An investigation of techniques to assist with reliable specification and successful simulation of Fire Field Modelling Scenarios

Yanbo Wang

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

May 2007

Acknowledgements

I am so grateful for having had the support of so many people and without it this would not have been possible.

Firstly, I have been very fortunate with my supervisors. Dr John Ewer and Dr Mayur Patel were excellent supervisors, who complemented each other perfectly. They are great scholars and above all, wonderful human beings. John wrote a research proposal that allowed me to hit the ground running, which is always a nice way to start. I also owe him dearly to his guidance, continued advice, support, enthusiasm and attention to detail. Mayur provided the additional perspective, critical questions and many refreshing ideas.

I would like also thank Professor Edwin Galea, for assistance in the research as the Fire Field Modelling expert and other capacities where has to put up with great deal from me. Dr Fuchen Jia and Dr Angus Grandison also deserve acknowledgement for supplemental expertise, providing alternative point of view.

Dr ZhaoZhi Wang also deserves a special mention, as a fellow researcher and good friend who has provided some enlighten discussion regarding issues with the project, as well as other topics.

I grateful acknowledge the financial support of FSEG of the University of Greenwich through its research bursary program. I would also like to thank many others members in the FSEG group for providing support and help, Piotri Rymarczyk, Nathan Hurst, Denis Molodchenko, Hongjun Jiang, Arthur Wasiel , Hiu Xie, Arun Mahalingam and many more. Mrs Francoise Barkshire deserves extra thanks for special help over the years.

Lastly, I would like to thank Professor Brian Knight for the encouragement. Extra special thanks go to my parents, my sister for continuing encouragement and support and my wife, Jie Qiu for her understanding and believing in me.

Abstract

Computational fluid dynamics (CFD) based Fire Field Modelling (FFM) codes offer powerful tools for fire safety engineers but their operation requires a high level of skill and an understanding of the mode of operation and limitations, in order to obtain meaningful results in complex scenarios. This problem is compounded by the fact that many FFM cases are barely stable and poor quality set-up can lead to solution failure. There are considerable dangers of misuse of FFM techniques if they are used without adequate knowledge of both the underlying fire science and the associated numerical modelling. CFD modelling can be difficult to set up effectively since there are a number of potential problems: it is not always clear what controls are needed for optimal solution performance, typically there will be no optimal static set of controls for the whole solution period to cover all stages of a complex simulation, there is the generic problem of requiring a high quality mesh – which cannot usually be ascertained until the mesh is actually used for the particular simulation for which it is intended and there are potential handling issues, e.g. for transitional events (and extremes of physical behaviour) which are likely to break the solution process.

In order to tackle these key problems, the research described in this thesis has identified and investigated a methodology for analysing, applying and automating a CFD Expert user's knowledge to support various stages of the simulation process – including the key stages of creating a mesh and performing the simulation. This research has also indicated an approach for the control of a FFM CFD simulation which is analogous to the way that a FFM CFD Expert would approach the modelling of a previously unseen scenario. These investigations have led to the identification of a set of requirements and appropriate knowledge which have been instantiated as the, so called, Experiment Engine (EE). This prototype component (which has been built and tested within the *SMARTFIRE* FFM environment) is capable, both of emulating an Expert users' ability to produce a high quality and appropriate mesh for arbitrary scenarios, and is also able to automatically adjust a key control factor of the solution process.

This research has demonstrated that it is possible to emulate an Experts' ability to analyse a series of simulation trials (starting from a simplified, coarse mesh test run) in order to improve subsequent modelling attempts and to improve the scenario specification and/or meshing solution in order to allow the software to recover from a complete solution failure. The research has also shown that it is possible to emulate an Expert user's ability to provide continual run-time control of a simulation and to provide significant benefits in terms of performance, overall reliability and accuracy of the results.

The instantiation and testing of the Experiment Engine concept, on a chosen FFM environment – *SMARTFIRE*, has demonstrated significant performance and stability gains when compared to non Experiment Engine controlled simulations, for a range of complex "real world" fire scenarios. Preliminary tests have shown that the Experiment Engine controlled simulation was generally able to finish the simulations successfully without experiencing any difficulty, even for very complex scenarios, and that the run-time solution control adjustments, made to the time step size by both the Experiment Engine and by the Expert, showed similar trends and responses in reacting to the physical and/or numerical changes in the solution. This was also noticed for transitional events seen during the simulation. It has also been shown that the Experiment Engine (EE) controlled simulation demonstrates a saving of up to 40% of simulation sweeps for complex fire scenarios when compared with non-EE controlled simulations. Analysis of the results has demonstrated that the control technique, deployed by the EE, have no significant impact on the final solution results – hence, the Experiment Engine controlled simulations are able to produce physically sound results, which are almost identical to Expert controlled simulations.

The research has investigated a number of new methods and algorithms (e.g. case categorisation, case recognition, block-wise mesh justification, local adaptive mesh refinements, etc.) that are combined into a novel approach to enhance the robustness, efficiency and the ease-of-use of the existing FFM software package. The instantiation of these methods as a prototype control system (within the target FFM environment – *SMARTFIRE*) has enhanced the software with a valuable tool-set and arguably will make the FFM techniques more accessible and reliable for novice users.

The component based design and implementation of the Experiment Engine has proved to be highly robust and flexible. The Experiment Engine (EE) provides a bi-directional communication channel between the existing *SMARTFIRE* Case Specification Environment and the solution module (the CFD Engine). These key components can now communicate directly via status- and control- messages. In this way, it is possible to maintain the original Case Specification Environment and the CFD Engine processes completely independently. The two components interact with each other when the EE is operating. This componentization has enabled rapid prototyping and implementation of new development requirements (as well as the integration of other support techniques) as they have been identified.

CONTENTS

1 Introduction

1.1 Overview

1.1.1 The development and advancement of Fire Field Modelling

During the last two decades, considerable progress has been made in the research of Fire Field Modelling (FFM) [Galea-89]. The development of sophisticated fire models (e.g. radiation models, smoke models, combustion models etc.) and more efficient solution algorithms, together with increasing computational power and reducing costs of computer hardware, have led to the increased use of FFM techniques by fire safety engineers, building regulators, fire services and others for research, development and design tasks in industry.

The need for Fire Field Modelling software means that there are many commercial and non-commercial Computational Fluid Dynamics (CFD) software, e.g. FLUENT[Fluent-6-2], STAR-CD [Eagles-98], PHOENICS[Spalding-81], FLOW3D[FLOW3D-91], JASMINE[Kumar-91], SOFIE[Lewis-97], CFX[CFX97], FDS[Friday-01], and *SMARTFIRE*[Ewer99a], that can and are being used to predict and analyse the effects of fires in safety critical situations. Typically this modelling will include air movement (i.e. flow rate of gas through openings, and production of certain toxic gas species, etc.) and heat transfer (gas and surface temperatures, heat fluxes impinging on surfaces) induced by thermal sources and might also include the modelling of strength reduction and structural failure of building elements and activation times for sprinklers and detectors.

There are two classes of CFD system typically used for FFM simulation, namely: general purposes CFD codes (e.g. FLUENT, START-CD, etc) and FFM specific CFD (e.g. FDS, *SMARTFIRE*, etc). Some of these CFD codes have developed from scientist/research driven batch mode codes that tend to have the typical set-up / configuration, meshing and post processing data analysis tools that are all completely separate from the numerical CFD engine. Such modelling systems are usually not very easy to use, since they are typically hard to configure and require a high degree of expertise – on the part of the user – to specify all of the complexity of the simulation case. Furthermore, in non FFM CFD codes, the user is also expected to activate all of the correct modules and parameters to ensure that the CFD code will behave correctly for FFM scenarios.

Clearly most, if not all, commercial and research based CFD codes are undergoing continuous enhancement and many facilities have been added or further developed during the period of this current research. However, even now, few CFD developers are aiming to provide the code interactivity and automated solution control that has been investigated in this research.

1.1.2 Challenges to use FFM codes

The special difficulties of CFD based FFM warrant some discussion because FFM practitioners are not always CFD Experts, and the area of FFM has some of the more difficult modelling and stability issues facing CFD simulation.

CFD based FFM codes offer powerful tools for the testing of fire safety designs but their operation still requires a high level of skill and understanding of their configuration and mode of operation to obtain meaningful results in complex scenarios. The challenges that FFM users will encounter when using CFD Fire Models include:

- Problem of identification and specification of the simulation scenario in terms of the physical and chemical phenomena that need to considered (i.e. estimation and selection of appropriate input data such as material properties, combustion, radiation, turbulence parameters and boundary conditions).

- CFD simulation is generally highly computationally intensive – especially for large scale geometries (e.g. whole buildings, stations, airport terminals, etc). The CFD governing equations are non-linear and are generally solved iteratively for every computational cell of a highly refined computational mesh. Many hours or even days are required for the solution of problems of fire safety interest.

- Problem of understanding and controlling the numerical solution algorithm (i.e. to guarantee convergence, consistency of results and stability).

- Performing the actual CFD computation itself requires operator skills of a different kind. For instance, selection of appropriate computational domain and selection of the best mesh to obtain reliable and accurate outputs, while minimising the computational time that is used. Successful simulation results are very likely to depend on the quality of the geometry specification and the meshing.

- CFD is typically very difficult for novice users. It will usually involve a complicated manual problem set-up with a huge number of modelling choices, e.g. Long processing period (usually requiring many halts, analysis and correction, followed by restarts). Even if the CFD simulation is able to get to a solution, it is not always clear if the final results are correct or reliable. This requires in-depth analysis and interpretation of the voluminous outputs generated by the CFD models and validation of the modelling outputs against suitable experimental fire data.

There are also significant dangers of misuse of FFM codes if they are used without adequate knowledge of both fire science and of numerical modelling [Kumar-01]. The most experienced and Expert users of FFM codes usually overcome the long learning curve through a Fire Safety Degree, an MPhil or PhD studies. However, there is not always time available for fire safety engineers (and other FFM users) to learn about all of the potential issues relating to the use of FFM or the interpretation of FFM data.

These issues have created an increasing need for adding human expertise to FFM codes in order to support users who lack detailed CFD knowledge, but nevertheless have to use or make decisions based on FFM, for instance, consultants designing buildings who may have limited CFD knowledge or regulators presented with CFD modelling results who have to approve designs but do not know if the model has been run appropriately. Automated control of the simulation is also highly desirable, although Expert users will always need the ability to re-justify/correct the control parameters when things go wrong and in some cases this will require a complete re-specification and re-meshing. It is a tedious task for Expert users to monitor for early signs of solution changes and to make necessary parameter control changes accordingly.

It is often the case that some Expert users will take extra caution and run the case with an overly fine mesh and overly fine time step size in order to try to get the simulation results without frequent interventions – although this solution process will probably be highly non optimal. Other users will be constrained by available time and will run with defaults and have to use the results whatever the quality. Others might not know of the quality issues and might not check the solution quality and hence might make decisions based on flawed/incorrect results – with potentially catastrophic consequences.

1.1.3 Outcomes of the research

This dissertation describes a systematic approach to investigate potential supporting techniques for the accurate, efficient and robust use of CFD based Fire Field modelling software through set-up, meshing and CFD solution control. The investigations, research and prototyping have used an in-house FFM code at the University of Greenwich, called *SMARTFIRE* as a research vehicle for testing and developing many of the concepts form the core of this thesis. This decision to use *SMARTFIRE* was also aided by the fact that there is a significant body of in-house expertise about the FFM and the *SMARTFIRE* system, which was hugely beneficial to the knowledge elicitation process. The ultimate outcome of this research is the identification and creation of a set of robust and effective procedures for managing key stages of a FFM simulation so as to ensure that a solution is reached, in as short a time as possible and with minimal need for human intervention. This body of research has been instantiated as the prototype Experiment Engine, which has demonstrated that it is capable of emulating an Experts' ability to run a FFM simulation. The EE embodies a number of control strategies that come from investigating the steps that an Expert user would make to ensure that they obtain a successful set-up and a suitably high quality mesh for an arbitrary scenario. It has been identified that an Expert user will typically perform a number of test experiments in an attempt to understand both the mesh quality and the simulation stability. Another issue that has been researched is the Expert user's ability to control the solution process by continually monitoring and adjusting the numerical processing. The capture of these methods and Expert knowledge has allowed the prototyping of a tool which performs these actions during the set-up, meshing and simulation of a previously unseen FFM scenario. This prototype "Experiment Engine" has been seamlessly integrated into the *SMARTFIRE* software architecture and, as a result, the robustness and ease of use of the FFM environment has been greatly enhanced. Numerical stability and convergence have been demonstrated to be more assured, and this maximises the successful rate of fire simulations. The Experiment Engine also demonstrates significant performance gains over non Experiment Engine controlled simulations for complex real world fire scenarios. The Experiment Engine offers complete user support during the whole simulation process, can generally obtain good solutions without user intervention, and therefore makes these FFM techniques more accessible for novice users.

1.2 Research Objectives

In the light of the difficulties facing the use of CFD FFM, especially for novice users, the main objective of this investigation has been to research and test potential supporting techniques that can enhance the robustness, performance and accuracy of CFD based FFM. It is intended that this knowledge will assist with the reliable specification and successful simulation of Fire Field Modelling Scenarios.

In order to understand the difficulties experienced by novice CFD FFM users, a thorough investigation (please refer to chapter 2) was conducted into the issues and problems that can be encountered when setting-up or using Fire Field Models. The key stages of CFD based FFM, that can cause problems, were identified. It was also observed that Expert users are generally able to deal with many of the problems encountered through the critical stages of creating and performing a FFM simulation.

In addition, the particular expertise that is needed to overcome these encountered difficulties is identified. The investigations of the problems – and their mitigating strategies – lead to the following research questions.

1.2.1 Main Research Questions

- To what extent is it possible to emulate an Expert users' ability to analyse a series of trials starting from a simplified, coarse mesh simulation run (or even to learn from a complete simulation failure) to help make a better set-up and meshing solution.

Prior to this investigation it was not known how Expert users would model and simulate a previously unseen scenario. Analysis of how Experts tend to use a CFD FFM code showed that they have an ability to learn from "quick and dirty" solutions in order to make the set-up and meshing better for the particular scenario. Furthermore, investigations have demonstrated that this expertise can be emulated using a Knowledge Based control module.

Appropriate set-up and meshing are vital ingredients to the success of the simulation both in terms of getting a solution and ensuring the accuracy of the solution. Most CFD based FFM codes are structured around the numerical algorithms and typically contain three main components: a pre-processor, a solver and a post-processor. These components usually operate in sequence. In the pre-processing stage, the user's activities include: the definition of the complete geometry of the computational domain of interest, selection of the most appropriate physical models for the scenario, specification of appropriate boundary conditions, followed by the generation of a sufficiently high quality mesh. All of this information is then passed to the Solution module (typically the pre-processing information is usually stored in files). In the next stage, the solver will calculate increasingly updated physical properties in all of the mesh cells using all of the boundary conditions to drive the solution. The calculations are derived from a numerical time and spatial discretization of the governing Partial Deferential Equations (PDEs) that determine the transport of all of the physical properties. In the post-processing stage, the huge amount of output numerical data that has been generated during the processing stage is usually analysed and represented using graphic or visual representation of some of the key solution features in the domain of interest for a specified time. These three distinctive stages of the operation are typically separate and have been defined in this way due to the manner in which most CFD software has been developed and constructed.

This modularity can be quite inconvenient, for example, a problem found in later stages of the simulation (e.g. due to poor quality meshing or because a particular physical model should have been activated) means that it will be necessary to re-run the simulation from the start using a revised set-up.

The inherent complexity and difficulties facing CFD based FFM mean that, even an Expert user, is not guaranteed to obtain a successful and well converged solution from the first attempt to simulate a previously unseen fire scenario. However, it has been observed that human Experts can subsequently improve on an initial problem set-up and meshing, based on qualitative assessment of the solution state/results of the failed/completed simulation or even from analysis of results from a simplified/coarse mesh version of the simulation. It is often the case that Experts cannot make definitive conclusions based on a single trial run. Consequently, it appears to be necessary to run more trials, each with improved set-up and meshing, from information provided by the previous trials. Typically, an Expert would also conduct a mesh independence study to give some assurance that the final solution is not dependent on the nature of the mesh used for the final simulation.

The initial stage of this research was to investigate if it possible to use human expertise to support novice users through set-up and mesh stages of creating a FFM simulation and, if this is possible, to investigate any tangible benefits due to emulating how Expert users learn from experimental trials and from failures to make the problem set-up and meshing better for previously unseen fire scenarios. These improvements could be: an improved successful rate of simulating previously un-encountered fire scenarios, better solution reliability, prevention and recovery from poor quality set-up and meshing, improved solution robustness and improved ease-of-use of running Fire Field Models.

The first research question is obviously only a partial solution to the problems encountered in the whole FFM simulation process. However, it has been shown that these investigations provide a solid foundation on which to construct a methodology for the successful simulation of arbitrary fire scenarios. The second research question is:

- To what extent would it be possible to emulate an Expert's ability to control the production run assuming that appropriate set-up and meshing has been obtained in the experiment stage.

The investigation of exactly how Expert users achieve successful FFM simulation outcomes indicates that it is vital for them to be able to continually monitor the simulation state/results and to make control adjustments as appropriate. These run time configuration changes are made to ensure solution convergence and to ensure stability of the simulation. Ideally the procedures and knowledge that Expert's apply, to ensure a successful outcome, must be emulated in an automatic and efficient manner, in order to improve productivity of using CFD based FFM techniques. Although it is understood that an Expert user can typically optimise a FFM simulation by performing run-time adjustment of various solution control parameters – based on assessment of the simulation state – the generally long run times for CFD FFM simulations mean that this is tedious at best and often completely impractical, since it is unrealistic to expect an Expert to sit in front of a computer during the whole simulation process of any non-trial scenario which could last hours, days or even weeks.

This investigation is not a completely new idea in terms of emulating an Experts' ability to efficiently control a simulation. Former research, at the University of Greenwich, into an "Intelligent Control System" (ICS) [Janes-02] demonstrated some limited success. However, to investigate the solution control techniques in an isolated manner – without adequate assurance of having the appropriate set-up and sufficient quality of meshing – inevitably had flaws, because the success of any simulation has to be built on having appropriate set-up and meshing. Conversely, it has been demonstrated that, having used a suitable set-up and mesh, the solution control investigations have also been beneficial. It has been possible to assure that the experiments stage would lead to tangible performance gains, greater levels of convergence and greater assurance of reliable outcome for arbitrary fire scenarios and to offer complete support for the use of FFM techniques. These benefits are desperately needed by novice users but can also be of significant support to Expert users for difficult and/or large scenarios.

Having identified the need for understanding how an Expert is able to mitigate the problems of running a FFM scenario, a follow on research question is defined as:

- Is it possible to instantiate the Knowledge acquired through this research into an automated software system (a Knowledge Based System integrated into the software architecture of a CFD code) such that the captured knowledge offers complete and automated user support for using the software through all stages from set-up, through meshing and also the run time control. The ultimate purpose of this system is to enhance robustness, stability and convergence of the CFD simulation, to provide performance gains and consequently enhance the rate of making successful fire modelling simulations?

Research into the two previous research questions produces state of the art techniques and application specific expertise to assist with the reliable specification and successful simulation of Fire Field Modelling scenarios. However, it is equally important that these techniques can be instantiated and demonstrated as a supporting module which offers robustness, flexibility and is practical to use. Prior to this investigation, much of the related research has focused on finding a solution to a particular problem that can be encountered during the CFD simulation process. For example, there are a number of automatic meshing, mesh refinement; and other techniques that aim to control the CFD solution. However, few – if any – investigations have attempted to fully automate and integrate these techniques into a robust and fail safe component that is embedded into either a CFD or FFM. It was not formerly known if such techniques could be integrated together to offer complete support to the whole simulation process and yet still be flexible enough to accommodate a differing range of user needs (i.e. the Experts may want to do the set-up and meshing manually and then use the automatic solution control capability of the

Experiment Engine, while novice users would typically benefit from having maximal automation of the simulation process).

No investigation of automated support can hope to be complete, so it is also vital that the architectural design is fully extensible to allow integration of any future support technologies and/or knowledge. It was found that these investigations produced a significant opportunity to design a practical, effective and efficient tool which would benefit all users – regardless of their experience level – to be able to use FFM techniques, and to make the CFD based FFM techniques more accessible for novice and Expert users alike.

It was soon realized that there are many aspects of a simulation that an Expert user could change during the course of trial runs/experiments and the production run. With so many degrees of freedom, attempts to make a control system comprehensive enough to cover all the aspects of the set-up, meshing and solver control would make the proposed system less efficient (i.e. more degrees of freedom mean that significantly more tests are needed to reach a satisfactory conclusion, whilst the running of tests consumes time and compute resources and is, above all, costly) and would be fatally flawed because even the experienced Expert user's do not always fully understand the exact impact of complicated combinations of parameter changes, on the solution.

The trade-off, between the comprehensiveness of the response and the efficiency, means that it is necessary to find the most frequent, effective and critical changes that would be made by Experts during the trial and production runs. During the interviews with the Expert users, it was decided that time step size, mesh cell budget and the mesh distributions are among the most important parameters which Expert users typically modify to influence the outcome and quality of a simulation.

1.2.2 Subsidiary research questions

During the course of these investigations it was realised that there were number of subsidiary questions that would need to be answered.

- Can an arbitrary "real" fire scenario be characterised into a distinct category and is it possible to build a corresponding library of meshing parameters for each category?
- What are the indicators to signal that a mesh needs to be refined, and to what degree, and how should it be refined?
- In what circumstances would Expert users tend to change times steps size, to what degree should the time step size be changed, and what possible impact would the Expert expect these changes to have on the solution?
- How can the convergence of every time step size and simulation stability be assured throughout the whole simulation process?
- Is it possible to prevent and recover from all types of solution fault?
- Can the system provide improvements in terms of simulation speed when compared to non-controlled simulations?

There is little or no knowledge to what degree or to which parameter and according to what conditions, the Expert users should change, in order to justify or optimise the set-up, the meshing or to control the solution. Consequently, the first task of this investigation is to identify and formalize the actions that Expert users would take during the course of a simulation (including any trial runs). In addition, the conditions (simulation state and/or results), leading the Expert to make decisions, have to be identified and the degree of the change has to be measured. The answers to these questions should provide the fundamentals to the Experiment Engine's knowledge base. When this knowledge is obtained, collated and refined, then it becomes possible to design an appropriate architecture for a conceptual Experiment Engine that would be capable of applying the information, knowledge and techniques to emulate an Expert user. This system will enable a better set-up and meshing based on adaptive trials and efficient control of the production run, must then be devised.

1.2.3 Design and Implementation issues arising during this research

During the investigations, it was realised that there were a number of important issues that needed to be considered for the design and implementation of the Experiment Engine.

