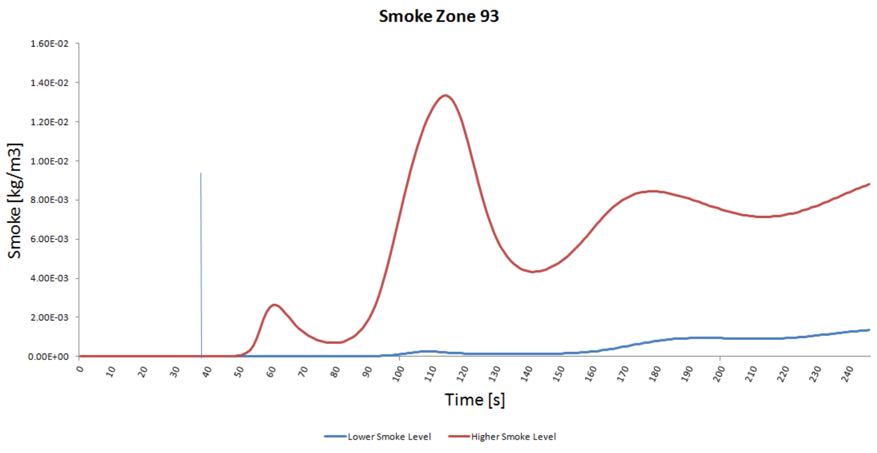


(c) Radiation outputs for zone 92 non-coupled.

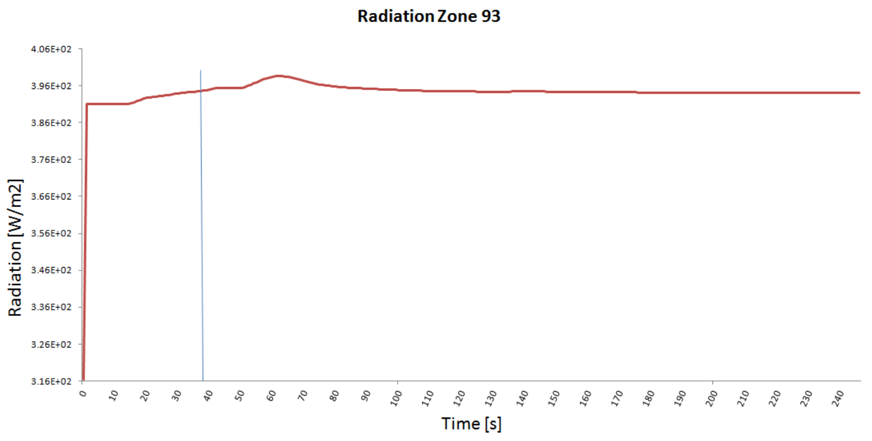
Figure A.63: Temperature, smoke and radiation vs. time for zone 92.



(a) Temperature outputs for zone 93 non-coupled.



(b) Smoke outputs for zone 93 non-coupled.



(c) Radiation outputs for zone 93 non-coupled.

Figure A.64: Temperature, smoke and radiation vs. time for zone 93.