- Given the complexities of the FFM software, how should the Experiment Engine concept fit into an existing FFM architecture?
- Is the design flexible and extensible, in order to accommodate future developments and research directions? and,
- Is there a role for the Experiment Engine to provide a framework for the operation of other CFD supporting techniques (e.g. ICS and Group Solvers) that also aim to facilitate and/or optimize fire modelling simulations?

Investigation of these questions will prevent taking an inappropriate design decision or implementation of limitations within the Experiment Engine concept.

As previously mentioned, most CFD software packages have been designed to run the simulation in a strict ordered sequence, i.e. CASE SPECIFICATION/SET-UP before CFD SOLVER PROCESSING before POST PROCESSING. In this structure, the CFD solver is generally unable to communicate with the set-up that has already been made. If a problem is detected in a later stage, that was due to inappropriate set-up, then there is nothing that the software can do to correct the problem, other than to re-run the simulation. In order to change this mode of running to offer complete support during the whole simulation process, dynamic interaction is needed between the CFD solution process and the set-up environment. The existing software architecture has to be revised in order to mitigate this limitation, hence, it has been decided that the Experiment Engine should provide a two way communication channel between the Set-up Environment and the CFD Solver.

Solutions to all of these problems / issues should lead to the seamless embedding of the Experiment Engine into the *SMARTFIRE* architecture, with the ultimate goal of being capable of emulating Experts' ability to better set-up and meshing for the simulation and also to make run time solution control more efficient and effective with almost no requirements for user intervention.

It is believed that Experiment Engine concept should be able to use any existing techniques and deliver tangible benefits in terms of improved successful rate of simulating real world fire scenarios, greater solution reliability, prevention and recovery from "poor quality" set-up and inappropriate/"poor quality" meshing, give greater robustness and give better ease-of use of the software.

1.3 Research Methodology / Plan

In order to achieve the research goals proposed in this investigation, in a manageable way, three key phases has been identified in which to focus the research efforts.

The first phase of this work is to gain a solid background in understanding the area of fire field modelling and in particular, perform a detailed investigation of the difficulties experienced by novice CFD users when having to specify and run a fire modelling scenario. This will include a major study of the available literature. A decision was taken to research the expertise based on Expert users' formulation, implementation and specification of simulation scenarios within the *SMARTFIRE* system. This decision was made because there is considerable in-house expertise about the *SMARTFIRE* FFM environment and the *SMARTFIRE* system is widely used and is specifically targeted at novice and in-experienced CFD users. A part of this study involved attending the MSc courses "Principle and Practice of Fire Modelling" and the "Principles and Practice of Evacuation Modelling" where novice users were observed meeting a FFM for the first time and awareness was gained of some of the difficulties of using FFM.

As part of this familiarisation process, it was necessary to undertake a research study of issues relating to the specification and nature of fire definition within the FFM software. This study was intended to produce reliable data as to the most appropriate form of fire specification and recommendations (e.g. as to the use of the Heskestad fire height correlation model [SFPE-02]). In addition, it was necessary to undertake a range of validation studies using various fire representations in order to better understand the strengths and weaknesses of the various representations and to suggest further avenues for development.

From the work completed in the first phase, an appreciation of the importance, to the specification of fire field modelling scenarios, of the decision made during the specification of a case and how these affect the simulation should be developed. This lead to the development of routines to check the consistency and appropriateness of the configuration for an arbitrary simulation scenario so that the parameters passed to the CFD engine are, as far as possible, the most appropriate set of parameters and control setting possible for that scenario. In this stage, it was also very important to investigate the techniques for appropriate meshing of scenarios based on the nature of the problem to be solved and to categorize the various fire field modelling cases that are likely to be encountered. This included formulating and conducting interviews with fire field modelling Experts, after which it was possible to begin to formulate case specific meshing rules and key indicators for arbitrary case recognition. These knowledge elicited meshing rules and case recognition methods could then be implemented as supporting technologies with FFM set-up environment. In addition, it was determined that it was necessary to investigate how to provide an adaptive mesh refinement method, within the automated structured meshing system of the FFM environment, in order to support the further investigations.

The earlier investigations serve as important building blocks for the research of techniques and knowledge appropriate for the Experiment Engine (EE) concept. The EE concept is intended to be capable of using a simple coarse mesh analysis to enable a better understanding of the simulation scenarios so that this knowledge can be used to optimise (where appropriate) the simulation set-up and meshing, and to make the settings as robust as possible. This is analogous to the mode of operation of fire field modelling by CFD Experts, and can be described as partial automation of the friendly assistant expert.

Furthermore, the operation logs of the Experiment Engine decision making process will provide training support function that will help the user to make the correct/most appropriate choices during the specification and meshing of a particular scenario. Also the additional outputs from Experiment Engine help users to be able to identify the simulation states more thoroughly and precisely.

The final phase of this investigation saw the completion of the knowledge elicitation and technology investigations undertaken in the second phase, and then concentrates efforts on techniques that will further enhance and extend the Experiment Engine concept. In particular it was deemed necessary, to improve and diversify the solution control techniques. Eventually, it is intended that an enhanced Experiment Engine will provide a fully robust and automatic control technique to take control of the whole fire simulation process. Finally, to give assurance of robust and appropriate behaviour it is necessary to undertake comprehensive testing and validation of all of the new features. Any further improvements or requirements that emerge in the testing phase will also be researched in this phase as well.

On completion of the above research tasks, the Experiment Engine concept was fully instantiated and tested and was used to produce the final results for discussion and evidence supporting this thesis (for details please refer to Chapter 8). The summary and the conclusions of these considerations are presented in Chapter 9. Finally, any issues relating to further work need to be identified in order to realise the full potential of these techniques in future research.

1.4 Contribution/major achievements

The following summary indicates the significance of this research that has made CFD based Fire Field Modelling software more robust, more reliable, automatically monitored and more accessible to the wider audience and more supportive to CFD novice users.

- Identification of shortcomings and limitations of most existing CFD software platforms in terms of providing a complete user sport during the whole simulation process.
- Investigation of how an Expert user will mitigate the problems and issues that are typically encountered when running a previously unseen FFM scenario.
- Investigation of a sophisticated Experiment Engine concept that is capable of emulating the key aspects of a FFM Expert user's decision making based on a series of trial runs to make the set-up and meshing as robust as possible and with newly developed advanced control techniques, the Experiment Engine is able to take control of the whole simulation process with convergence assurance.

This research identified a number of existing techniques that offer partial user support to perform particular tasks during the CFD simulation process. It also highlighted some specialized tools /Expert systems for automating CFD applications, such as mesh generation, run-time solver parameter control etc. However, even now, few – if any – CFD software systems are able to provide a complete supporting environment to users during the whole simulation process. Rather the existing systems offer techniques developed mainly targeted at specific phases of the simulation process and these tools and techniques have little interaction with each other and are not integrated to become a complete functional module that offers a complete supporting solution. And yet few CFD developers are aiming to provide the code interactivity and automated solution control that has been investigated in this research.

This research indicated that there were key mitigation and improvement strategies for handling the issues and problems encountered when running CFD based FFM scenarios. The research further demonstrated that the Experiment Engine concept was able to provide a complete supporting environment (which has been built and tested extensively within the *SMARTFIRE* FFM environment) and use methods similar to those deployed by Expert users to make the FFM set-up and meshing as robust as possible and this process has been optimized as far as possible. It was revealed that, due to the complexity and diversity of many "real world" fire scenarios, a suitably well-constructed set of short trials with different self-learning sets of set-up and meshing parameters can make significant contribution to the proper set-up and consequently make the software operation more autonomous, more robust, more reliable and able to offer friendly assistance to novice CFD users. The Experiment Engine concept will also continually monitor the simulation state/results and be able to make run-time adjustments to ensure that the solution is always converging and that the stability of the simulation is maintained automatically and that the whole solution process is conducted in an efficient manner. As a result, the Experiment Engine concept was demonstrated that it could control simulation runs automatically with no user interventions required after the problem was defined. The EE was then able to make automatic parameter changes as, and when, necessary (i.e. decisions include the auto-refinement of the mesh for regions where finer mesh resolution required; changes of time step size in responding to transient events, heat release rate changes and the handling of convergence and stability problems, etc.) In this way, the Experiment Engine concept maximizes the success rate of fire simulations, and makes sure that convergence and simulation stability are guaranteed (if a solution can be reached). The EE concept is fully automatic and has been designed to minimize extra computational costs (in the worse case scenario, maximum costs are limited to no more than a 20% overhead, by careful design of the Experiment Engine rule processing strategy). In fact, the Experiment Engine enabled simulations, for complex cases, demonstrated that it was able to achieve more than 40% performance gains in terms of simulation efficiency compared to non EE supported simulations.

- Investigation of a new technique that seamlessly link the pre-processor and the CFD solver to allow feed back from the simulation into the scenario set-up.

This enhancement of the existing FFM architecture is componentised and fully extensible.

CFD based FFM software is typically highly complex and traditionally has an architecture that is divided into, at least, two main components (namely a user interface in which to specify the problem / case set-up and a solver component which is dedicated to solving the complicated PDEs). The usual communication is one-way from case specification set-up environment forwards into the solver with the problem definition passed to the solver as formatted data (i.e. there was never any communication from the solver to the set-up component/user interface). In order to make the Experiment Engine concept work as proposed, a new component was designed and implemented into the existing FFM software architecture. The Experiment Engine was embedded in between the Case Specification Environment and the CFD Solver. In this way, the Experiment Engine is able to control all of the components involved during the simulation process. The Experiment Engine provides a two-way communication channel between the two major components. As a result, the communication can go in both directions using a message passing structure and protocols. This conceptual and physical componentization of the software architecture also allows rapid prototyping and implementation of new algorithms, solvers, supporting technologies and other ideas that may offer potential improvement. Hence the prototype research system has been designed to fully support further research in this (and any related) area.

Although this has not been investigated and therefore is not confirmed, the author strongly believes that the same architecture can be successfully applied to other general purpose or even other application specific CFD codes. The proposed Experiment Engine architecture and mode of operation should be sufficiently generic to suit other similar control problems.

- Investigation of performance based local adaptive mesh refinement methods for the mesh quality improvement of three dimensional multi-block structured hexahedral meshes.

Generating an appropriate and sufficiently high quality of mesh, for any given scenario, is the major task during the problem set-up phase of the simulation and is vital to the successful outcome of the simulation. The validity and efficiency of the Experiment Engine concept depends on having an effective and efficient mesh refinement method. There are many existing adaptive grid techniques. However, to be effective and efficient, the adaptive techniques have to be tailored to the particular type of the initial grid and how it was generated. The newly developed local block-wise mesh refinement method is based on the general principle of h-adaptive techniques and these have been implemented and optimised for use with the particular FFM automated structured meshing system to make ensure that it is suitable, efficient and effective. This investigation is a performance based approach to meshing since it is able to make 'run-time' adjustments to the mesh to suit the precise meshing requirements of the scenario, which obviously vary from problem to problem. The methodology that has been investigated and adopted, fully exploits the multi-block structure of the structured meshing system and also utilises the earlier work for characterising cases and the block-wise mesh justification algorithm.

- Investigation of new techniques for case recognition and characterisation of scenario specific meshing libraries.

The CFD software using parameterised automatic mesh generation techniques has been found to suffer (for many arbitrary scenarios) from inappropriate or only partially appropriate meshing parameters in the meshing library. Usually the situation is that the meshing library only provided suitable mesh parameters for a certain class of scenarios (i.e. it was developed for room based fire scenarios), but fails to do as well for the other classes of scenarios (e.g. "tunnel fire" scenarios are commonly inappropriately meshed when using "room fire" meshing rules). This research investigated knowledge elicitation from Expert users to improve the utilization of the meshing library by characterisation of real world fire cases into a range of typical categories and building a set of meshing libraries accordingly. Each meshing library consists of the complete set of mesh parameters and rules which are most appropriate for that class of scenario. In order to automatically select an appropriate meshing library, for a given scenario, during the automated mesh generation process, a dedicated case recognition algorithm was developed. This took a hybrid approach with a combination of backward rule base reasoning and case based reasoning [Kolodner-93]. The general format of the case recognition algorithm was derived from weighted nearest neighbour case retrieval algorithm from case based reasoning.

The techniques developed here have also benefited the adaptive mesh refinement method, described earlier, since they ensured a good quality for the initial mesh, thus reducing the requirement for further refinement. In this way, the efficiency of the adaptive mesh refinement procedure is significantly improved.

- Improved solution controls techniques

Previous simulation control techniques have only investigated control during the solution CFD processing. This means that qualitative assumptions have to be made about the appropriateness of the set-up and the mesh, and that a suitable convergence level is "guessed" solely according to the user's experience. These factors and assumptions have not actually been thoroughly tested thus there is a considerable degree of uncertainty which will quite probably affect the performance of the control techniques deployed. With the development of the Experiment Engine concept, it has been possible to take the control decisions in the context of the whole simulation process – including any trial runs conducted and the set-up, meshing. Hence an appropriate convergence level has been thoroughly tested and all the extra information obtained in the experiment stages are considered to be valuable in terms of implementing more efficient and effective control techniques. For instance, the evaluation of the solution state can take a more direct approach because the convergence level has already been firmly established with a high degree of confidence in the experiment stage. In this way, the prior experiments make convergence assessment easier and more effective. In addition, the parameterised changes are limited to a single parameter (i.e. the time step size) so many unnecessary tests are omitted based on reasonably well established expertise. This enables the design of control actions in more manageable ways to prevent any potential excessive search costs thus improving the efficiency of the control.

1.5 Background to this research

SMARTFIRE was chosen as the testing vehicle for this research. *SMARTFIRE* [Ewer-00] [Petridis-95] [Taylor-97a] was developed in an attempt to make the CFD based FFM techniques more accessible for novice users. *SMARTFIRE* has been developed an open architecture, interactive CFD code with integrated Knowledge Based System components. The *SMARTFIRE* system attempts to make Fire Field Modelling accessible to non-Experts users. *SMARTFIRE* makes many efforts to address issues of overall efficiency, reliability and to be user-friendly. In order to provide easy access to its solving power, *SMARTFIRE* includes sophisticated user interfaces to support scenario set-up, meshing, parameter configuration and to provide sophisticated run-time graphics capabilities and monitoring tools for comprehensive interactive monitoring and effective control of the simulation process.

The recently developed prototype Intelligent Control System [Janes-02] was intended to create an intelligent agent capable of emulating the Expert user's ability to effectively control CFD simulations and it was demonstrated that the ICS could provide similar benefits in terms of performance, overall reliability and result accuracy for relatively simple room based fire simulations. The Development of Groups Solvers Techniques [Ewer-99b] [Hurst-04] was able to reduce some of the high computational costs and memory overheads for CFD based fire simulation by focussing processing effort only on cells where continual updates are required. (For more details about Groups Solvers Techniques please to refer to the related section in chapter 3).

However, these new techniques are all highly dependant on having appropriate set-ups and a sufficiently high quality mesh for any given scenario. The question is how can the system ensure that an arbitrary modelling scenario can be effectively and efficiently meshed and that the specification and configuration is optimal (or at least near optimal) and appropriate. This question was the origin of this current research.

1.6 Structure of the thesis

Chapter 1. This chapter provides an overview of the research problems, enumerates the research questions posed and indicated how the objective of this research was achieved in a manageable way. It also presents the significance of the research. Major achievements and the contributions to knowledge have also been highlighted. The chapter also indicates the prior research that has been conducted in this area.

Chapter 2. This chapter provides a background to fire modelling techniques in general with special emphasis on fire field modelling. The operational skills needed to perform a successful fire field modelling simulations have been identified. The positive influence, of using knowledge-based problem set-up, on the simulation results has also been investigated and demonstrated.

Chapter 3. This chapter reviews the applicable literature relating to this research. It covers the various techniques available, mainly for improving the performance and ease of use CFD codes, and conveys to readers the knowledge and ideas that have been established and which are relevant to this research.

Chapter 4. In this chapter, attention is focussed on meshing issues, in order to seek any possible improvements to the current mesh generation and mesh quality control techniques. This investigation led to the discovery of a novel approach to structured meshing which enabled the meshing system to handle a wider range of real word scenarios more efficiently and effectively.

Chapter 5. This chapter describes the investigation and development of local block-wise mesh refinement methods for the structured meshing system. This meshing technique is based on adaptive finite element techniques commonly used in Finite Element Analysis, but adapted to the block structured target meshing system.

Chapter 6. This chapter describes the knowledge and technologies behind a new architecture for the control of a FFM solution process. This novel framework includes a new component called the Experiment Engine which is able to perform "trial and error" based experiment search for better set up, meshing and control parameters. The EE concept has been instantiated as a prototype within the *SMARTFIRE* environment. Initial test results of the EE are presented to show the validity of the local block-wise refinement regime and the effectiveness of the EE concept.

Chapter 7. In this chapter, additional investigations into extending the Experiment Engine concept are described. The Experiment Engine now emulates more of the Expert user's role in continually monitoring the solution process after providing an appropriate simulation set up, gained in the previous stages of the Experiment Engine process. This extension to the EE concept allows it to make run time adjustments of the solution control parameters. As a result, the fully automated Experiment Engine enabled simulation can respond to transitional events and heat release changes, and the EE enabled simulation process can obtain good solutions with almost no user interventions, therefore the EE offers complete user support for the whole simulation process.

Chapter 8. In this chapter, three computational examples are presented in order to show the effectiveness, efficiency and robustness of the prototype Experiment Engine by comparing the Experiment Engine enabled simulation to those from Expert users' applying manual control of a simulation and fully non-controlled "default" simulation.

Chapter 9. This chapter draws conclusions on how the investigations, conducted during this research, the supporting techniques researched in the early stages of this study and the prototype Experiment Engine concept benefit the wider FFM code users. The chapter then identifies the shortcomings and limitations of the current research and suggests avenues for further investigation.

2 Background to Fire Modelling

[Chapter Overview]

This chapter provides background knowledge about Fire Modelling in general. The advantages and disadvantages of using mathematical models over the traditional experimental investigations are briefly discussed. This is followed by an introduction to fire modelling techniques with special emphasis on fire field modelling, which takes a CFD based Fire Field Modelling (FFM) code as an example to illustrate the operational skills needed to perform a successful fire field modelling simulation and to help to identify how Expert knowledge could help with fire modelling simulations.

Finally the chapter presents a concrete simulation example to show the positive influence, of using knowledge-based problem set-up, on the simulation results.

2.1 Introduction

Prediction of the course of fire can be obtained by experimental investigations and mathematical modelling.

A full-scale experiment is ideal to give the most reliable information about a fire process. However it is expensive in terms of both resources and time. Sometimes it is even impossible to do so, because of high costs, difficulties in actual measurements and hazards that may be involved. The usual alternative is to perform experiments on reduced scale and then the resulting information must be extrapolated to full scale and general rules for doing this are often unavailable. Furthermore, the reduced scale experiment does not always have all the features of a full scale experiment. This further reduces the usefulness of the experimental investigations. Finally it should be kept in mind that, even where full-scale experiments are achievable, the results are not necessary entirely accurate. The measurement process is seldom free from errors.

Over the last two decades there has been a significant increase in our understanding of fire development and its influence on its surroundings [Prahl-75][Quitiere-84][Cox-95]. Understanding of fire changed from being empirical to being scientifically based. This change and more widespread access to powerful computers - at low cost - has resulted in ever more fire safety engineering solutions that have been found through the use of mathematical models [William-02]. The development of numerical methods [Patankar-80] further increases the possibility of using mathematical models, in Partial Differential Equation (PDE) form, which describe the physical and chemical processes predict many fire phenomena of practical interest.

2.2 Experimental investigation compared to Mathematical fire modelling

Using mathematical models to predict fire phenomenon has been in use for more than two decades. The use of fire modelling techniques has become widely used by designers and practitioners in many area of fire safety design [William-02].

There are many advantages that mathematical fire modelling can offer over a corresponding experimental investigation. First of all, the most important advantage of a mathematical fire modelling is its comparatively low cost. The cost of computer simulation is most likely many times lower than the cost of even a reduced scale of experiment. As ever, the increasing speed of CPUs and the availability of larger computer memory mean that computing has become cheaper and this cost will likely reduce even further. With the computer based simulation, it is very easy to try different set up and configurations with virtually no additional costs. Secondly, a computer solution of a problem gives detailed and complete information. It can provide the values of all relevant variables throughout the domain of interests (even in areas that were not thought to be important prior to the investigation but which were later deemed to be important due to the nature of the results). On the other hand, experiments cannot be expected to measure the distributions of all variables over entire domain. Finally, with computer simulations, it is generally possible to simulate most real scenarios, even those with extreme physical conditions. This contrasts with experimental investigations where it is not always possible, practical or feasible to perform the experiments or the required measurements. Furthermore, a simple computer simulation is ideal to study basic phenomena. In a basic phenomenon study, we need only focus on a few essential parameters and eliminate all irrelevant features. Then mathematical models become ideally suited to this kind of study, as a simple computer simulation with some simplified or ideal conditions such as one dimension and constant pressure can be easily set up. With the experimental method, such idealized conditions are not always easily obtained.

With the advantages mentioned above, it would seem that mathematical models outclass experimental investigations completely. However, it is very dangerous to think that 'old fashioned' experimental investigations can be abandoned. A computer simulation just works out the implications of an approximate mathematical model rather than computing/observing reality itself, so a computer simulation is useful only if it is adequately based on the validity of mathematical models and the numerical method used to solve the mathematical equations which describe the physical and chemical processes. Up until now, for some of the most complex phenomena, such as turbulent flows and non-Newtonian flows etc, adequate mathematical models have still not been completely and satisfactorily worked out. For simulations involving these complex phenomena, it is highly desirable to have sufficient experimental data to check the validity of the numerical simulations.

2.3 Computer based fire modelling

In general, the mathematical models consist of a series of equations which describe a certain physical process. If the equations are simple enough, they could (theoretically) be solved on a simple calculator. More commonly, the equations are not simple and there are a huge number of simultaneous equations that must be calculated. Consequently, a computer is required for their solution. Thus when reference is made to Fire Modelling, it actually means "computer based Fire Modelling" and this is normally realized as a computer program. The first computer program used to predict room fires was developed in the U.S. and was released in 1975 [Babrauskas-75], and other research in this field goes back even further. Zone modelling [Walton-02] and field modelling [Galea-89] are two fundamentally different modelling strategies used to simulate the effects of fires in enclosures.

2.3.1 Zone modelling

Zone models solve the conservation equations for distinct and relatively large zones (typically these zones are sub-region layers of the whole domain or compartment). The basic assumption of zone models is that for each zone in a compartment, the physical properties such as gas temperature and species concentrations are assumed to be uniform. These zones interact by exchanging mass and energy [Quintiere-89]. The most commonly used type of zone model (in single compartment fires) is the two-zone model which divides the room into two distinct control volumes. One is the upper hot layer descending from the ceiling, and the other is cooler, lower layer as shown in Figure 2-1 and these are based on the following assumptions:

- Room is small in both area and height
- Two layer approximation is valid
- Constant pressure in the room
- Venting is possible through a small opening

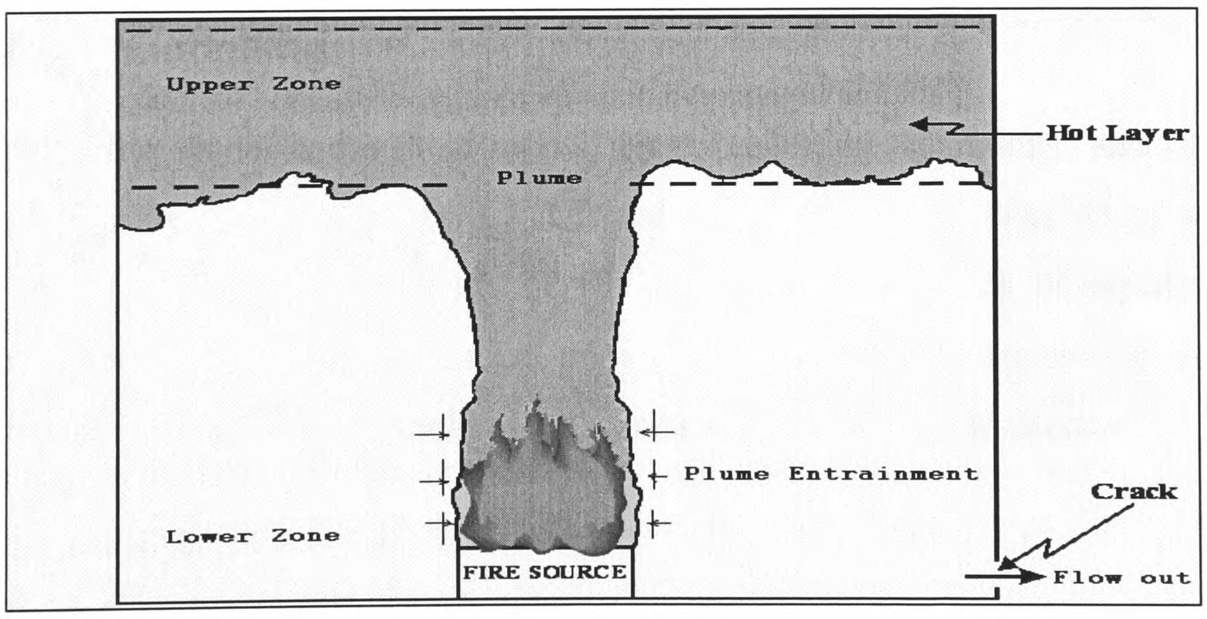


Figure 2-1: Illustration of the two zone model concept in an Enclosure

Figure 2-1 shows a schematic diagram how a compartment is modelled in a two-zone model. An example of an advanced Zone model computer program is called CFAST [CFAST-93] [CFAST-00] [CFAST-04]. CFAST is a multi-room compartment zone-based fire growth model, which was developed by the National Institute of Standards and Technology. The code has been widely used in the fire protection community to support alternate design approaches, post-fire investigations and as a research tool to better understand fire phenomena.

Zone modelling has proved to be a practical method for providing estimates of fire processes in enclosures and it usually provides very fast solutions comparing to that of using field models. However, for obvious reasons, the Zone model application does have the following limitations,

- Relies on a priori understanding of how fires behave.
- Maths used in zone models relies heavily on empiricism.
- Zone models are mainly developed for approximating values on gas layer temperature and location of smoke interface.
- Advanced zone-based fire models can handle multi-compartment, but there is still a limit on the number of compartments that can be modelled.

2.3.2 Field modelling

The newest fire model is the Field model. Field modelling technology, and use, has advanced rapidly in recent years. CFD based Fire Field model consists of several mathematical models [Galea-97]. Each of which comprises a series of equations to describe a certain physical or chemical process. These equations are solved with the highest available resolution to yield distributions of the variables of interest.

2.3.2.1 CFD codes and Field models

The CFD code and the Fire models are the two essential components in the Fire Field Modelling approach. The CFD code solves the complex PDEs describing the conservation of mass, momentum, enthalpy, and species etc. within the physical domain of interest. The Fire models (sometimes called sub-models) are used to describe the complex processes of combustion, turbulence and thermal radiation. CFD based field model presents a higher resolution approach than zone modelling, and the whole history of the fire evolution can be provided in all of the control volumes in the solution domain.

The numerical solution of fire and other related processes can begin when the laws governing these processes have been expressed in mathematical form, generally in terms of differential equations. The principles of Computational Fluid Dynamics (CFD) involved are:

- Mass is conserved
- Momentum is conserved
- Energy is conserved

The continuity, momentum and energy conservation equations are Navier-Stokes Equations, which are three dimensional and time dependent. Starting with N-S equation, and applying vector notation gives what is commonly known as the Convection-Diffusion (CD) equation. This represents the Navier-Stokes equations in a compact form as given below.

$$\boxed{\underbrace{\frac{\partial(\rho\phi)}{\partial t}}_{\text{Transient term}} + \underbrace{div\,(\rho u\phi)}_{\text{Convection term}} = \underbrace{div\,(\Gamma_\phi \nabla\phi)}_{\text{Diffusion term}} + \underbrace{S_\phi}_{\text{Source term}}} \qquad \textbf{(2.1)}$$

Where,

ϕ = dependent variable to be solved

u = the velocity of the fluid

S_ϕ = source /sink term

ρ = the density of the fluid

Γ_ϕ = the diffusion coefficient

Transient term represents the rate at which ϕ accumulates per unit volume.

Convection term is the accumulation of ϕ per unit volume due to the divergence in its convective flux field.

Diffusion term is the accumulation of ϕ per unit volume due to the divergence in its diffusive flux field.

Source term includes all the additional sources of ϕ per unit volume which is not covered by the previous terms.

All the essential equations can be constructed using this generalised form and the coefficient and source terms for each of the equations are listed in Appendix A (Table A-1). A detailed discussion about the mathematical formulation of the equations that govern the processes of interest is out of the scope of this dissertation. For the readers who are interested in the complete derivation of the required equations, and for the mathematical models for complex processes like turbulence, combustion and radiation should turn to recommended standard text books on the subject [Pantakar-80-6] [Cox-95-8] [Karlsson-00].

2.3.2.2 *Numerical methods*

Numerical methods are useful for solving fluid dynamics, heat and mass transfer problems, and other partial differential equations of mathematical physics when such problems cannot be handled by exact analysis techniques because of non-linearities, complex geometries and/or complicated boundary conditions. It includes the task of providing a set of algebraic equations derived from the differential equations through a process known as discretisation. The discretisation process is achieved by dividing the physical space defining the solution domain into vast collection of smaller sub-domains. Sub-domains described by collection of discrete grid points which carry one value for each of the variables. Interaction with neighbouring grid point variables is described by the algebraic equations. Essentially three different but related discretisation techniques commonly used are Finite Difference Method (FDM), Finite Element Method (FEM) and Finite Volume Method (FVM) (also known as the control volume formulation).

FDM describes the unknowns φ of the flow problem by means of point samples at node points of a grid of co-ordinate lines. Truncated Taylor series expansions are often used to generate finite difference approximations of derivatives of φ in terms of point samples of φ at each grid point and its immediate neighbours. Those derivatives appearing in the governing equations are replaced by finite differences yielding an algebraic equation for the values of φ at each grid point [Smith-85].

FEM use simple piecewise functions (e.g. linear or quadratic) to describe the local variations of unknown flow variables φ. The Navier-Stokes equations are precisely satisfied by the exact solution φ. If the piecewise approximating functions for φ are substituted into the equation it will not hold exactly and a residual is defined to measure the errors. The residuals are next minimized in some sense by multiplying them by a set of weighing functions and integrating. As a result a set of algebraic equations for the unknown coefficients of the approximating functions is obtained [Bathe-96].

For FVM, a formal integration of Navier-Stokes equations over all the control volumes of the solution domain is carried out. A variety of finite-difference-type approximations for the terms in the integrated equation representing flow processes such as convection, diffusion and sources are then applied. This converts the integral equations into a system of algebraic equations [Versteeg-95].

Each of these methods has its advantage depending on the nature of the problem to be solved. For example, FDE is very easy to learn and apply for the solution of PDEs encountered in the modelling of engineering problems for simple geometries (i.e. not very irregular). For problems involving irregular geometries in the solution domain, the FEM may have the flexibility, since the region near the boundary can readily be divided into subregions. There is no best method for all problems. The clear relationship between the numerical algorithm and the underlying physical conservation principle forms one of main attractions of the FVM and makes the concepts much simpler to understand by engineers than other methods. So far, the control volume method is used by the majority of main commercially available CFD codes.

2.3.2.3 FFM codes

The development of sophisticated radiation models, smoke models and combustion models, along with advances in, and lowering costs of computer power, means that fire field modelling has begun the transition from the confines of research laboratory to the desk of the fire safety engineer. Fire Field Modelling (FFM) techniques are now widely used by fire safety engineers and others for research, development and design tasks in industry. Many CFD based applications are now capable of or dedicated to solving modelling fire scenarios. Among them the most popular currently in use are Fluent, StarCD, Phoenics, FDS, CFX, JASMINE, and *SMARTFIRE* etc.

Fluent, StarCD, and Phoenics are the leading general purpose, commercial, CFD codes which are capable of solving steady, transient, laminar Newtonian and non-Newtonian flow problems. These CFD solvers possess a variety of heat transfer, radiation, turbulence and gaseous combustion models thus can also be applied to fire modelling.

Fire Dynamics Simulator (FDS) is a computational fluid dynamics program designed specifically for fire protection practitioners. FDS solves numerically using a form of the Navier-Stokes equations appropriate for low-speed, thermally-driven flow with an emphasis on smoke and heat transport from fires [Kevin-02].

CFX is a commercial model that was developed by AEA Technology. It can be used for assessing fire dynamics, fire structure issues, and fire suppression. CFX solves the Navier-Stokes equations in three dimensions to determine heat and flow fields in an enclosure [CFX-web].

JASMINE was developed as a fire specific code by the Fire Research Station in the United Kingdom in the early 1980s. Processes of convection, diffusion and entrainment are simulated by the Navier-Strokes equations [Cox-86].

SMARTFIRE [Ewer-00] [Petridis-95] [Taylor-97a] is an open architecture, interactive CFD code with integrated Knowledge Based System components that attempt to make fire field Modelling accessible to non-Experts users.

CFD based FFM techniques offer a powerful tool for fire safety design in buildings etc. However, it should be noted that field models are relatively complex to use and generally require operators with a good knowledge of both fire science and of numerical modelling. The accuracy of numerical simulation depends on many factors such as mesh/grid resolution, model specification and the appropriateness of the numerical methods being used.

2.4 Operations of FFM fire simulations

Like other Computational Fluid Dynamics applications, the field modelling software usually consists of a pre-processor, a solver and a post-processor. The pre-processor is used to define the actual problem (i.e. setting the boundary- and initial- conditions, selection of fire models to be used and generating mesh etc.) through a friendly and supportive Graphical User Interface (GUI). The solver uses the input data from the pre-processor to find a solution to the problem by using a selected numerical method. Finally, the resulting solutions are presented by the post-processor. The post-processing tools are usually able to display vector plots, 2D surface plots and provide view manipulations etc.

This section concentrates on the operation issues of the numerical simulation codes, in particular those within the *SMARTFIRE* environment, as this is the main platform for this research. In Figure 2-2 (drawn from an Expert user point of view), the whole *SMARTFIRE* simulation process is broken down to several stages and the difficulties of implementing each stage are highlighted. In addition, the expertise that is needed to overcome these encountered difficulties is also indicated. In this way, it is possible to identify when human expertise is needed and where it should be embedded into the system. The aims of this overview is to provide concrete information about the range of skills required to perform a successful simulation and to illustrate what ought to be done in order to assist with the reliable specification and successful simulation of Fire Field Modelling scenarios.

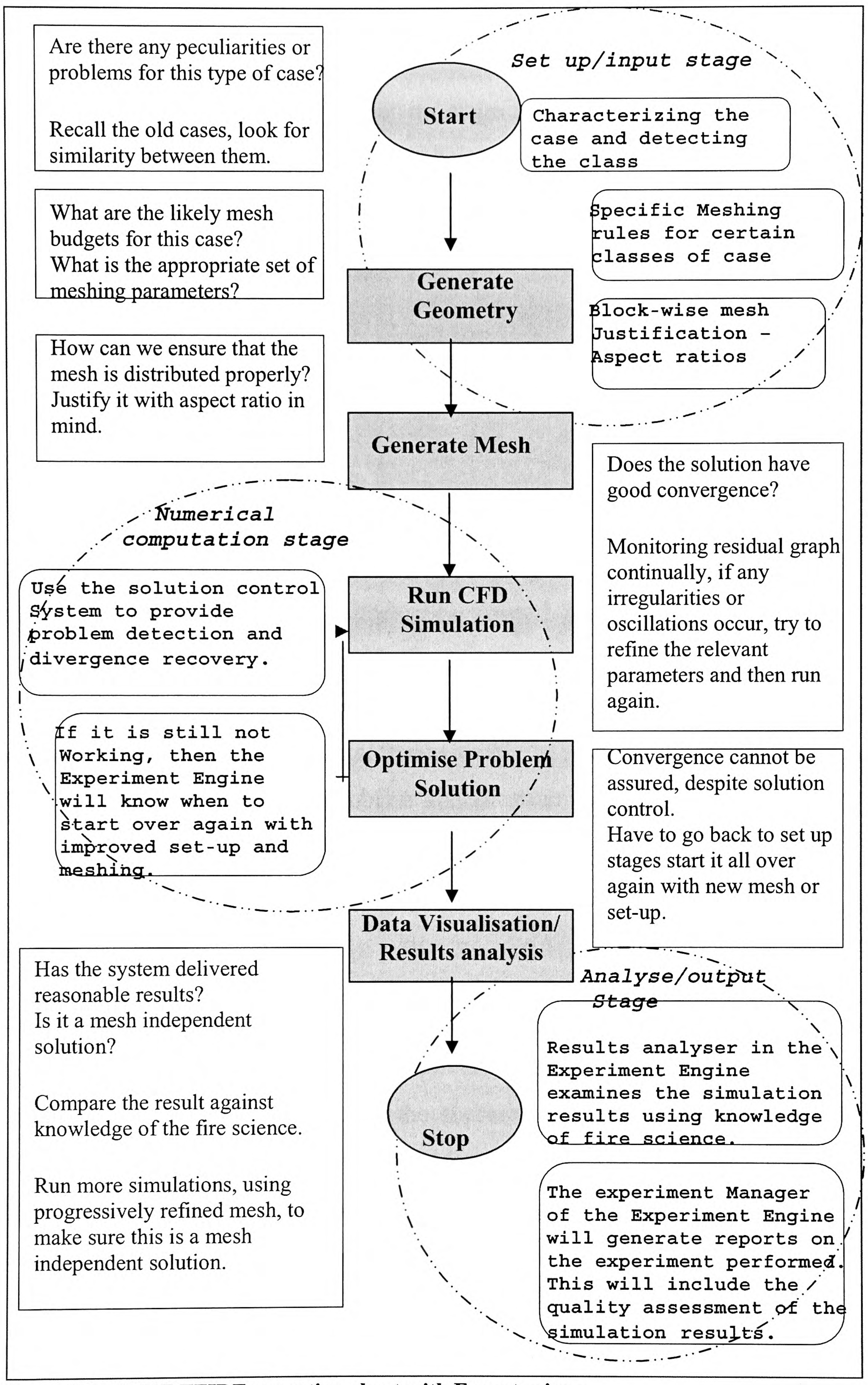

Are there any peculiarities or problems for this type of case?
Recall the old cases, look for similarity between them.
What are the likely mesh budgets for this case?
What is the appropriate set of meshing parameters?
How can we ensure that the mesh is distributed properly? Justify it with aspect ratio in mind.
Set up/input stage
Start
Characterizing the case and detecting the class
Specific Meshing rules for certain classes of case
Generate Geometry
Block-wise mesh Justification - Aspect ratios
Generate Mesh
Does the solution have good convergence?
Monitoring residual graph continually, if any irregularities or oscillations occur, try to refine the relevant parameters and then run again.
Numerical computation stage
Use the solution control System to provide problem detection and divergence recovery.
Run CFD Simulation
If it is still not Working, then the Experiment Engine will know when to start over again with improved set-up and meshing.
Optimise Problem Solution
Convergence cannot be assured, despite solution control.
Have to go back to set up stages start it all over again with new mesh or set-up.
Data Visualisation/ Results analysis
Has the system delivered reasonable results?
Is it a mesh independent solution?
Compare the result against knowledge of the fire science.
Run more simulations, using progressively refined mesh, to make sure this is a mesh independent solution.
Analyse/output Stage
Results analyser in the Experiment Engine examines the simulation results using knowledge of fire science.
Stop
The experiment Manager of the Experiment Engine will generate reports on the experiment performed. This will include the quality assessment of the simulation results.

Figure 2-2: SMARTFIRE operation chart with Experts views

Solid shapes ----- Normal operational flow chart of a CFD simulation.
Square boxes ----- Analysis followed by Expert user answers.
Rounded Square boxes ----- Embedded/Proposed human expertise.
Dotted Oval shapes ----- Operational stages.

2.4.1 Set up stage

The main tasks involved in the set up stage are problems specification and mesh generation.

2.4.1.1 Problem specification

A correct problem specification is sometimes highly dependant on how much the user knows about the problem. For some cases, the user has very detailed information about the values of important variables and the surrounding environment from a problem description, for example, heat release rate, fire starting and ending time, wall materials, ambient temperature, external pressure, material inside the domain of interest etc., that will help the user in specifying the problem. However, sometimes the user does not have all the information needed. It is in this kind of situation that Expert users are able to make an educated guess or recall from the past experiences to select appropriate models and assign a reasonable value for the unknowns. For novice users this task is not easy, so the software should be able to offer friendly assistance to them, in order to make the problem specification as appropriate and complete as possible.

2.4.1.2 Mesh generation

The second task, in this stage, is to generate an appropriate mesh before the information can be passed on to the solver. Mesh generation is a vital part for the whole simulation process, because the success of a numerical simulation is highly dependant on the quality of the mesh. Many of CFD systems provide the capability for automated mesh generation. But the automated mesh very often has a lower or higher mesh budget than the one actually required, because the meshing rules are often too general to suit each individual different type of simulation scenario. This problem is further compounded when the CFD code is general purpose and the meshing rules might be quite unsuited for fire modelling. Also the automated generation of non-uniform hexahedral meshes (e.g. a *SMARTFIRE* mesh) are very likely to suffer from cell aspect ratio problems. Justification of the mesh is usually required after the initial automated mesh is generated. However, even for Expert users, to justify the mesh manually are not always straightforward and usually take considerable time, especially for large complex 3D real world scenarios. Therefore the extension and enhancement of the meshing system to suit a variety of scenario types is very necessary and an automatic mesh justification method is highly desirable to ensure that a high quality mesh is obtained with minimal effort.

2.4.2 Numerical computation stage

The solution algorithm (within ***SMARTFIRE***) is iterative in nature. In a converged solution, the so-called residuals (solution error values), which are measures of the overall conservation of the flow properties, should be very small. Progress towards a converged solution can be greatly assisted by careful selection of appropriate values for the control parameters (e.g. linear relaxation parameters, solver iteration parameters and time step size etc). Unfortunately the actual values of these control parameters are difficult to determine in advance of the simulation since they are very much problem dependent. Generally speaking, to optimise the solution in terms of improving simulation speed and reliability requires considerable experience of the particular CFD code itself, which can only be acquired by extensive use of the software. If the simulation takes too long to converge or even diverges and/or consequently produces unsatisfactory data or meaningless results. In such cases, usually a simulation restart on the current time step is required with a different or improved set of control parameters. In some situations, the solution faults cannot be recovered from, and thus a complete re-run of the simulation is required using a reformulated solution strategy.

Until recently, research was directed towards a control technique called the Intelligent Control System (ICS) which was developed for the *SMARTFIRE* system. The ICS aimed to emulate an Expert's ability to control the CFD simulation and to provide similar benefits in terms of performance, overall reliability and result accuracy. However, its usefulness was limited because it relied largely on having a correct set-up and a high quality initial mesh. Furthermore, the ICS was only tested for the fairly simple room based scenarios. It was subsequently found that the ICS performed less favourably in more complex scenarios that included extensive geometries, large output fires and/or transient events.

2.4.3 Interpretation and analysis of the results stage

At the end of a simulation, the user must make a judgement as to whether the results are good enough. The most commonly used method, to assess the validity of the simulation results, is comparison with experimental data. Typically there will be no experimental data for the particular case in question, so the user has to rely on previous experience and knowledge of fire science. Sometimes it will be difficult to decide on the quality of the solution. Inappropriate mesh/grid design is among the most likely source of errors. There is no "gold standard" to estimate these errors. The mesh/grid dependence study is the usual approach to attempt to check the quality of the solution. By running more simulations on the same problem, each with a progressively more refined mesh, until certain key results do not change, then a mesh independent solution is said to be obtained. This cannot guarantee that the solution is correct but does give confidence that the solution is not dependent on the actual mesh that has been used.

Roache[Roache-98]has suggested a methodology called grid convergence index (GCI), which is based upon a grid refinement error estimator derived from the theory of generalized Richardson Extrapolation, to provide a consistent manner in reporting the results of grid convergence studies (also known as grid refinement study) and perhaps provide an error band on the grid convergence of the solution. The GCI can be computed using two levels of grid. However, three levels of refinement are recommended in order to accurately estimate the order of convergence and to check that the solutions are within the asymptotic range of convergence. The GCI is a measure of the percentage the computed value is away from the value of the asymptotic numerical value thus gives an error band on how far the solution is from the asymptotic value. This analysis indicates how much the solution would change with a further refinement of the grid. A small value of GCI indicates that the computation is within the asymptotic range.

The GCI on the fine grid is defined as $$\boxed{GCI_{fine} = \frac{F_s|\varepsilon|}{\left(r^p - 1\right)}} \quad \textbf{(2.2)}$$

Where, Fs is a factor of safety. The refinement may be spatial or in time. The factor of safety is recommended by Roache to be Fs=**3.0** for comparisons of two grids and Fs=**1.25** for comparisons over three or more grids. ε is the relative error between two levels of fine grids. r is the refinement ratio between two levels of grids. P is the order of the computational method.

However, because the amount of work involved (i.e. the solutions of at least two levels of fine grids are needed) for GCI analysis and let alone for independently generates grids, it is also problematic for non-uniformly refined grids and unstructured grids. So this method has not been widely adopted.

2.5 A simulation example

Here we present a simulation example on a simple room fire known as Steckler room fire [Steckler-82] to demonstrate that even small set-up details can affect the simulation results. The example also demonstrates the positive effects of applying a knowledge-based set-up. Three simulations (a, b, and c) are performed with three different fire flame heights/shapes set-ups, and then the subsequent results are compared with the experimental data. Flame characteristics and the Heskestad plume [SFPE-02] are briefly discussed as these are applied in simulation (b) and simulation (c).

2.5.1 Flame characteristics

- **Diffusion** Diffusion flames refer to the case where fuel and oxygen are initially separated and mix through the process of diffusion.

- **Buoyancy:** when a mass of hot gases is surrounded by colder gases the hotter and less dense mass will rise up ward due to the density difference, or rather, duo to buoyancy.

- **Turbulence**: Very small diffusion flames can be laminar, such as the flame on a candle. Larger diffusion flames are turbulent and tend to fluctuate with periodic oscillations and large eddies shedding at the flame edge.

- **Definition of mean flame height**: This is most conveniently done by averaging the visible flame height over time.

2.5.2 The Heskestad plume

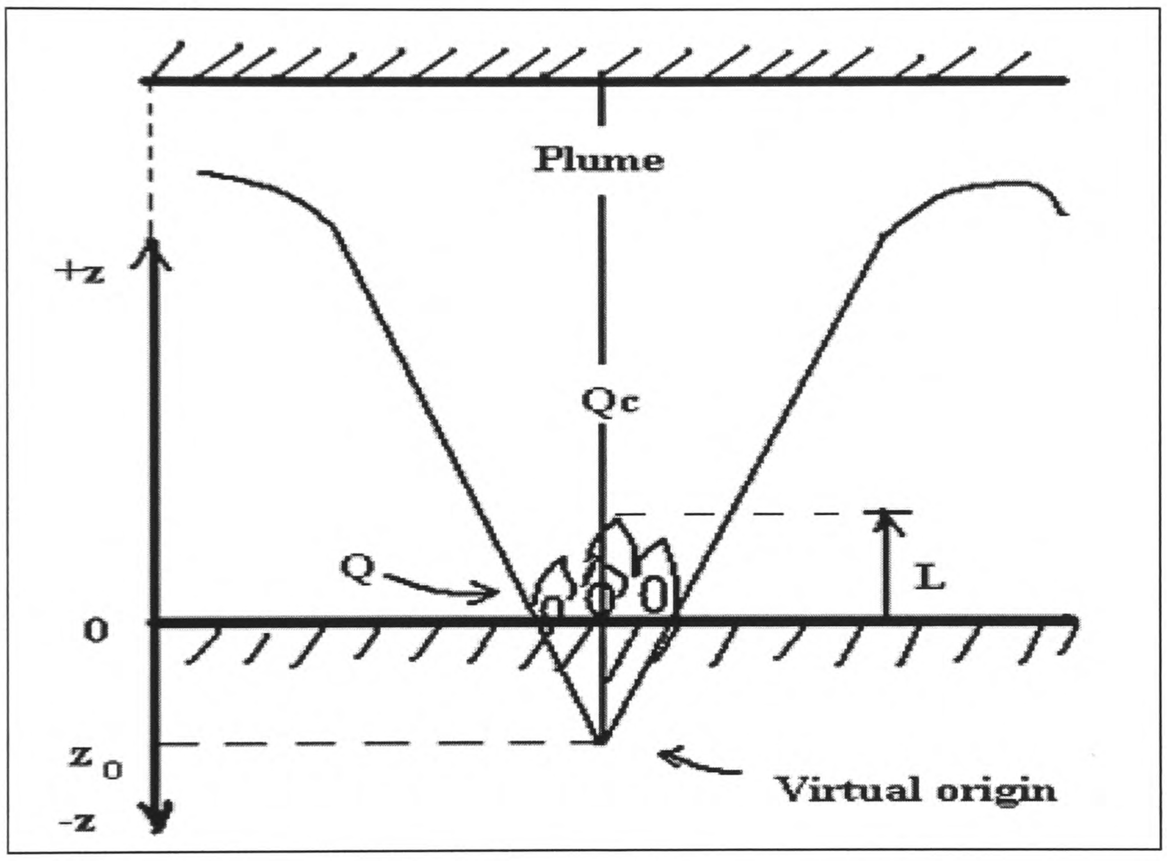


Figure 2-3: Illustration of the Heskestad plume

Z_0=0.083*power(Q,2/5)-1.02D **(2.3)**

Mean flame height: given by **L=0.235*power(Q,2/5)-1.02D** **(2.4)**

D = diameter of fuel bed

Energy release rate: The total energy release rate **Q** is used when calculating the mean flame height and position of virtual origin. The convective energy release rate **Q_C** is the part of energy release rate that causes buoyancy.

2.5.3 Description of the Experiment

Steckler [Steckler-82] et al. conducted a benchmark compartment fire experiment (known as Steckler room fires) in 1982. A series of 45 experiments were conducted to investigate fire induced flows in a compartment. As seen from Figure 2-4, the fire was a natural gas burner with various heat output at several possible locations in a 2.8 x 2.8 x 2.18 m compartment. A doorway/Window with various sizes was located in the centre of the front z-plane wall.

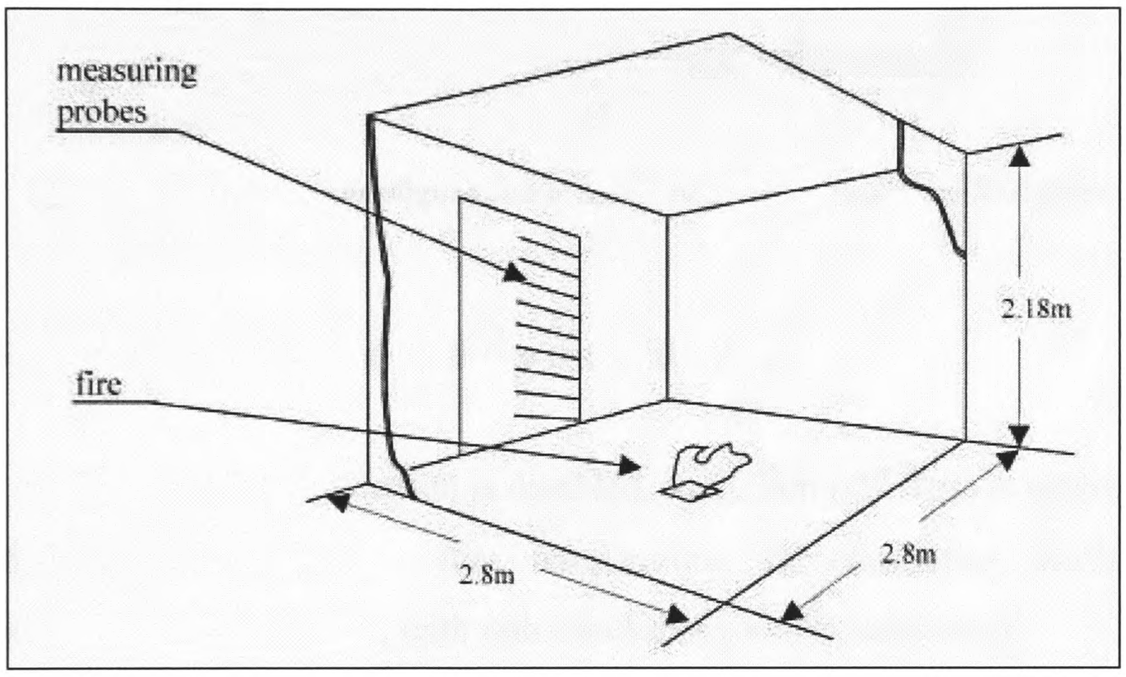


Figure 2-4: Sketch of the Steckler room fire experiments

The steckler test cases are one of the sets of validation cases for many fire models [Xue-01][Zhang-04] since, although far from perfect, they provide a considerable amount of experimental data.

In the test selected here, the door measured 0.74m wide and 1.83m high and the fire, which was centrally located on the floor (position A in Figure 2-5), was represented by a gas burner measuring 0.3m in diameter. The burner power was 62.9 kW. In the *SMARTFIRE* simulation it is assumed that the fire is a volumetric heat source with a constant heat release rate of 62.9 kW. The plan view of fire source, thermocouple and velocity probe locations is shown in Figure 2-5.

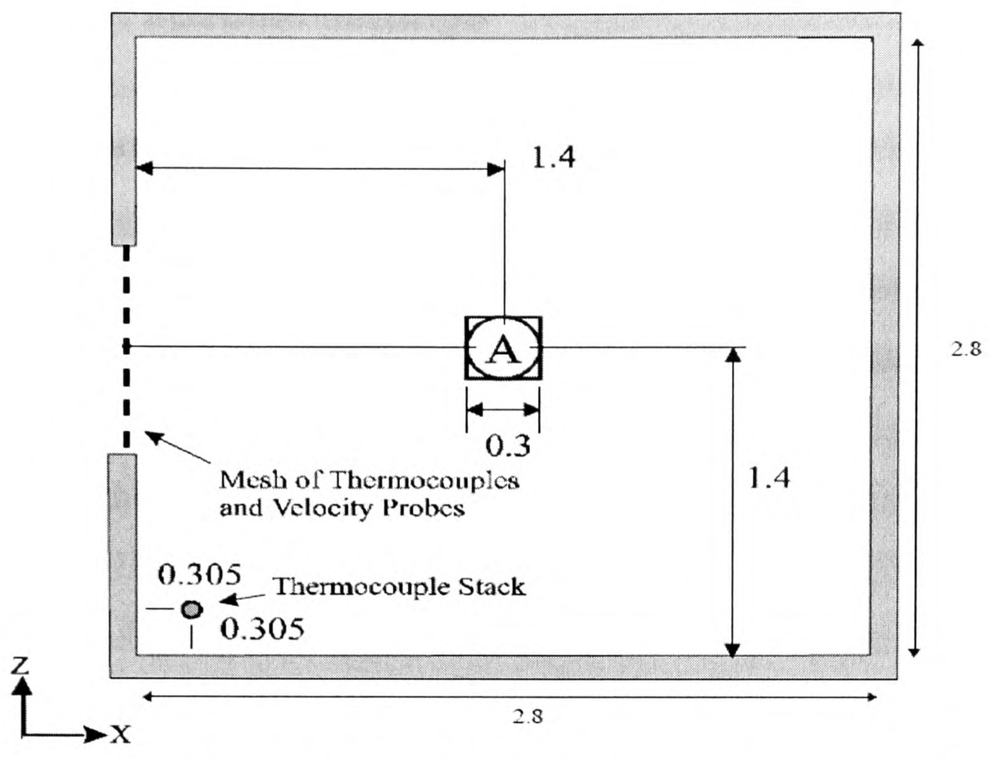


Figure 2-5: Geometric configuration of the Steckler experiment in X-Z plane

2.5.4 Model Setup and boundary conditions

- Flow model is used. Combustion is disabled, Radiation (six flux) is activated.
- Initial temperature was ambient temperature; the insulating walls of the compartment were modelled with non-conducting walls (adiabatic).
- The total heat source is the same as in experiment (62.9kW).
- Simulated using 200 time steps with time step size of 1 second, the maximum number of iteration was set to 50.
- Fine meshes were used to try to ensure mesh dependency was not a factor in the simulation results.
- A symmetry plane was used in the configuration, with only half of the scenario needing to be modelled.

2.5.5 Simulation specific settings

Three simulations are performed:

a. **Flame height was set up with 0.3m** most Expert users feel comfortable with. Using recommended mesh budget of 37840 (40*43*22) cells.

b. **Flame height was set up as 0.9m with a triangular volume shape** (achieved in *SMARTFIRE* by overlapping three smaller fires). The flame height value are given by adopting Heskestad Equation (L=0.235*Power (Q, 2/5)-1.02D). Using recommended mesh budget of 33600(50*28*24) cells.

c. **Flame height was again set up as 0.9m but with a rectanglar volume shape.** Using recommended mesh budget of 38266 (53*38*19) cells.

2.5.6 Simulation Results

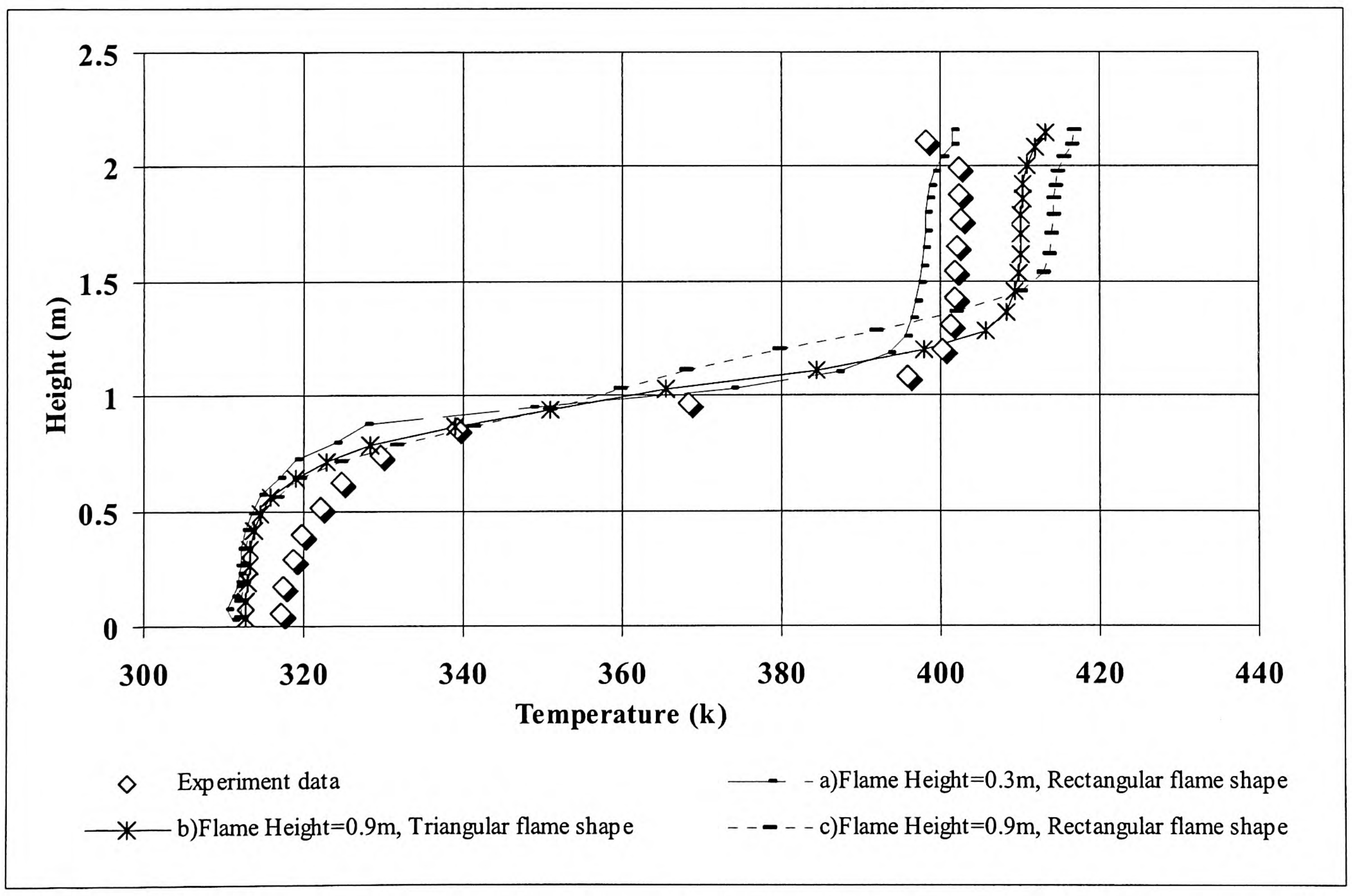


Figure 2-6: Vertical corner stack temperatures at 0.305 from the front wall and side.

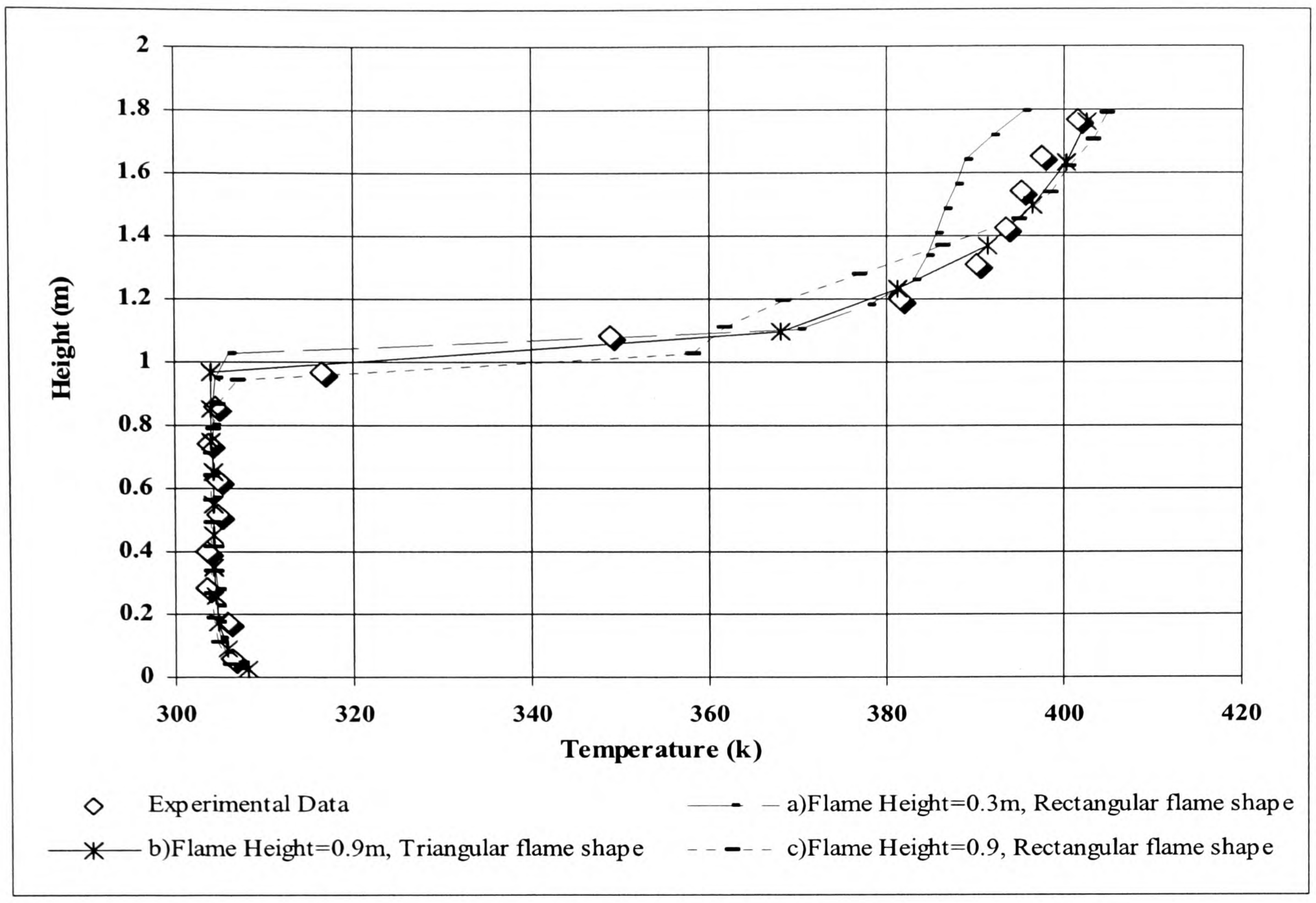


Figure 2-7: Vertical doorway temperature profile in the middle of the door.

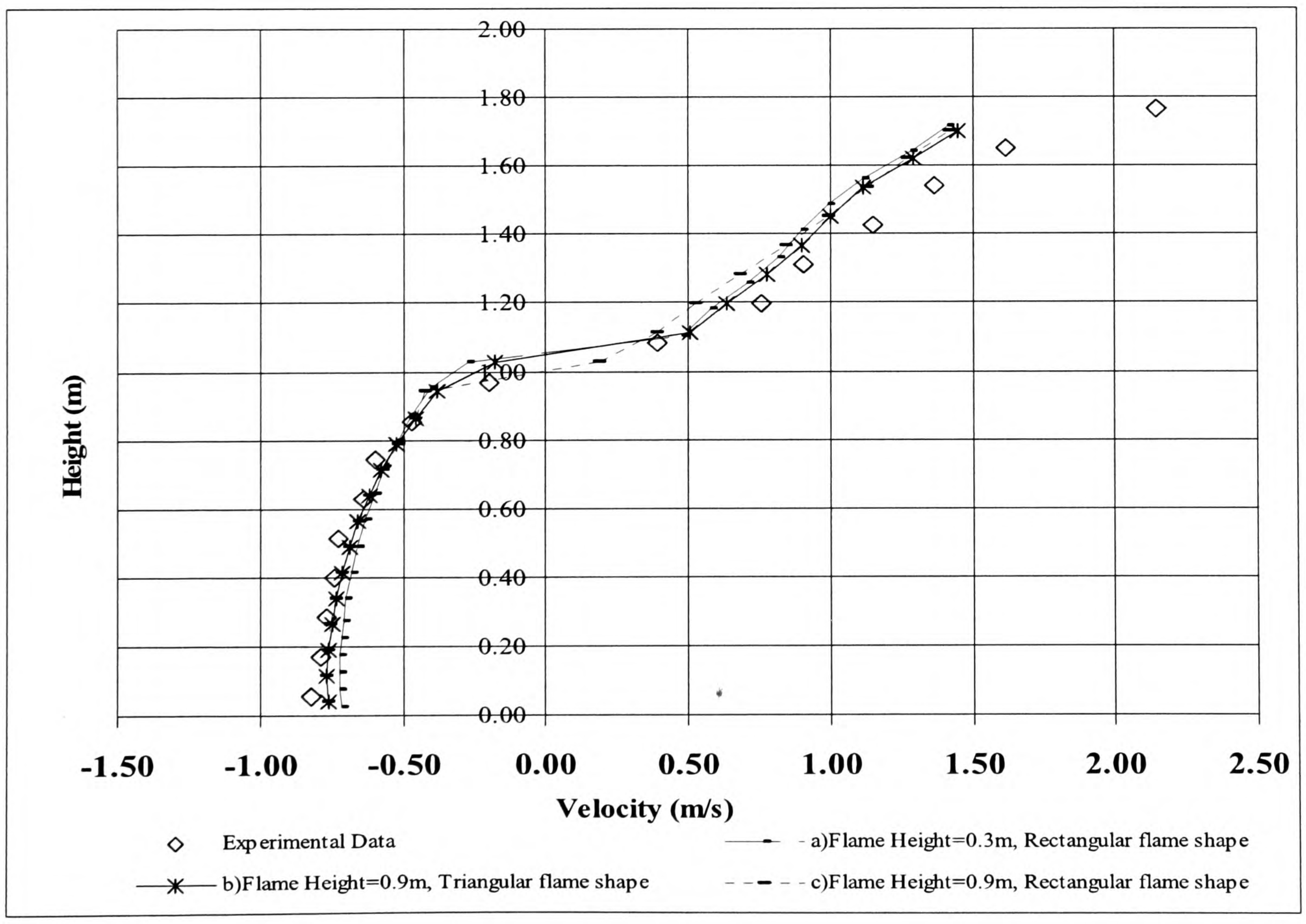


Figure 2-8: Horizontal velocity profile for a vertical stack in the middle of the door.

2.5.7 Interpretations of result

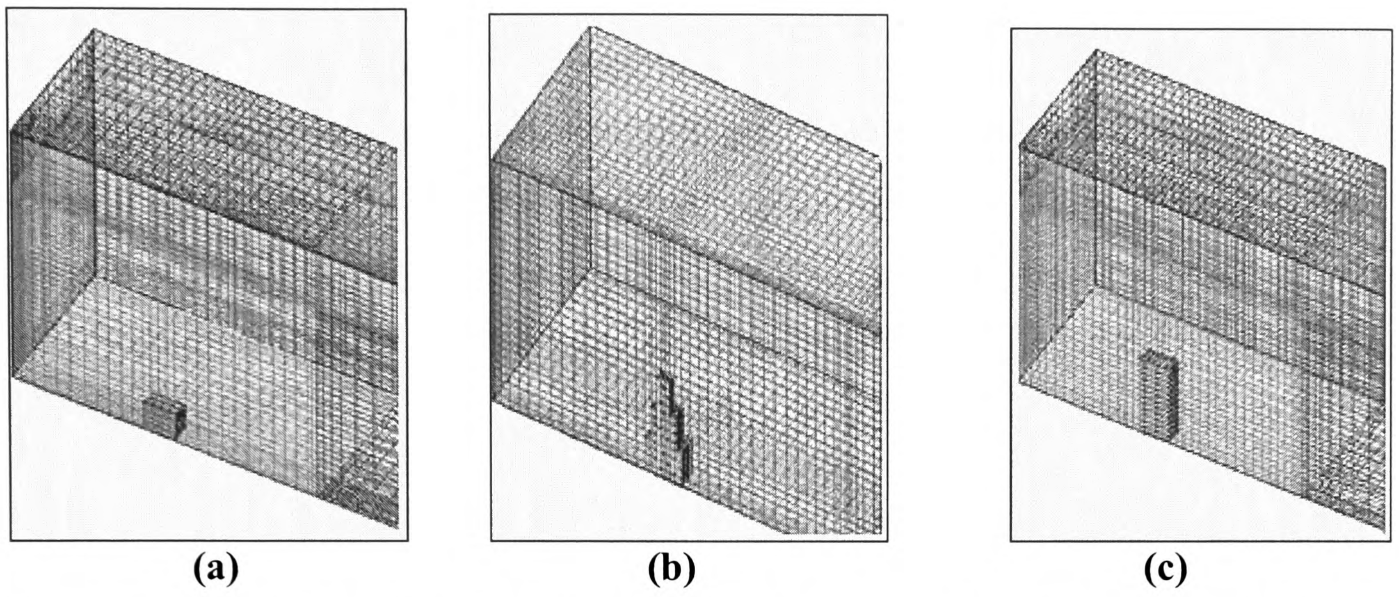


Figure 2-9: Different flame height/shapes set-up for three simulations in *SMARTFIRE*

- **The *simulation (b)* doorway profiles for temperature (Figure 2-7) and velocity (Figure 2-8) are in the best agreement with the experimental results in these three simulations.** Because simulation (b) takes the flame shape and flame height into account. So the heat distribution in the fire flame is more realistic.

- **As figure 2-6 shows that simulation (b) does not have a clear advantage over the other simulations in the prediction of the corner temperatures.** This can at least in part be attributed to the flame lean in the vented fire. If vent presents, hot combustion gases may escape and cool ambient air entrained into the compartment. These types of flow are extremely complex and transient. Several regimes of vent flow behaviour will be established. Predominant behaviour pre-flash over is bi-directional, hot gases exhausted out through top of vent and cool air entrained into the compartment through lower portion of vent.

None of these three simulations have fully reflected the actual fire flames in the experiment. E.g. we still have small amount of heat added at wrong place in the simulation (b). The conclusion that can be drawn from the above simulation example is that Heat distribution in the fire flames plays a very important role in the successfully modelling vented fire simulations. It is concluded that the Heskasted equation should be used with caution, especially with fire in vented room situations – because of possible flame lean, which would not be correctly modelled using a statically configured fire shape.

[Chapter Summary]

Mathematical models have been in use, to predict/study fire phenomena, for more than two decades. One of the most recent forms of fire modelling is the CFD based Fire Field model. The advancing power and lower costs of computers together with recent development of more sophisticated fire models means that there are many computer programs based on field modelling techniques, now in use. Fire Field Modelling (FFM) techniques are now widely used by fire safety engineers (amongst others) for research, development and design tasks in industry. The computer based fire simulation becomes not only an alternative to the experiment investigations, but also an essential, invaluable and sometimes the only feasible tool with which to study and investigate fire related phenomena. Much research has been directed into developing better sub-models to describe the complex processes of combustion, turbulence and thermal radiation etc. However, this chapter has discussed another equally important research issue by looking at the field modelling technique from the user perspective, where consideration is given to the operational issues of field modelling software and the user-skills required to perform a successful simulation. Detailed analysis of the stages of usage of a CFD based Fire Field Modelling Environment, have been presented and on each stage the required user assistance has been outlined. The final simulation example has shown the positive influence, of using knowledge-based problem set-up, on the simulation results. However, it also shows that it is very important to understand the constraints and limitations of such knowledge, before it can be applied to any system (especially those which rely heavily on empiricism).

3 Literature Review

[Chapter Overview]

The purpose of this chapter is to convey relevant prior knowledge and ideas, which have informed the current research, and to indicate the strengths and weaknesses of the previous work. The various techniques, mentioned in these discussions, are mainly for improving the performance and the ease of use of CFD codes. It should be noted that these observations and discussions are by no means exhaustive. However, it is hoped that this chapter will provide a brief, yet informative, insight into existing techniques for mesh generation, refinement / adaptation and automated solution control techniques for CFD simulations. Although none of these techniques have been adopted directly, some of the ideas explored in this chapter have been of considerable relevance to the investigation of the new techniques. As this thesis continues, the readers should be able to appreciate that the techniques that have been investigated are sufficiently generic for use by any CFD application, however the fine tuning during instantiation and testing of the new techniques have served the needs of fire modelling codes and, in particular, the *SMARTFIRE* application.

3.1 Introduction

In recent years, the development in computer technology, especially of graphical software brings much greater convenience to CFD users. More often, CFD users are no longer CFD Experts. However, to be successful at running a CFD simulation still requires that users have considerable operational skills, knowledge and experience in field of CFD modelling. First, users need to able to set up the case appropriately in terms of specifying geometry and to choose the right physical models to use (i.e. flow model, radiation model, and combustion model etc.) and to generate a sufficiently high quality mesh. In the set-up stage of CFD simulation, the most labour-intensive, time consuming and difficult part – as found by many users – is that the generation of an appropriate mesh which is suitable for the problem being simulated, adequate for the solution accuracy required and within the resource constraints in terms of time and available computer performance and/or memory. After set-up, the simulation has to be properly configured to maintain simulation stabilities, to ensure overall convergence and finally to obtain the results needed within the available time. The aim of this research is to investigate those techniques which can assist users to go through this whole simulation process successfully regardless of the level of a users' experience and to make the whole simulation process automatic with minimum manual intervention. Prior to this research, very few techniques were developed that were aiming to provide complete support for users during all of the key stages of the whole CFD simulation process. It should be noted that this research highlighted a number of existing techniques that offer partial user support to perform particular tasks during the simulation process. For example, there are techniques to help with automatic mesh generation and some of these techniques are able to refine the mesh by applying an adaptive procedure. Other techniques concentrate on providing functionality that gives automatic run time solver control. There are also interesting research techniques from the Artificial Intelligence (AI) domain that have been applied to certain aspects of controlling a CFD simulation. For the purposes of completeness and clarity, these various techniques will be discussed in turn. The first section will explore some existing ideas in mesh generation, adaptation and consequent mesh refinement so as to give a good understanding of the aims and methods behind many of the mesh related techniques. The second section details control techniques applied to CFD and other similar fields. This is followed by a brief review on specialised tools and Expert Systems developed for automation of CFD simulation in various application areas other than Fire Field Modelling. The final section in the chapter described techniques for improving performance of CFD codes and other research which is relevant to this work.

3.2 Techniques to assist CFD setup

Many CFD software systems were originally designed for users with a strong fluid dynamics background to maximize the CFD capacity, allowing the users to directly alter multiple set-up and control parameters. As the simulation software grew more advanced, the variety and exoticness of the parameters available for tuning, also increased. On one hand, this offers considerable interactivity and scope for control to the CFD Experts, so that they can apply their expertise to make the most of the CFD software. On the other hands, from CFD novice users' point of view, the modern CFD software becomes even more mysterious and difficult to use, since the novice users do not really understand the purpose, effect or range of values of some of the parameters, thus additional user assistance from the software is necessary. This section concentrates on those techniques which are used to assist with CFD set-ups and, in particular, the automatic mesh generation and refinement techniques.

3.2.1 Automated Mesh generation

Automatic mesh generation for complex two and three dimensional domains has recently become a topic of intensive research. Numerous research activities have been devoted to the development of automatic mesh generation techniques.
Automatic unstructured mesh generation algorithms have lent themselves readily to triangular and tetrahedral meshing [Owen-98]. The Delaunay method [Delaunay-34] [Lawson-77] [Watson-81], Quadtree/Octree methods [Yerry-84] [Shephard-91] and Advancing Front [Lohner-96] [Lo-91] are main categories of techniques that are used to automatically generate triangular elements (for two-dimensional cases) or tetrahedral elements (for three-dimensional cases) for a given set of nodes. These techniques have been well studied and utilized [Sibson-78] [Baker-89] [Weatherill-94] [George-91] [Vavasis] [Pirzadeh-93]. As a result, most literature and software make use of triangle or tetrahedral meshing. The automatic mesh generation of hexahedral elements for 3-D, on the other hand, is still an open problem, compared to tetrahedral elements, hexahedral elements are more suitable to strongly nonlinear problems and are more relevant to this research, therefore this

section concentrates only on those techniques which are capable of automatically generating a quadrilateral and/or hexahedral mesh.

3.2.1.1 A case based approach

A general-purpose automatic mesh generator should be capable of generating quality meshes without excessive user intervention. Such a system was developed by Taylor [Taylor-97b] for *SMARTFIRE*. The system takes geometrical input data from users and translates it into a reasonable computational mesh for CFD simulation. The system captured the qualitative reasoning of an Expert user of CFD, in the assessment of room geometries, in order to generate a suitable mesh for the given problem. This was implemented as a hybrid system comprising C++ code and rules based on case base reasoning techniques. A library of cases are stored in the system, the geometric input data from users formed as a source case then the library cases are matched against the source case to determine which was the most suitable. If a closest case was found then the mesh in the best-matching case is adopted. The system is also capable of dealing with new types of problem (previously un-encountered) by adapting and combining existing cases in the library. However the prototype system produced by Taylor's research could only specify and mesh, single room geometry with a single fire, therefore further research would be needed to extend this to multiple rooms and/or more complex buildings with multiple fire sources – before the idea could be developed as a usable tool for CFD simulation.

3.2.1.2 Intelligent Local Approach

Human experts, with knowledge of mesh generation, can generate quadrilateral and hexahedral meshes for an arbitrarily shaped domain using their superior capability for image recognition and qualitative judgement, if the number of elements to be generated is relatively small. To systematize the mesh generation processes performed by Experts, an automated mesh generation method called Intelligent Local Approach (ILA) [Yoshimura-99] , developed by Yoshimura et al can be used to control both the size and aspect ratio of quadrilateral cells in a two-dimensional plane. The ILA was implemented using an object-oriented technique. Mesh cells are created sequentially, considering local information on geometrical constraints and user's demand on quality of elements. A user can specify the cell size and the aspect ratio requirement. A field of priority of cell creation is also specified within the ILA. The method utilised a fuzzy logic knowledge processing to make the qualitative judgement of the quality of the mesh. The actual system, that was developed, is based on the following steps to generate a good quality mesh. First of all, the system makes an initial judgement for the whole distribution of cell sizes and a suitable aspect ratio then mesh cells are generated one by one starting from the boundary in either an inward or outward direction. Secondly, before one or more new mesh cells are to be generated, the system collects geometrical information from the local region to decide where and how the cells should be generated, and the system first "generates" virtual cells and the quality of "virtual cells" are judged by the fuzzy logic knowledge process. If the old cells are not compatible to the new cell, then they have to be regenerated. The same process repeats until the entire mesh has been generated. The most interesting part of the method is the fuzzy knowledge processing, which is used to determine the precise node location. It takes account of the cell shape, cell size, cell aspect ratio and the priority of cell creation. Although it was demonstrated that the developed system is only capable of generating good quality quadrilateral mesh, it is believed that the ILA could be straightforwardly extended to hexahedral elements in a three-dimensional solid.

3.2.1.3 A feature based Approach

Most of automatic hexahedral mesh generations were developed for a specific application area of CFD and many meshing algorithms (e.g. mapping [Cook-82] and sub-mapping [White-95]) have been developed and have proven to be very reliable on certain classes of geometry. It is not clear how these well established existing works can be utilised to automatically generate hexahedra mesh on an arbitrary geometry. Liu described a technique called feature based meshing methodology [Lu-01] to do just that. It utilises the existing mesh algorithms by partitioning the entire geometry into "meshable" pieces matched with appropriate meshing algorithms, the original geometry becomes meshable and can, consequently, achieve a higher mesh quality.

The method employs a feature recognition technique [Tautges-97] to guide the decomposition for hexahedral meshing. The term "feature" used here has different meanings from those used in CAD software, the "feature" here is defined as for example, geometries are regarded as the same feature as long as the same algorithm can be used to mesh them. There are four phases in this approach: "Feature Determination" to extract decomposition features, "Cutting Surfaces Generation" to form the cutting surfaces, "Volume Separation" to generate separate volumes, and "Meshing Algorithm Assignment" to match volumes decomposed with appropriate meshing algorithms.

Considering how an Expert would mesh a complex geometry. If the geometry is too complicated to mesh automatically, it is decomposed into smaller pieces, each of which is "meshable" using some known automatic meshing patterns. A great deal of meshing knowledge is employed by the Expert to help guide the decomposition process and this process is often heuristic-based, with few deterministic rules available. So first of all, Heuristic rules, as used by human experts, are identified and used in the decomposition process. After identifying possible features, cutting surfaces are then generated. These cutting surfaces are used to decompose the geometry, thereby slicing off the identified features. The process is applied recursively to the resulting pieces until no further decompositions are needed. On completion of the decomposition and volume separation processes, the appropriate meshing algorithms are employed.

The method described above has been implemented in the CUBIT mesh generation toolkit [CUBIT]. Although the system is not guaranteed to mesh a given geometry fully automatically, it can certainly reduce the amount of geometry the user has to mesh manually. The methodology proves to be effective and the results are encouraging.

A conclusion, which can be drawn from these works, is that techniques from AI domain have been very important ingredients to the methods used in the automatic generation of quadrilateral/hexahedral meshes.

3.2.2 Mesh refinement and adaptation techniques

The accuracy of numerical approximation of the governing equations of fluid dynamics (or any other non-linear governing equation) is limited by the number of cells in the mesh. In general, larger numbers of cells allow greater solution accuracy to be achieved. If the numerical approximation is consistent, then the solution error can be reduced to zero only as mesh cell size is reduced to zero. This is obviously impractical to do. Although the solution error cannot be eliminated completely, it can be reduced by increasing the number of cells in the mesh. However, computer resources and time would be largely wasted on the global uniform refinement on those sub-domains whose solutions do not require the maximum resolutions.

3.2.2.1 Overview of adaptive mesh refinement

Adaptive mesh refinement is a class of strategies that address the above problem. Adaptive mesh refinement (AMR) is a computational technique for improving the efficiency of numerical simulations of systems of partial differential equations. The basic idea is to refine those regions of the computational domain in which high mesh resolution is needed to resolve developing features, while leaving less interesting parts of the domain at lower mesh resolutions.

The use of AMR, in the numerical solution of partial differential equations, has become a popular technique for improving existing approximation schemes. AMR strategies have been developed for elliptic, parabolic and hyperbolic systems. The many approaches vary considerably, in both philosophy and implementation. The adaptive technique has to be tailored to the type of initial grid and how it was generated. The diversity of ways that grids are generated has effectively prevented the development of universal adaptive techniques. Since there are many grid generation techniques, there are also many adaptive grid techniques [McRae-01].

In general, there are three classifications of mesh adaptation. The first, h-refinement, adds extra nodes to an existing mesh to improve local grid resolution. A second techniques, p-refinement, employs higher order numerical schemes to improve local accuracy as well as to approximate troublesome derivatives. The third approach is r-refinement, which maintains the existing number of nodes globally but relocates them strategically and, more importantly, efficiently over the domain. These techniques may be used in combination. By far, h-refinement remains the most popular mesh adaptation technique and it is also very much applicable to this current research, therefore it will be given more consideration in these discussions.

3.2.2.2 Adaptive mesh refinement algorithms

In general, a typical adaptive mesh refinement algorithm usually involves the creation of an initial mesh, performing some analysis and error estimation, followed by improvement of the mesh based on the result of error analysis.

Research done by Arney et al. shows a good example of such algorithms. His work [Arney-90] presented a mesh-moving and local mesh refinement algorithm for time-dependent systems of partial differential equations in two dimensions that are discretized by either finite-difference or finite-element methods. A coarse base mesh of quadrilateral cells is moved by an algebraic mesh-movement function in order to follow and isolate spatially distinct phenomena. The local mesh-refinement methods recursively divides the time step and spatial cells of the moving base mesh in regions where error indicators are high until a prescribed tolerance is satisfied. The static mesh regeneration procedure is used to create a new base mesh when the existing one becomes too distorted. The error indicator, used in the mesh refinement, is based on estimates of the local discretization error obtained by Richardson extrapolation [Arney-87].

The mesh-moving procedure was based on an intuitive approach rather than more analytic approach. The idea is to move the mesh so as to roughly follow isolated non-uniformities and this generally reduces dispersive errors and allows the use of larger time steps while maintaining accuracy and stability. The refinement strategy consists of first calculating a preliminary solution on the base mesh for a base time step. An error indicator is used to locate regions where greater resolution is needed. Finer grids are adaptively created in these high-error regions by local bisection of the time step and the sides of the quadrilateral cells of the based grid, and the solution and error indicators are then computed on the finer grid. The procedure is recursive, thus, fine sub-grids may be further refined by adaptively creating even finer sub-grids.

The two essential elements of a mesh-generation or regeneration procedure are the determination of the number of nodes and their optimal location. A base mesh having too few nodes will result in excessive refinement, while one that has too many nodes will reduce efficiency. The mesh generation approach used here is to use the error indicators computed by a trial solution to determine an initial mesh that approximately equi-distributes the error indicators.

The results on three examples indicated that mesh moving can significantly reduce errors. The use of local refinement without mesh moving provided increased efficiency relative to uniform-mesh calculations, although not as dramatic as that found using mesh moving. However, it was also found that the mesh moving procedures perform better alone than with refinement in this approach.

3.2.2.3 Implementation of adaptive mesh refinement techniques

The literature contains many works regarding error estimates and refinement methods [Brackbill-82] [Benson-91] [Jacquotte-99] [Ainsworth-00] [Babuska-01] [Bangerth-03]. However, the main interest of this former research is that it shows that these methods can be used to fully automate and integrate these tools into a robust failsafe algorithm when dealing with a complex geometry. Until recently, some effort has been made to apply the existing techniques to fail-safe processes that can run on real world geometry have been reported in the literature.

3.2.2.3.1 An automatic solution process integrates with adaptive mesh refinement techniques

Tristano et al presented a framework for an automatic solution process integrates with adaptive mesh refinement techniques to obtain Finite Element Analysis (FEA) results with a desired accuracy [Tristano-03]. The processes communicate via data passed through COM Interfaces. The interfaces are implemented in several components (DLL's). The solver component solves the FE model. The driver component determines what elements should be refined, based on error norms and other information output by the solver. This component communicates between the solver and mesh refinement module and determines when the adaptive looping should stop. The robustness of the refinement process was demonstrated by a computational example where the initial mesh of the test case was intentionally set to the coarsest possible settings. However, the method is only capable of refining tetrahedral meshes.

3.2.2.3.2 Automatic Nested Refinement

Another technique called 'Automatic Nested Refinement' [Rajagopalan-03] was implemented in the framework of GridPro [GridPro] for addressing multi-scale problems in the context of multi-block structured grids. It enables conformal multi-block grids for multi-scale problems to be easily generated. The basic idea was to stack up elementary topological elements in a certain way so as to handle scale geometries in grids. A recursive methodology for stacking up a single element was preferred because such a technique could be programmed and hence provide for an easy and automated way to handle multi-scale problems. As a result, a program called "Nest" was developed which takes in a certain input and gives a nested topology as output. The user then loads in this nested topology (as a file), and links it to his existing topology in a few mouse clicks. Nest operates in an abstract topological level only, and does not need to know anything about the actual surfaces. The grid generation engine in GridPro takes care of topology conforming to the actual surfaces. To make the utility more accessible, a button was added in the GridPro GUI which leads to a dialog box which runs the program to create the nested topology. Automatic nested refinement offers automation and great flexibility in handling scale differences for grid generation.

3.2.2.3.3 Component-based adaptive mesh control procedure

In general, the automated adaptive mesh refinement procedure takes the following steps,

- Create a geometry-based problem definition,
- Create a initial mesh,
- Perform analysis and error estimation,

- Improve mesh based on the result of error analysis,
- Repeat the last two steps.

The first two steps are used to generate the mesh-based problem definition operated on by the analysis code (i.e. the solver). Because the only interaction between the component (to perform the first two steps) and the analysis code is the output from the component, the introduction of the automatic mesh generation is a relatively simple task. However, when the mesh is modified as the result of error analysis, this must be reflected in the mesh-based problem description used in the next analysis. The complexity of introducing extra levels of required interaction, between the component and the analysis code, has dramatically slowed the introduction of adaptive analysis methods in practice.

One approach, to address the mismatch between the needs of fixed mesh and adaptive mesh analysis procedures, is to alter the analysis code to directly interact with the adaptive analysis process. However, the modification of an existing fixed mesh code may require the introduction of entirely new data structures thus forcing an extensive rewrite of the code. For most well established codes this is considered as a prohibitive overhead. Shephard et al propose an alternative approach [Shephard-04] to address the problem which is to leave the analysis code unaltered and to use a set of interoperable information communication tools [Tstt] to control the flow of information between the set of components used for creating the problem definition, mesh generation, error estimation and correction indication, and the mesh improvement procedure. In order to enable the existing software components to share information properly, an additional component called "Field interface" is introduced. This provides complex functionality to obtain the solution information needed for error estimation and to support the transfer of solution fields as the mesh is adapted. The effectiveness of the resulting adaptive loops was demonstrated through two examples which are completely automated and give accurate results.

3.3 Automated control techniques for CFD

Unlike in the early years of CFD research, when most CFD users were mainly specialists in the subject, it has recently been observed that there has been a sharp increase in the number of specialists – from other subjects – who are turning to CFD techniques to support their own areas of expertise. But even for the most experienced CFD users, it is sometimes difficult to ensure solution convergence and to avoid divergence during a CFD simulation. Therefore many commercial CFD codes are trying to address this issue. It is believed that the CFD code itself should be able to choose appropriate control parameters and make adjustments as and when required to safeguard the convergence. This would reduce the need for user interactions/skill and improve the performance and reliability of CFD simulations. Ideally the users of CFD codes can then fully concentrate on the correct specification of the geometrical and physical input data and interpreting the physical results.

3.3.1 Automatic adjustment of the relaxation parameters

The CFD governing equations enable the errors associated with solved variables to be computed. In order to reduce these errors, a common solution is to use relaxations. Linear relaxations are more commonly used but sometimes a false time step relaxation has to be used because experience has shown that, for many problems, linear relaxation does not work for all variables. Inappropriate use of relaxations could seriously affect the simulation speed and it is difficult to decide on the actual values that should be used for the linear- and false time step- relaxations. It is believed that the most appropriate value is probably related to the speed at which events will take place in the flow. However, in the normal operation of the CFD simulation, these relaxation parameters are set before the flow has been calculated. A common mistake, often made by inexperienced users, is to choose excessively small values of false time step size and these tend to appear to give converged solutions because the resulting sweep-to-sweep changes are extremely small.

In the effort to address these problems, PHEONICS [Spalding-81] initially introduces a fairly simple algorithm called SARAH to calculate a false time step [Sarah]. SARAH stands for Self-Adjusting Relaxation AlgoritHm, The false time step is given by:

$$dt_f = \text{SARAH} \times (\text{internally calculated value}) \qquad \textbf{(3.1)}$$

Typical values of SARAH range from 0.00001 to 0.1, depending on the application. The value can be changed during run-time from the graphics monitor display.

However, SARAH does not always lead to improvement of the rate of convergence in all circumstances. It may 'over-shoot', making excessive changes in an effort to increase speed.

Most recently in 2004, PHEONICS included a new feature called CONWIZ (Convergence-Promoting Wizard)[spalding-04] in its latest release version 3.6. The aims of CONWIZ were designed to ensure the convergence of PHOENICS solutions, whenever the input data was sufficient and self-consistent. CONWIZ starts by making "guesses" about reference values of length, velocity, density and temperature, and it then deduces and sets some initial values of variables based on the reference values. It sets linear under-relaxation factors for all variables and maximum values to the increments per sweep for some variables. It can also make improvement of these setting during run time of the simulation, if required. It is acknowledged that No 'theory of convergence-promotion' has been invented in the development of CONWIZ. The idea behind the CONWIZ is by differentiating the momentum equations using SIMPLE-type CFD algorithms. Then the relationship between the false time step, mass, convection coefficients, etc. in respect with the rates of change of velocity with pressure difference at every point can be worked out, so CONWIZ can switch on and off the false time step and linear relaxation, and then further adjust the values of the relaxations if required according the flow conditions and convergence behavior.

It was demonstrated that CONWIZ can cope with difficult cases quite well, for example, the case with increased velocity etc. and is able to achieve convergence in various circumstances, while with the 'standard settings' could not do the same. However, CONWIZ does cause slower convergence comparing to long-ago-optimized standard settings and it does not always work, for example, Fires represented as fixed heat sources, regardless of flow conditions, have led to solutions which not even CONWIZ can persuade to settle down.

It was claimed that CONWIZ makes conservative rather than optimal choices and if CONWIZ has failed to achieve complete convergence, then it may indicate that there is some special instability promoter in the way in which the problem has been set up. Nevertheless, it was gratifying to observe that convergence has been achieved, with

"CONWIZ=T" as the only setting which the user has had to make to active CONWIZ in PHOENICS.

3.3.2 Intelligent Control System

Another CFD control technique was developed in the *SMARTFIRE* codes, it is similar to PHOENICS approach in a way, as it also makes run time relaxation changes, but it can also make time step size changes and look for performance gains via automatic control of the solver. Ewer attempted automatic control of fire simulations [Ewer-00]. Initially Ewer developed a simple rule based system to dynamically monitor and control the solution of a particular class of fire simulations according to the most recent local convergence behaviors. The system was demonstrated to be quite good for 2-Dimensional fire scenarios by speeding up the simulation by 50% in some cases, but failed to provide similar benefits for more complex 3-Dimension fire scenarios. It was believed that the failure was due to the lack of observing persistent trends in the solution behavior. Based on Ewer's work and his advice for further development of the control system, Janes started his research initially trying to make improvements on Ewer's work and developed more comprehensive assessment algorithms to assess the solution convergence behavior [Janes-04]. Unfortunately, even with the new contribution from Janes, the system was still not able to deliver the simulation time or stability improvement that were hope for. After analysis and consultation with CFD Experts, it was deemed that one source of the problems was the "kick effect" (when the relaxation parameters are modified during the time step then residual errors change abruptly) occurring immediately after the changes were applied, consequently Janes decided to move away from Ewer's control architecture and the Experts' knowledge was reassessed. The new control system developed by Janes called Intelligent Control System (ICS) was based on heuristic search techniques. The new system was able to make changes for false time step, linear relaxation and time step size automatically during the simulation and all control actions are applied between the time step, while the original Ewer's control system makes changes for false time step and linear relaxation only and it happened during the time step. The ICS system was designed to use a run time heuristic search to determine appropriate control parameters and heuristic evaluation function based on residual graphs is used for the assessment of both the simulation and the search result. Through the test cases, ICS was demonstrated to provide significant performance improvements, full convergence where it is possible and reliable automatic recovery from solution faults for simple room based fire scenarios. However, ICS was not able to control more complex cases and sometimes it moves away from the optimal control settings. These deficiencies, of the ICS, can be attributed to the fact that the designed control actions was based on the experiment of a particular class of fire simulations (i.e. simple room typed scenario) which are not always suitable for variety of fire scenarios and the system was trying to modify the three parameters at a time – in which the effect of these combination of changes was not always fully understood. However, the development of ICS system is still a valuable contribution. It tested the concept of applying control actions in between the time steps and was able to eliminate the kick effect suffered by Ewer's initial control system.

Ewer and Janes's earlier work served as a starting point for the development of the new control system in the Experiment Engine, therefore those systems are referenced with more details in later chapters.

3.4 Specialized tools/Expert Systems for automating CFD applications

Many application specific tools, templates and procedures exist, which streamline repetitive tasks, capture and standardise CFD process and can make CFD modelling more accessible and efficient for design engineers and analysts alike. Such application specific tools are capable of automating the process, such as mesh generation, boundary conditions definition, or post processing. These tools usually automate the CFD process based on a knowledge base which is the capture of Experts' knowledge in the field and formulation of "Best practice".

Slack etc [Slack-03] described such a system designed to automate the CFD process for cyclone and hydrocyclone simulations based on documented best practices. "Easy to use" was the primary objective for the system and it also offered a high degree of versatility, so that experienced users can add additional variables to the system. The development of a custom interface in the system increased the availability of CFD to the non-Expert users and the system automates the CFD process by executing a series of command files defining operations to be carried out by the CFD pre-process to generate mesh, set up and run the simulation. It also automatically generates user specific outputs or reports. However, there was no reported run-time control of the solver.

Morvan [Morvan-05] reported an interesting work on automating an atmospheric pollution case using a scripting language around and within the structure of the CFD command files. The code used was CFX-5 with PERL as a scripting 'language'. The testing simulation is based on a pre-defined generic CFD model, for which initial conditions, boundary conditions and source terms of atmospheric pollutant release are written automatically by the scripts using data recorded by measuring devices and stored on computers every half an hour as the simulation runs. When the correct amount of time has elapsed, the simulation pauses and the script updates the set-up using the newly recorded data. It then proceeds further, restarting from the appropriate result files. At each pause, a HTML report is also produced, which contains pictures of the area and summary tables. If a suitable criterion is defined in the post-treatment algorithm, such as a critical concentration for example, an alarm bell can be started, so that the technician knows the simulation has found a potential problem within the large domain that is monitored.

Another procedure was developed by Chyu et al [Chyu-95] to improve the turn-around time for computational fluid dynamics (CFD) simulations of an inlet-bleed problem involving oblique shock-wave/boundary-layer interactions on a flat plate with bleed into a plenum through one or more circular holes. This procedure is embodied in a preprocessor called AUTOMAT. With AUTOMAT, once data for the geometry and flow conditions have been specified (either interactively or via a name list), it will automatically generate a grid system and those for the initial and boundary conditions needed to perform a three-dimensional Navier-Stokes simulation of the prescribed inlet-bleed problem.

Furthermore, most – if not all – well known commercial CFD software have built in some degree of user assistance to facilitate using the software. For example, PHOENICS has tools to choose relaxation parameters automatically as mentioned above.

STAR-CD includes specialized tools to deliver application-specific methodology for pre-processing, meshing and post-processing. These are a range of Expert System tools called "ES-tools". Which are focused on capturing best practices within different applications, accuracy, the quickest turnaround time possible and on providing performance criteria. They are designed to get the solution needed to complex problems quickly and easily. For example, The "ES-ice" expert system automates setting up and running the sophisticated moving-mesh technology required for engine simulation. It automatically produces a parameterized meshed template that can be altered for specific engine configurations [STAR-CD-web].

Fluent offers CFD automation tools/templates that are specific to a particular customer's needs and workflow. Used by CFD Experts and non-Experts, tailor made templates are created to address a specific application and facilitate the CFD process by automating the model generation, set-up, solution, and post-processing. Each template consists of an easy-to-use, customized interface that can incorporate the customer's Expert knowledge and CFD best practices. For example, the glass-lined mixer template allows users to choose from several baffle and impeller types, set the size parameters associated with the mixing tank, and select the physical properties of the fluid [FLUENT-web].

ANSYS is famous for its multi-physics capabilities and provides engineers and analysts with a full range of engineering design analysis and optimization functions, including modelling and meshing, pre/post processing, graphics and design optimization features. Its star product is CFX and the software package comes with many Knowledge-based CAE tools, for example, ANSYS DesignSpace is a powerful, yet easy-to-use simulation software package that gives product designers and engineers the power to conceptualize, design and validate all their ideas right on their desktops. Using Knowledge-Based Automation™, it is a streamlined, user-friendly simulation tool [CFX-sol].

3.5 other interesting and/or relevant research

3.5.1 Group solvers

One of many interesting concepts implemented within *SMARTFIRE* is the development of group solvers techniques [Ewer-99b] [Hurst-04]. The idea behind it is to characterize the control volumes to allow control volume grouping based on processing requirements so as to reduce high computation costs normally associated with any large scale geometry scenario. Within the geometry of most large scale scenarios, it was noticed that there are obvious regions of the domain which will undergo intensive fire related activity and far field regions which appear to show little solution development over time, so the groupings can be identified from differences in the control volumes "activity" or residuals. Each cell membership group is assigned a specific solver control regime. The group solver concept utilizes the widely used SOR and JOR numerical solvers. The concept allows these solvers to be applied on a group by group basis with different settings depending on the required demands of the group. In this way computation effort can be targeted to groups of cells that require the most computational effort, rather than applying the same effort to every cell in the domain. In large scale problems, it was demonstrated that Group Solver techniques can reduce run times by at least 18%.

3.5.2 Parallel processing

Parallel processing distributes the computational task over a number of processors and therefore allows computational problems to be solved in a shorter time. Parallel processing techniques have been successfully applied to CFD calculations including fire field modeling [Galea-93] [Stuben-97]. In the past, the majority of this work has focused on the use of specialized hardware based around the UNIX operating systems. However, in recent years with the increasing power of PCs and the improved performance of Local Area Networks (LAN) it is the time where parallel processing can be usefully utilized in a typical office environment where many such PCs may be connected to a LAN. A prototype parallel version of the *SMARTFIRE* code has been designed to be used on a conventional LAN of PCs. The implementation of the parallel *SMARTFIRE* code was based on a systematic partitioning of the problem domain onto an arbitrary number of sub-domains. Each sub-domain is computed on a separate processor and runs its own copy of the *SMARTFIRE* code. At the boundary of the domain partitions each sub-domain needs to communicate with its neighboring sub-domain to exchange necessary data. This communication was implemented using the MPI parallel library. The parallel implementation of *SMARTFIRE* allows the efficient solution of large fire field modeling problems. Furthermore this is achieved on non-specialized PC equipment which may typically exist in many fire safety engineering offices. It was found that good speed up could be achieved on homogeneous PCs, for example, a problem composed of approximately 100,000 cells would run on a network of 12 PCs with a speed up of 9.3 over a single PC [Grandison-03].

3.5.3 *Influence of time step size on the convergence behavior and numerical accuracy for CFD*

The time step size and outer-loop iteration limit are also regarded as having significant influence on convergence speed and numerical accuracy, amongst the other important control parameters. It is generally accepted that using large time steps for a simulation can save CPU time for transient cases, but this comes at the cost of numerical accuracy. To some extent, this can be mitigated by increasing the outer-loop iteration limit. However, a larger outer-iteration limit will result in a longer CPU time if the limit is frequently reached during processing. There is trade off between time step size and outer-loop iteration limit, in terms of convergence speed and numerical accuracy. Liu [Liu-03] et al. conducted an interesting study on the influence of the time step size and outer-loop iteration limit for the integration of Reynolds Averaged Navier-Stokes equations, to improve the accuracy and efficiency of the numerical simulation of transient flow phenomena. They concluded that in the view of convergence, a small time step size is always favorable to a large one. A large time step size with more outer-loop iterations is not recommended because it tends to give poor convergence performance, while a smaller time step size with fewer out-loop iterations is recommended. This view is also echoed in Janes's ICS research.

[Chapter summary]

Many of the techniques that have been developed, and discussed in the chapter, show the response from both academia and from industry, to the fact that more and more people look for engineering solutions through the use of CFD techniques and that some of the users are not CFD Experts. The introduction to CFD always used to be via lengthy academic and/or industrial research, however, with today's fast pace, many new CFD users are coming to CFD with almost no CFD experience. Most modern CFD software systems have been developed with the aim of providing easier software use though the provision of extensive interaction. This means that Experts have great power to run the CFD by applying their knowledge. At the same time, because of the huge market expansion, vendors aim to make the CFD simulation process as automatic as possible, in order to attract even more users from other relevant industries.

However, even now, few – if any – CFD software systems are able to provide a complete supporting environment to users during the whole simulation process. Rather the existing systems offer techniques developed mainly targeted at specific phases of the simulation process and these tools and techniques have little interaction with each other and are not integrated to become a complete functional module that offers a complete supporting solution.

Nevertheless, the existing knowledge and techniques provide a solid foundation on which to help to find (at least partial) answers to the primary and subsidiary research questions of this research. This body of research also confirmed that the objective to provide a complete supporting solution, for using CFD based FFM codes, is both desirable and potentially achievable. Furthermore, much of this existing research demonstrates that successful supporting techniques make use of "best practice" to provide a working solution. The fact that this existing research is mostly disjointed and only capable of offering partial solutions or support, tends to indicate that there is considerable scope for integrations of many separate techniques into a complete supporting framework that is capable of enhancing the usability and robustness of CFD based FFM. These considerations mean that it will be necessary to analyse and investigate key aspects of appropriate "best practice" for FFM, in order to provide a technology capable of meeting the research goal.

4 A novel approach to support mesh generation and automatic mesh quality control

[Chapter Overview]

Chapter 3 discussed literature that is relevant to this research. In this chapter, attention is focussed on meshing, in order to understand any issues that affect solution quality/stability and to seek any possible improvements to the mesh generation and mesh quality control techniques. This investigation has led to the discovery of a novel approach to structured meshing which enabled the meshing system to handle a wider range of scenarios more efficiently and effectively. As a result of this meshing research, a number of key meshing technologies were investigated in depth, including case classification, case recognition and block-wise mesh justification. The validity and overall benefits provided by these novel methods are then discussed in the context of an existing structured meshing system. Whenever possible, this research has adhered to the main aims, of this Thesis, for providing ease-of-use and offering friendly assistance and automation to an existing meshing system.

4.1 Introduction

A Fire Field Modelling prediction, like other CFD based applications, works out the consequences of the underlying mathematical model, rather than those of an actual physical model. For FFM, the physical and chemical processes reduce to mathematical models mainly consisting of a set of differential equations and, where appropriate, empirical models. The governing differential equations are very complex and most of them cannot be solved by the methods of classical mathematics or their solutions contain infinite series, special functions, transcendental equations etc.

Fortunately, the differential equations can now be solved using numerical methods. These are dependent on the availability of powerful computers since the discretization and the iterative solution requires massive numbers of floating point calculations to be able to solve realistic problems. The basic idea behind the numerical approach to problem solving is that, for example, if we wish to obtain the temperature field in the domain of interest. It may be sufficient to know the values of temperature at discrete points of the domain. One possible method is to imagine a grid that fills the domain, and to seek the values of temperature at all of the grid points. In this sense, a numerical method is akin to a laboratory experiment, in which a set of instrument readings enables us to establish the distribution of the measured quantity in the domain under investigation [Pantakar-80].

4.2 Meshing / grid generation

The process of breaking up a physical domain into smaller sub-domains, known as meshing/grid-generation, facilitates the numerical solution of partial differential equations used to simulate physical systems. The clearest description of the process of meshing is given by Thompson, who states that: "Numerical grid generation can be thought of as a procedure for the orderly distribution of observers, or sampling stations, over a physical field in such a way that efficient communication among the observers is possible and that all physical phenomena on the entire continuous field may be represented with sufficient accuracy by this finite collection of observations. The structure of an intersecting net of families of coordinate lines allows the observers to be readily identified in relation to each other, and results in much more simple coding than would the use of a triangular structure or a random distribution of points. The grid generation system provides some influence of each observer on the others, so that if one moves to get into a better position for observation the solution, its neighbours will follow to some extent in order to maintain smooth coverage of the field. Another way to think of the grid is as the structure on which the numerical solution is built. As the design of the lightest structure requires consideration of the load distribution, so the most economical distribution of grid points requires that the grid be influenced by both the geometric configuration and by the physical solution being calculated. In any case, since resources are limited in any numerical solution, it is the function of the numerical grid generation to make the best use of the number of points that are available, and thus to make the grid points an active part of the numerical solution" [Thompson-85].

Generally speaking larger numbers of grids/cells lead to better solution accuracy that can be achieved. Both the accuracy of a solution and its cost in terms of necessary computer hardware and computation time are dependent on the fineness of the grid/mesh. Optimal meshes are often non-uniform: finer in areas where large variation in solution occur from point to point and coarser in regions with relatively little change. An ideal mesh generation strategy is really a way of finding a suitable compromise between the desired accuracy and the practical solution cost [Versteeg-95]. It is been said that over 50 percent of the time spent in industry, on a CFD project, is devoted to the definition of the domain geometry and the meshing/grid-generation task. That is why most existing CFD software environments make huge efforts to improve the user interface and interaction with users, in order to make problem set-up an easy task and to help the user to generate an appropriate mesh satisfying the accuracy requirement with the minimum possible number of mesh cells.

There are two broad categories of mesh generation methods, namely structured and un-structured grid methods. The techniques employed for un-structured meshing are largely dependent on the requirements and structure of the particular CFD code that will use them. This is, to some extent, also true for structured meshes. Structured meshes are still commonly used by many CFD applications; and in particular, often used by FFM software packages (e.g. FDS, Jasmine and *SMARTFIRE*). A discussion of unstructured mesh improvement would quickly become mesh system and CFD system specific. In order to investigate technologies that have a wide benefit to FFM, the next section focuses on structured mesh generation techniques and seeks to identify the shortcomings and limitations of such methods, in order to find suitable improvement strategies.

4.2.1 Structured mesh generation technologies

Structured mesh generation methods take their name from the fact that the mesh is laid out in a regular repeating pattern called a block. Structured meshing has a considerable advantage, over many unstructured mesh generation methods, since they allow the user a high degree of control of the mesh quality (i.e. unstructured meshing methods typically limit user involvement to the boundary surfaces of the mesh and the mesh system automatically fills the interior). Structured meshing systems utilize quadrilateral elements in 2D and cubic/hexahedral elements in 3D. Hexahedral and quadrilateral elements, which are very efficient at filling space, can support a moderately high amount stretching and some skewing before the solution will be significantly affected. In addition, because the user can interactively add the mesh cells, the arrangement of cells is most often flow-aligned, thereby yielding greater accuracy within the solver. Structured block flow solvers typically require the lowest amount of memory for a given mesh budget and tend to execute faster because they are often optimized for the structured layout of the mesh. Lastly, post processing of results – on a structured block grid – is typically much easier because the logical mesh planes make excellent reference points for examining the flow field and plotting/displaying the results.

It used to be the case that structured meshes could only have a single block. It is very difficult to model complex structures using just one block. It has been possible to employ optimizations to effectively "turn off"/remove portions of a single block. More recently, multi-block structured grid generation schemes have been developed which allow several blocks to be connected together to construct the whole simulation domain. With a multi-block structured mesh, it is possible to represent all of the different physical or geometrically important regions within different blocks. While multi-block grids give the user more freedom when constructing the mesh, the block connection requirements can be restricting and are often difficult to construct manually.

The major drawback of structured block meshing is the time and expertise required to lay out an optimal block structure for an entire model. Often this comes down to past user experience. The time required for manually generating a high quality multi-block mesh was usually measured in days, if not weeks_[Wyman-01].

A new hybrid structured meshing approach became very popular, and widely adopted by the existing structure meshing systems. The aim of the hybrid meshing approach is to speed up the structured mesh generation process by offering partial automation of the meshing process, whilst giving the user a high degree of control to adjust the meshing. Such a system typically employs both an automatic grid generator and a manual grid generator. Because automatic mesh generation techniques are usually rule based, the meshing system normally gives a quick solution to the initial mesh. It is easy to produce a mesh even for complex scenarios. However these can give some "poor" quality meshes due to the "general purpose" nature of the meshing rules – which were based on analysis of Expert users meshing relative simple scenarios. In such cases, a considerable amount of time is needed for manually editing the initial mesh, in order to achieve an adequate mesh quality. This is particular true for generating a high quality mesh for complex geometries. The final mesh is highly dependent on having a suitable automated initial mesh since the block structure obtained from the initial meshing (where the blocks represents the fixed edges of all of the geometric objects). These blocks can not be modified later by the manual mesh editor unless the user chooses to re-arrange and/or move objects within the scenario. Also because of the non-uniformity of the multi-block structure, automated multi-block structured mesh (typically using a dedicated cell distribution algorithm to arrange the cells in each individual block) is very likely to suffer from cell aspect ratio problems. This issue is discussed in some detail in later sections. Some structured meshing systems do attempt to handle this problem automatically by offering a refinement option, which increases the number of cells by a predefined degree on all the blocks of all directions across the whole domain of interest. This can be a massively costly operation because it can increase the overall mesh budget dramatically.

Because the initial automated mesh is so important to the final mesh quality, it was decided that it was worth looking at the possibility of extending and enhancing the ability of using the hybrid structured meshing approach to produce a suitable initial mesh for arbitrary scenarios. Furthermore, the cell aspect ratio problems are commonly encountered after the initial mesh has been generated, so it is also worth attempting to make justification of cell aspect ratios problem more efficient and effective in an automatic way.

4.3 Case Classification

The identification of the current structured mesh generation method and its limitations served as a starting point in the process of identifying areas for improvements. At this early stage of the research, effort was focused on knowledge acquisition and involved several informal interviews and case studies with Experts who were familiar with CFD based FFM system and had some knowledge and experience of other CFD software packages.

4.3.1 Knowledge acquisition

The process began with an initial interview using informal discussions, with more than one Expert, in an attempt to elicit a general overview of the knowledge domain. This informal discussion was dedicated to identify the "standard" procedures in which an Expert user constructs a suitable mesh for any given scenario. Most of the Expert users agreed that the first thing they can decide – almost immediately – is that when they have the problem description, the geometry of the domain of the interest and the accuracy requirements, they will have a rough idea about the overall cell budget that will be needed – or at least the sort of mesh density that will be required. This specific "knowledge/feel" is mainly coming from the experience of the similar scenario that an Expert has encountered before. However, detailed control parameters decisions are very much scenario dependant. For example, the acceptable cell aspect ratio is varying depending mainly on the type of geometry. For a geometry that is highly elongated, a relatively high cell aspect ratio may be acceptable especially if the fire region is relatively small comparing to the whole domain of interest, e.g. a long tunnel. It was found that Experts are very comfortable to make changes over various control parameters defining the overall mesh density and structures (blocks) to create a satisfactory initial mesh for arbitrary scenarios.

Following the initial interview, a set of case studies was conducted to see if different Experts would produce similar final meshing solutions for the cases that were being studied. The cases are all different in terms of the overall geometry, the complexity of the domain and with different heat release rates, etc. The final meshes are not exactly identical. However, there are some observed consistencies in terms of the overall density and mesh structures of the initial mesh created for a similar type of cases by different Experts. This was very encouraging for it indicates that it could be possible to characterize cases into typical scenarios. So a further formal set of interview questions was handed out to five CFD and FFM Experts in order to identify the typical categories and the characteristics of each category. The main criteria for selecting the Experts was that the Experts, involved in the questionnaire, must have extensive experience of running FFM CFD codes. The five Experts chosen for this knowledge acquisition process also represented a variety of different perspectives of fire field modelling knowledge. This was considered to be necessary because it was not desirable to have a single view of any of the knowledge elicitation questions and, indeed, it was necessary to try to understand the range of responses that could be obtained from various Experts working in the same field – but with different backgrounds. The backgrounds of the selected Experts included: a fire modelling Expert with broad knowledge of general fire models, an Expert in numerical solution algorithms, two Experts who are the main developers of the *SMARTFIRE* code whose knowledge covers different classes of fire field modelling scenario and a further Expert who has been heavily involved in the development of other CFD codes and in non fire related CFD modelling. The knowledge acquisition techniques used in this stage of the investigation are so called sorting techniques. Sorting techniques are a well-known method for capturing the way Experts compare and order concepts, and can lead to the revelation of knowledge about classes, properties and priorities [Rugg-92]. The Experts were given the task of sorting a large number of scenarios that were given to them. These scenarios demonstrated a variety of complexity in the geometry and had different fire source features mainly related to the heat release rate. The Experts were asked to sort these cases into potential categories according to their opinion about the acceptable solution accuracy (for each type of fire scenario) and the meshing requirements. Analysis of feedback from the various experts led to the organization and formation of the initial categories. Duplicated categories were deleted. The Experts were then asked to merge any similar categories. This process was repeated, until all the Experts agreed that the remaining categories were sufficiently distinctively from each other. Initial meshing rules – for each category – were formed from a simple combination of all of the Expert users' opinions about appropriate meshing for that particular category. Finally this captured knowledge was verified and tested by running the meshing system on selected scenarios from each category (using the initially proposed meshing rules for each of the categories) in the *SMARTFIRE* environment.

Analysis of these test responses resulted in the final refined expertise. This included the decisions about what the actual categories were and what the specific mesh requirements are for each of those categories. The following table (Table 4-1) summarises those case categories and each of categories' specific meshing requirements.

CATEGORIES	MESHING SOLUTIONS
TUNNEL	**long cells in direction of tunnel away from fire region – not appropriate to waste many cells away from the fire region since probably has uniform /layered flow**
TALL ATRIUM/BUILDING	**Ensure good mesh quality through floors and around holes. Also need consistent good quality mesh in the y-direction and above any top surface vents**
WAREHOUSE	**can use quite large cells beyond what would be used in a room case but finer near fires**
AIRPORT TERMINAL	**Need to have quite large cells. Watch for always inactive regions which can be turned off with group solvers**
STANDARD ROOM	**Probably default behaviour but watch for exceptional fire loads.**
SINGLE FLOOR OF BUILDING	**Watch for dead regions which can be deactivated with group solvers. Coarse mesh away from fire.**

Table 4-1: Identified typical fire scenarios and their specific meshing requirements

By naming each category, the Experts were also able to give information on the attributes and values they use to denote the properties of categories and important features which could identify the category. In order to prompt the Expert user to generate new attributes for each category, a so called "Three Card Trick" [Rugg-92] technique was used. This involves asking the Experts what is similar and different about three randomly chosen categories, i.e. in what way are two of them similar and different from the other. This is a powerful way of eliciting attributes and knowledge that are not immediately and easily articulated by the expert. This process was again repeated, until each of these categories was involved (at least once), against all other categories. This analysis was deemed to produce a reliable outcome. The resulting features for each category were thoroughly analyzed. In this way, a range of between five and eight of the key important features is identified for each of the categories. (E.g. n=[5,8] in equation (4.1) described later). This information was then used, to investigate the case recognition method described later in this chapter.

4.3.2 Building additional mesh libraries

In order to be able to generate an adequate quality of mesh from the automated structured meshing system, the meshing library should consist of category specific mesh control parameters that effect the following important aspects mesh quality,

- Overall mesh cell density (sets the overall "finesse" level of the mesh),
- Cell density weight over directions allows control of differently weighted distributions of cells in the x, y and z directions,
- Acceptable aspect ratios are used to indicate the tolerance of non-uniformity of cell edges and the reasonable length difference between the adjacent cells in all directions,
- Extra blocks required, for example, near wall layer, near fire layer and for the extend region, etc.,
- Other parameters to indicate the mesh distribution should take account of the physical significance (i.e. contents) of the blocks. For example, to set the minimum allowed number of cells according to block contents, i.e. there should be at least 3 mesh cells in the fire block in any one direction.

During the knowledge acquisition phase, knowledge of the domain was determined to fall into the following categories:

1) Geometric features including the overall dimension, layout and complexity of the objects in the domain,
2) Fire source features mainly related to the heat release rate,
3) Generally acceptable solution accuracy for a certain type of fire scenario.

These three categories provide the basis for detailed knowledge acquisition. The "Standard room" mesh library in the *SMARTFIRE* meshing system has been used as a "base" library due to the good understanding of this type of scenario. These categories provide the basic principles derived from the Expert's that can be used to construct a suitable mesh. Consequently the newly added mesh libraries are based on the original room based mesh library and have the parameters modified according to the difference between these three knowledge sources. The following table (Table 4-2) shows the example of "Warehouse" specific meshing rules which are derived from the Standard Room meshing rules with necessary modifications.

It should be noted that the actual values of the meshing parameters (as set in the mesh library) are based on an intuitive approach according to the Experts' "best practice" rather than any comprehensive analytical study. This means that the rules and parameters are almost certainly not perfect. However, it is representative of what the Expert is likely to do and is sufficient for purpose of verification and testing the identified technique, by implementing it as an extension to the existing meshing system.

```
MANUAL MESH SYSTEM - INPUT PARAMETERS

BEGIN PARAMETERS_SECTION

   BEGIN MESH_DENSITY_SECTION
      DEFAULT_CELL_DENSITY: 1.0
   END MESH_DENSITY_SECTION

   BEGIN EXTENDED_REGION_SECTION
      USE_CALCULATED_EXTENDED_REGION_LENGTH: ON
      MIN_LENGTH_OF_EXTENDED_REGION: 4
      MAX_LENGTH_OF_EXTENDED_REGION: 8
      DEFAULT_LENGTH_OF_EXTENDED_REGION: 5
      EXTENDED_REGION_CELL_DENSITY: 0.8
      USE_EXTENDED_REGION_SPLIT: OFF
      EXTENDED_REGION_SPLIT_FACTOR: 0.75
      MIN_NO_OF_CELLS_IN_EXT_REGION: 5
      MAX_NO_OF_CELLS_IN_EXT_REGION: 20
   END EXTENDED_REGION_SECTION

   BEGIN OBJECT_DENSITY_SECTION
      SIMPLE_FIRE_DENSITY: 0.5
      MULTISTAGE_FIRE_DENSITY: 0.5
      VENT_DENSITY: 0.8
      INLET_DENSITY: 0.2
      OUTLET_DENSITY: 0.2
      OBSTACLE_DENSITY: 0.5
      FAN_DENSITY: 0.5
   END OBJECT_DENSITY_SECTION

   BEGIN DIRECTION_RATIO_SECTION
      X_DENSITY_RATIO: 1
      Y_DENSITY_RATIO: 1.4
      Z_DENSITY_RATIO: 1
   END DIRECTION_RATIO_SECTION

   BEGIN EXTRA_LAYER_SECTION
      USE_NEAR_WALL_LAYER: OFF
      NEAR_WALL_LAYER_THICKNESS: 0.05
      NEAR_WALL_LAYER_PATCH_AREA: 0
      NEAR_WALL_LAYER_CELLS_FACTOR: 1
      USE_NEAR_OBSTACLE_LAYER: OFF
      NEAR_OBSTACLE_LAYER_THICKNESS: 0.05
      NEAR_OBSTACLE_LAYER_PATCH_AREA: 10
      NEAR_OBSTACLE_LAYER_CELLS_FACTOR: 1.2
      USE_NEAR_FIRE_LAYER: OFF
      NEAR_FIRE_LAYER_THICKNESS: 0.5
      NEAR_FIRE_LAYER_PATCH_AREA: 0
      NEAR_FIRE_LAYER_CELLS_FACTOR: 1.5
   END EXTRA_LAYER_SECTION

   BEGIN OTHER_PARAMETERS_SECTION
      MIN_LENGTH_OF_SMALLEST_VALID_BLOCK: 0.001
      MAX_LENGTH_OF_SMALLEST_VALID_BLOCK: 0.5
      DEFAULT_LENGTH_OF_SMALLEST_VALID_BLOCK: 0.05
      MIN_CELLS_SIMPLE_FIRE: 3
      MIN_CELLS_MULTISTAGE_FIRE: 3
      MIN_CELLS_VENT: 3
      MIN_CELLS_INLET: 3
      MIN_CELLS_OUTLET: 3
      MIN_CELLS_OBSTACLE: 1
      MIN_CELLS_WALL: 1
      MIN_CELLS_FAN: 1
      ACCEPTABLE_ASPECT_RATIO: 20
      ACCEPTABLE_LENGTH_RATIO: 33.3
   END OTHER_PARAMETERS_SECTION

END PARAMETERS_SECTION
```

Table 4-2: proposed warehouse mesh library

4.4 Case Recognition

Having been able to characterize the real world scenarios into typical categories, the next task is to investigate and devise a suitable case recognition algorithm to identify the best matching category for an arbitrary scenario. The identification is based on the input geometry and the problem specification. Subsequently, the categorization into a specific mesh library, is used to produce a suitable initial mesh for the given scenario.

4.4.1 Choice of reasoning techniques

Before devising the case recognition method, it is necessary to decide which of the available techniques to use. First of all, it should be noted that the proposed case recognition is different from the feature recognition techniques [Shah-01] used in Computer Aid Design (CAD) models, which involve the processing of a geometric model from a CAD system to find portions of the model matching the characteristics of interest for a given application. Feature recognition is a very complex process and very much a research area in its own right. Attempting to adopt the Geometric Feature Recognition techniques (from CAD modelling) would really miss the point, since the purpose of case recognition is only to utilise the case classification to find an appropriate set of meshing parameters for the given scenario and ultimately facilitate the automation of the case specified initial mesh technique through case classification. The need of feature recognition might be more relevant to future development of FFM applications, where it could load directly from the output of CAD models/building graphs into the GUI of the FFM system and find portions matching known features/objects and consequently be able to edit them. But for the purposes of case recognition, this methodology would be the wrong tool due to the complexity of the process.

For the purpose of this knowledge reasoning, the obvious choice seems to be to use rule based or case based reasoning techniques. Before choosing a particular approach, it will be useful to review these two reasoning techniques.

4.4.1.1 Rule Based Reasoning

Rule based reasoning is an important knowledge representation paradigm. It uses facts and rules to represent knowledge and a rule is the basic unit of knowledge. A rule is a structure which has an "if" component and a "then" component. For example, "if" - the customer closes the account, "then" - delete the customer from the database. After the word "if" is a condition, premise or antecedent and this represents some pattern which may be observed. After the word "then" is an action, conclusion or consequent and this represents some conclusion that can be draw, or some action that should be taken.

The rules here are not the same as an "if ... then ...else" as would be seen in procedural languages and are not really representative of a logic system. In a conventional program, the if...then... structure is an integral part of the code, and represents a point where the execution can branch in one of two (or more) directions. In a rule based reasoning system, the if...then... rules are gathered together in a rule base, and the controlling part of the system has some way of choosing a rule from this knowledge base which is appropriate to the current circumstances, and then using it as and when it is needed[Giarratano-04].

The rule based reasoning is commonly used to draw conclusions based on available evidence (i.e. forward reasoning). It can also be used for backward reasoning or pattern-matching system. The principle advantage of production rules is notational convenience. For example, the rules are distinct units of knowledge and a natural format for expressing knowledge. Rule based systems have a disadvantage when new knowledge – intended to fix a particular problem – may introduce unwanted contradictions therefore require addition rules.

4.4.1.2 Case Based Reasoning

Case based reasoning is a relatively recent problem solving technique and was introduced as an alternative to rule based reasoning. Case-based reasoning (CBR) solves new problems by adapting previously successful solutions to similar problems. CBR is attracting attention because first of all, CBR does not require an explicit domain model and so elicitation becomes a task of gathering case histories. Secondly, implementation is reduced to identifying significant features that describe a case, an easier task than creating an explicit model. Thirdly, by applying database techniques largely volumes of information can be managed. Finally, CBR systems can learn by acquiring new knowledge as cases, thus making maintenance easier [Watson-97].

In Case based reasoning, the emphasis is on the case, not the rule. CBR works with a set of past cases, a case-base. CBR seeks to determine a "source case" relevant to a given "target case". The main difference between rule based reasoning and case based reasoning is that rules are retrieved that match the input exactly in RBR while cases are retrieved that match the input partially in CBR. The major advantage of CBR over RBR is that the domain does not need to be completely understood and cases are easier to obtain than general knowledge.

CBR has been applied to classification tasks, e.g. to determine the type of an organism from observed attributes and to use case histories to determine whether or not a particular cancer treatment is used.

4.4.1.3 Reasoning techniques chosen for case recognition

Given that there are advantages and disadvantages to both of these approaches; it might seem difficult to choose one approach over the other. In both RBR and CBR, managing the interaction of multiple sources of guidance is crucial. In CBR, different cases can suggest conflicting conclusions; in RBR, several rules might conflict. Interplay of rules and cases is unavoidable. A first glance at the purpose of the case recognition system seems that it is perfect for backward RBR since it only has very limited possible conclusions and the general knowledge about those case categories is well understood. However, the rules for identification of a particular category are likely to be very complex and the simpler rules introduce a number of logical contradictions. For example if “Maximum domain length” is greater than 9 meters, then it could be a tunnel case, a tall/atrium building case or a warehouse case. There is no unique solution. Furthermore, even if we managed to produce a set of rules, it must also be the most appropriate. Otherwise, it is very likely the given scenario could not match any category due to the fact that the inputs have to match exactly to the selected rules in rule base. Given the constraints of RBR implementation enumerated earlier, it seems that CBR is the preferred implementation strategy. Considering a CBR implementation, first, to gather a relevant case is again straight forward, all that is needed is to construct a 'perfect' case for each of those categories, and this will certainly produce a matching result at the end due to the fact that the input is allowed to partially match cases in the case base. But to identify common determining features, that would describe all cases, proved to be very difficult, because the significant features in one category may become irrelevant to other categories. For example, a vent at each end may indicate that there is a good chance that the given scenario is tunnel case. But this fact might not matter in the identification of other categories. This difficulty means that it is necessary to reconsider the two approaches and to think about some kind of combination of them. After all, a case can almost always be viewed as a compact representation of a set of rules and in practice, the separation of CBR from other forms of reasoning is imperfect. A hybrid approach, to solve this problem, is to use the backward RBR as a starting point to identify the significant features to the individual category, and these features need not apply to all of the categories. This is due to the fact that, in RBR, the rules will fire only if the conditions match the current state. For example, if it were decided that the current case is a Tunnel case, then only those features and rules relevant to Tunnels would be applied. In this particular case, six important features were identified during the knowledge acquisition process (described earlier in section 4.3.1) and the significant features of a Tunnel case are shown in Table 4-3 below. Then it is possible to apply the backward RBR process. First of all, the system collects all categories (goals in RBR) to be tried and these are put on the stack. Secondly, categories are selected (one goal in RBR) one at time and a determination is made of all rules capable of satisfying the current category (goal). Finally, it is necessary to decide how well the relevant rules are met to the goal. This is where the CBR techniques are used, because it provides a means of evaluating solutions when no apparent algorithmic method is available for the evaluation. It provides a means of efficient solution generation, and evaluation is based on the best case available.

	TUNNEL	
Region Geometric parameters	***Relevance***	***Value***
Maximum length	**JAE(8)**	**>10m**
Minimum Area	**JAE(8)**	**>15m^2**
Maximum aspect ratio	**JAE(10)**	**>3:1**
Number of compartments	**JAE(5)**	**1**
Other comments	**JAE(7)**	**vent at each end**
		inlet and / or outlet also possible
Meshing solutions	**long cells in direction of tunnel away from fire region**	

Table 4-3: significant features of a Tunnel case

4.4.2 Implementation of case recognition

The general format of the case recognition algorithm was derived from weighted nearest neighbour retrieve algorithm in case based reasoning as follows,

$$\boxed{Similarity\ (T,S) = \sum_{i=1}^{n} (W_i \times sim(T_i,S_i)) / \sum_{i=1}^{n} W_i} \qquad (4.1)$$

In CBR where:

T : target case,

S : source case,

n : number of important features,

W: weight show the significance of the feature denote as a float number. Sim denotes the difference between the important feature i in the target case and in the source case. Similarity denotes the overall distance between the target case and the source case in n-dimensional space.

In our algorithm, in contrast, "Sim" denotes the similarity of the important feature i in target case and in the source case and it can only take the value of 0 or 1. That means that the target case either has the feature or it does not. The "Similarity" denotes the likeness between the target case and source in respect of n important features and it expresses this as a float number (i.e. an overall score). The scores are computed for the give scenario (target case) against all the 'perfect' cases in the case base which represent each of the categories. The scenario is then recognised as the category for which the similarity evaluation obtained highest score, provided that the score is high enough. A score of 60 percent or greater is a positive indicator, otherwise it is regarded as unrecognisable. Because this is not an exact "science", sometimes a scenario will be "unrecognisable", in such a situation, the safety measure is to apply the default "safe" meshing library.

4.4.3 Validation of case recognition

The case recognition algorithm has been validated by verification from the Expert regarding the correctness of the recognition. Different levels complexity of geometries was given to the new version of the *SMARTFIRE* GUI system which implemented the case recognition. It is observed to give reasonably good results. Two snapshots are provided here for demonstration purpose.

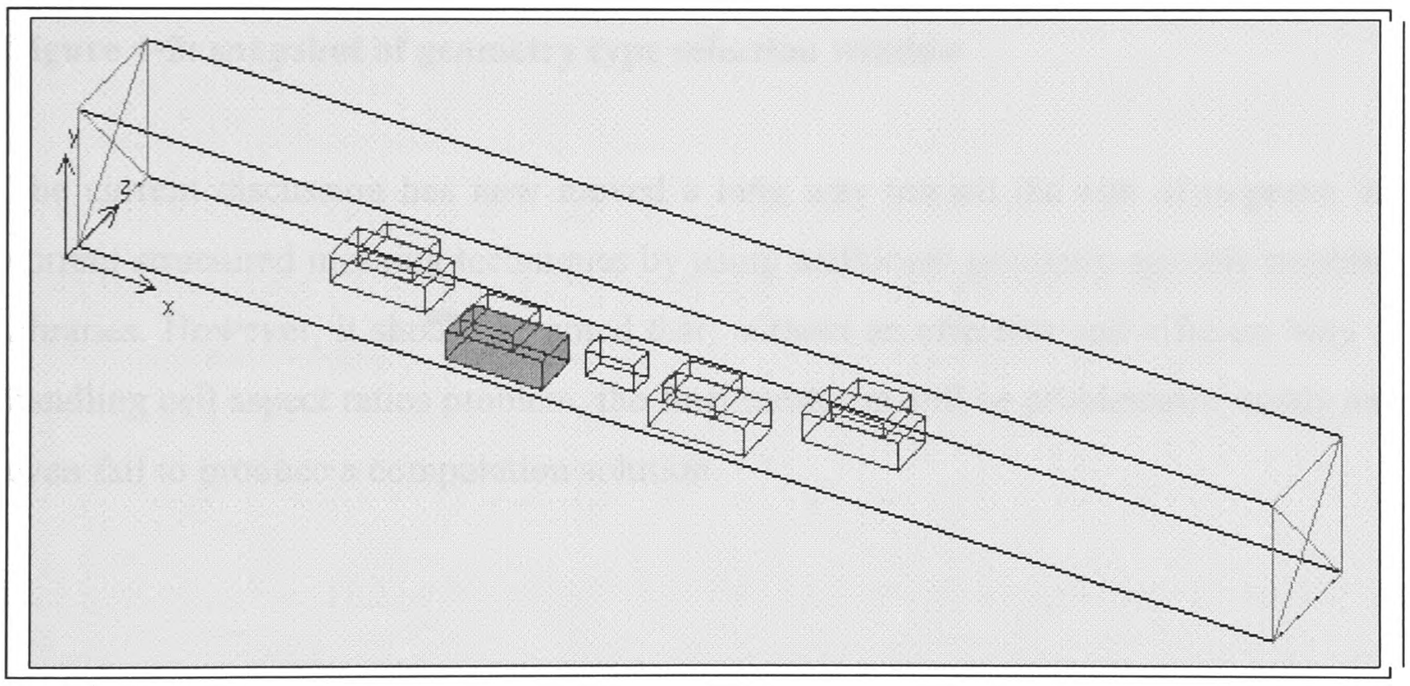
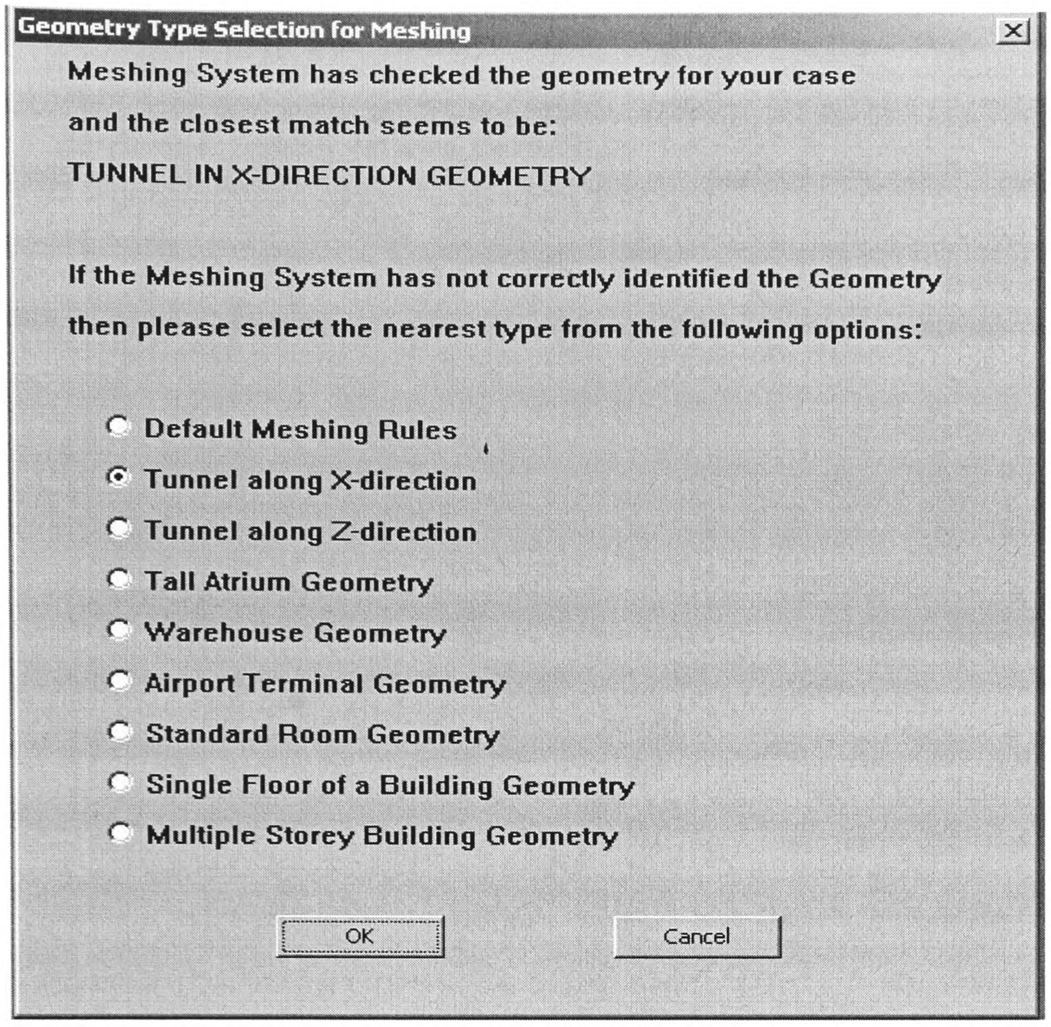


Figure 4-1: a geometry type example

Figure 4-1 above shows a geometry configuration example and Figure 4-2 shows the recognition results produced by the case recognition algorithm below.


Geometry Type Selection for Meshing

Meshing System has checked the geometry for your case
and the closest match seems to be:
TUNNEL IN X-DIRECTION GEOMETRY

If the Meshing System has not correctly identified the Geometry
then please select the nearest type from the following options:

- Default Meshing Rules
- [✓ Tunnel along X-direction]
- Tunnel along Z-direction
- Tall Atrium Geometry
- Warehouse Geometry
- Airport Terminal Geometry
- Standard Room Geometry
- Single Floor of a Building Geometry
- Multiple Storey Building Geometry

Figure 4-2: snapshot of geometry type selection window

The current discussion has now moved a long way toward the aim of improve the current structured meshing techniques by using additional geometry-specific meshing libraries. However, it should be noted that, without an effective and efficient way of handling cell aspect ratios problem, the final mesh can still be problematic, costly and even fail to produce a computation solution.

4.5 block-wise mesh justification

To construct a mesh of suitable quality (in order to get an acceptable solution) it is necessary to ensure that all cells pass particular tests. Cell internal aspect ratio test and cell-neighbor aspect ratio test are among those tests. Experts' rules suggest that, for a good quality mesh, both cell internal aspect ratio and cell neighbor aspect ratio should be within a certain range. Unbounded aspect ratios can lead to instability problems.

4.5.1 Cell aspect ratio

Cell internal aspect ratio is defined as the ratio of the largest edge to the smallest edge in a cell/control volume. In Figure 4-3, the cell internal aspect ratio is given by X/Z.

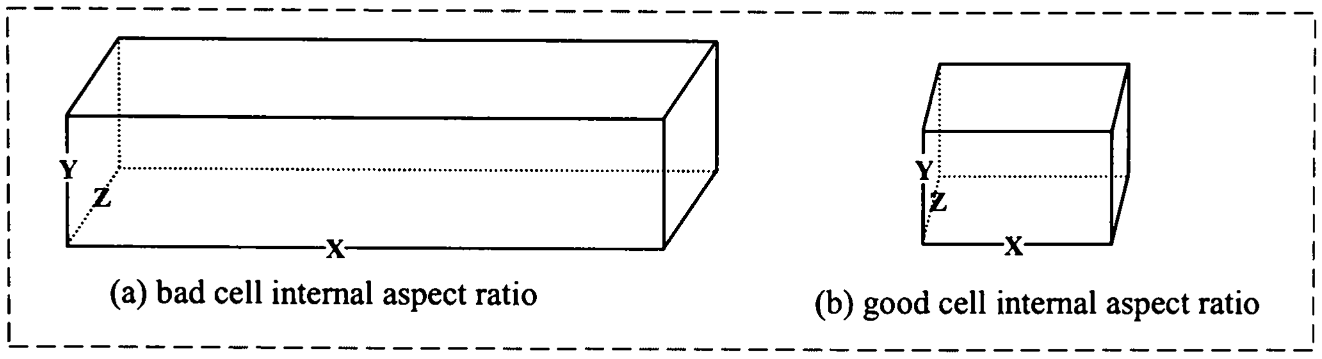

Y
Z
X
(a) bad cell internal aspect ratio
Y
Z
X
(b) good cell internal aspect ratio

Figure 4-3: Illustration of Cell internal aspect ratios

The cell neighbour aspect ratio is defined as the ratio of edge length between neighbouring cells in the same direction. In Figure 4-4, the cell neighbour aspect ratio is given by L_n/L_{n+1}.

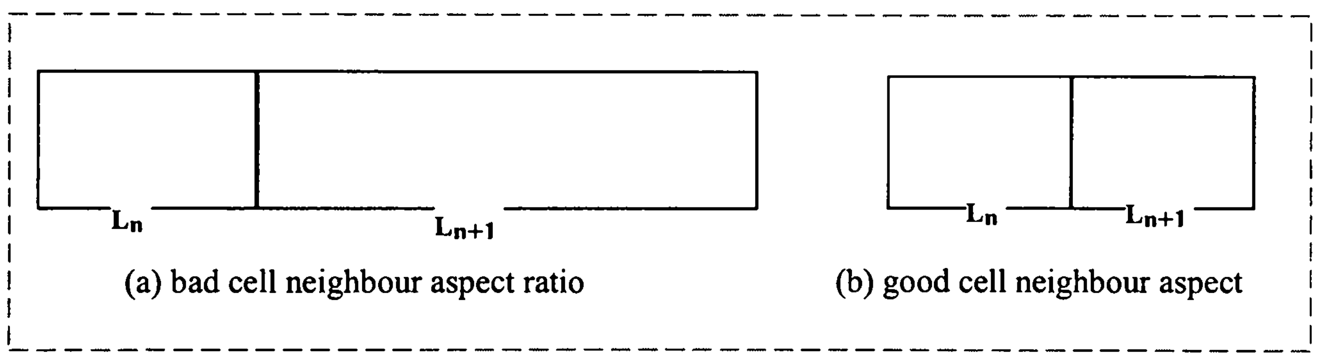

Ln
Ln+1
(a) bad cell neighbour aspect ratio
Ln
Ln+1
(b) good cell neighbour aspect

Figure 4-4: Illustration of Cell neighbour aspect ratios

A multi-block structured mesh which provides a useful compromise between the simplicity of single-block grids and the ability to handle complex geometry that completely unstructured grids would allow. It offers reasonable flexibility for a large variety of geometries. The first step of the multi-block mesh/grid generation is the construction of the grid topology, i.e. subdividing the calculation domain into a reasonable number of blocks and then each block is further divided into the actual cells for the grid generation. This is not at all straightforward; the quality of the topology directly influences the quality of the numerical grid (aspect ratio of control volumes, and the size ratio of two neighbouring volumes). When cell aspect ratio problems exist, a commonly used refinement strategy is to increase the number of cells in all directions across the whole domain. This is clearly not ideal and in fact, it will add considerably more cells than is actually necessary. This approach has overlooked the fact that the generated mesh is a multi-block structured mesh.

4.5.2 A new approach to solve cell aspect ratio problems

With multi-block structured mesh, there is definitely a better approach for handling cell additions locally (i.e. within a block). In doing so, it is more efficient and economical in terms of adding minimum possible cells to solve the cell aspect ratio problem. It is worth mentioning that adding more cells is not the only solution that could be used to solve the cell aspect ratio problem. If the problem is isolated to simply cell aspect ratios, then coarsening the mesh is an obvious alternative to solve this problem, however, it is very likely to compromise the existing mesh quality. So in the new approach proposed, the system should only ever consider adding more cells to the domain. The idea behind the new block-wise mesh justification algorithm is actually very simple, since it uses the multi-block structure of the generated mesh, and only adds cells to the target (problematic) blocks, in the problematic direction, instead of increasing the cell numbers in all blocks and in all directions. This should lead to a substantial reduction in the overall cell budget and consequently saves a considerable amount of computation time compared to the former refinement strategy.

The question remains, how is it possible to locally solve the aspect ratio problem (i.e. in a block wise sense)? It is worth concentrating on the cell internal aspect ratio problem, because it is more complex than the cell neighbour aspect ratio problem. The cell internal aspect ratio problem could be caused by excessive difference between length of any two edges, it is a 3D problem, on the other hand, cell neighbour aspect ratio problems are simply caused by the excessive difference between size of two adjacent cells in the same direction so it is one dimensional problem and should be easily dealt with using the existing cell distribution algorithm (i.e. to put a restriction on the maximum cell length difference allowed between adjacent cells).

If any of these aspect ratio problems are detected in x/y or y/z or x/z with greater than the configured tolerance (x, y, z are the edge lengths in the cell being considered), then that cell is deemed to have failed the cell-length aspect ratio test. For simplicity, consider the example of x/y greater than the tolerance, the solutions to this problem could be make x smaller or to make y bigger, or to do both. For the reasons just mentioned above, the action taken is to make x smaller (thus introducing more cells). Figure 4-5 above shows that how the proposed block-wise mesh justification procedure to solve the cell aspect ratio problems can be achieved in an abstract level.

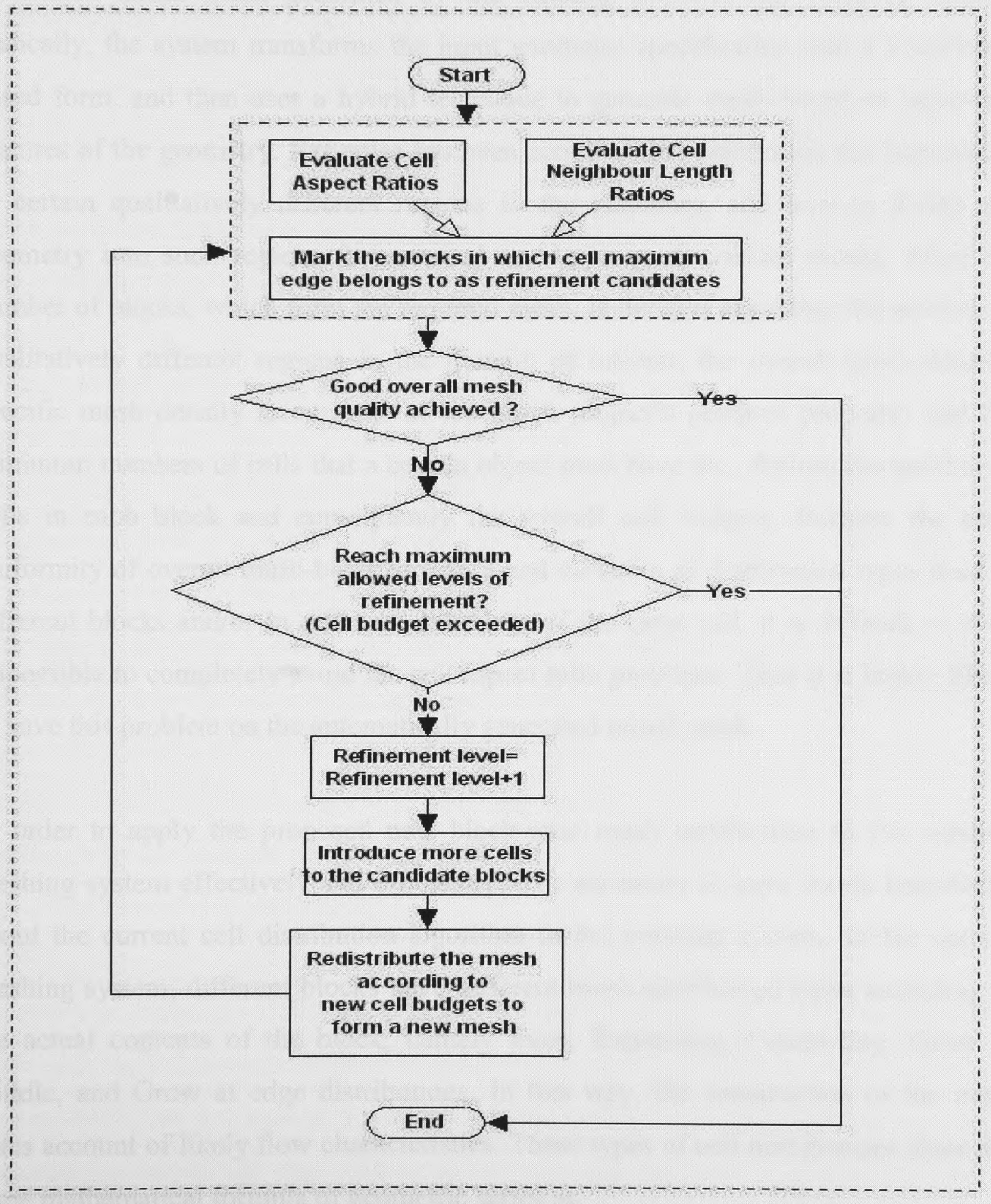

Start
Evaluate Cell Aspect Ratios
Evaluate Cell Neighbour Length Ratios
Mark the blocks in which cell maximum edge belongs to as refinement candidates
Good overall mesh quality achieved ?
Yes
No
Reach maximum allowed levels of refinement? (Cell budget exceeded)
Yes
No
Refinement level= Refinement level+1
Introduce more cells to the candidate blocks
Redistribute the mesh according to new cell budgets to form a new mesh
End

Figure 4-5: Flowchart of the block-wised mesh justification procedure

4.5.3 Implementation of block-wise mesh justification

In order to test validity and benefits of proposed block-wise mesh justification algorithm, it has been implemented in the *SMARTFIRE* meshing system [Ewer-00] [Taylor-97].

First of all, it would be useful to know that why this type of problem can happen in an automated initial mesh generated by the existing *SMARTFIRE* meshing system. Basically, the system transforms the input geometry specification into a knowledge based form, and then uses a hybrid technique to generate mesh based on important features of the geometry. Expertise has been acquired that determines the importance of certain qualitatively different regions in the geometry, and how to divide the geometry into such regions (represented as blocks in the actual mesh). After the number of blocks, which form the required mesh, is decided according the number of qualitatively different regions in the domain of interest, the overall mesh density, specific mesh density to contents of the block (object's physical property) and the minimum numbers of cells that a certain object must have etc., defines the number of cells in each block and consequently the overall cell budgets. Because the non-uniformity of overall multi-block structure and variation of distribution types used in different blocks and/or in different directions of the same cell, it is difficult or even impossible to completely avoid the cell aspect ratio problems. Thus it is highly likely to have this problem on the automatically generated initial mesh.

In order to apply the proposed new block-wise mesh justification to the existing meshing system effectively and efficiently, it is necessary to have inside knowledge about the current cell distribution algorithm in the meshing system. In the current meshing system, different blocks have different mesh distribution types according to the actual contents of the block, namely Even, Expanding, Contracting, Grow in middle, and Grow at edge distributions. In this way, the construction of the mesh takes account of likely flow characteristics. These types of cell distributions share the same mathematical formula of geometric series as,

$$\begin{array}{llll} a_n=a_1*q^{n-1} & & \text{(a)} & \\ s_n=a_1*(1-q^n)/(1-q) & (q\neq 1) & \text{(b)} & \textbf{(4.2)} \\ s_n=a_1*n & (q=1) & \text{(c)} & \end{array}$$

where,

a_n denotes the nth element in the geometric series

a_1 denotes the first element in the geometric series

q denotes the geometric series parameter

s_n denotes the sum of the n elements in the geometric series.

As seen from this formula, the distribution type is determined by the geometric series parameter q.

If $q = 1$ then the distribution type is Even

If $q > 1$ then the distribution type is Expanding

If $q < 1$ then the distribution type is Contracting

Grow in middle type is just another variation of the expanding type, and grow at edge type is another variation of the contracting distribution type.

In the current meshing system, first of all, the geometric series parameter q in a certain type of distribution for a block is calculated in consideration of the average cell length in the neighbouring block, so as to minimize the chance of having neighbouring cell aspect ratio problem caused by excessive difference between sizes of neighbouring cells in the adjacent blocks. Once the geometric series parameter q is determined, and the size of the block s_n (it was determined while qualitatively different regions were identified) and the number of cells n (calculated by the density requirements) in the block are already known, then it is very easy to obtain the size of the first element in the sequence according formula (b) or (c) if q equals to 1, consequently the size of the following element in the sequence can be worked out according formula (a) and so on.

Referring to equation (4.2) in the previous section, if the geometric series parameter q is fixed or only allowed to change towards 1 if it has to, then the size of any cell in a chosen block can get smaller by simply adding one or more cells (depending on the required degree of cell size reduction) to the block, and use the same distribution algorithm as before to redistribute all the cells in that block. Based on this knowledge, the block-wise mesh justification procedure implemented in the existing system to handle a distorted mesh (in terms of having cell aspect ratio problems) is presented by a detailed pseudo-programming language in Figure 4-6 below.

```
Procedure to the block-wised mesh justification (note: tolerance <1)

Begin {Justify Mesh Quality}

        Repeat {Evaluate Mesh Quality, Evaluate Cell Neighbor Length Ratios}
            Loop {checking on all the cells}
                If cell minimum edge/ cell maximum edge < tolerance
                    Mark the block which cell maximum edge belongs to
                End if
            End Loop {checking on all the cells}

            Loop {Adjust Mesh Quality}
                If {bad mesh quality and loop count < maximum loop and
                    cell budget is not exceeded }
                        loop count++
                        Loop {x direction blocks}
                            If the block Bx is marked
                            Introduce more cells in the block Bx.
                            End if
                        End loop

                        Loop {y direction blocks}
                            If the block By is marked
                            Introduce more cells in the block By
                            End if
                        End loop

                        Loop {z direction blocks}
                            If the block Bz is marked
                            Introduce more cells in the block Bz
                            End if
                        End loop

                        Redistribute the mesh according new cell budgets to
                        form a new mesh
                Else
                        End {Evaluate Mesh Quality}
                End If
            End loop {Adjust Mesh Quality}
        End
    End {Justify Mesh Quality}
```

Figure 4-6: Pseudo-programming description of the block-wised mesh justification procedure

In fact, this code has no way to know the exact number of cells that need to be added to the target block, and it is not guaranteed to solve the cell aspect ration problem at one pass since the refinement is a 3D problem with other directional changes potentially worsening the problems in the orthogonal directions. In the block-wise mesh justification algorithm presented here, cell additions are limited by adding one cell at a time to the target block, so as to make sure the overall cell budgets are kept at a minimum level. Some cell aspect ratio problems, for very complex geometries, are difficult to solve and may require many more iterations of adding cells to solve them. In this case, an iteration counter is also used to restrict the potential for excessive numbers of iterations to ensure that the overall cell budgets are always at an acceptable level.

4.5.4 Benefits of block-wise mesh justification

In order to show the benefits of this newly developed block-wise mesh justification to solve cell aspect ratio problem. Three scenarios with different size of domain and complexities are presented here.

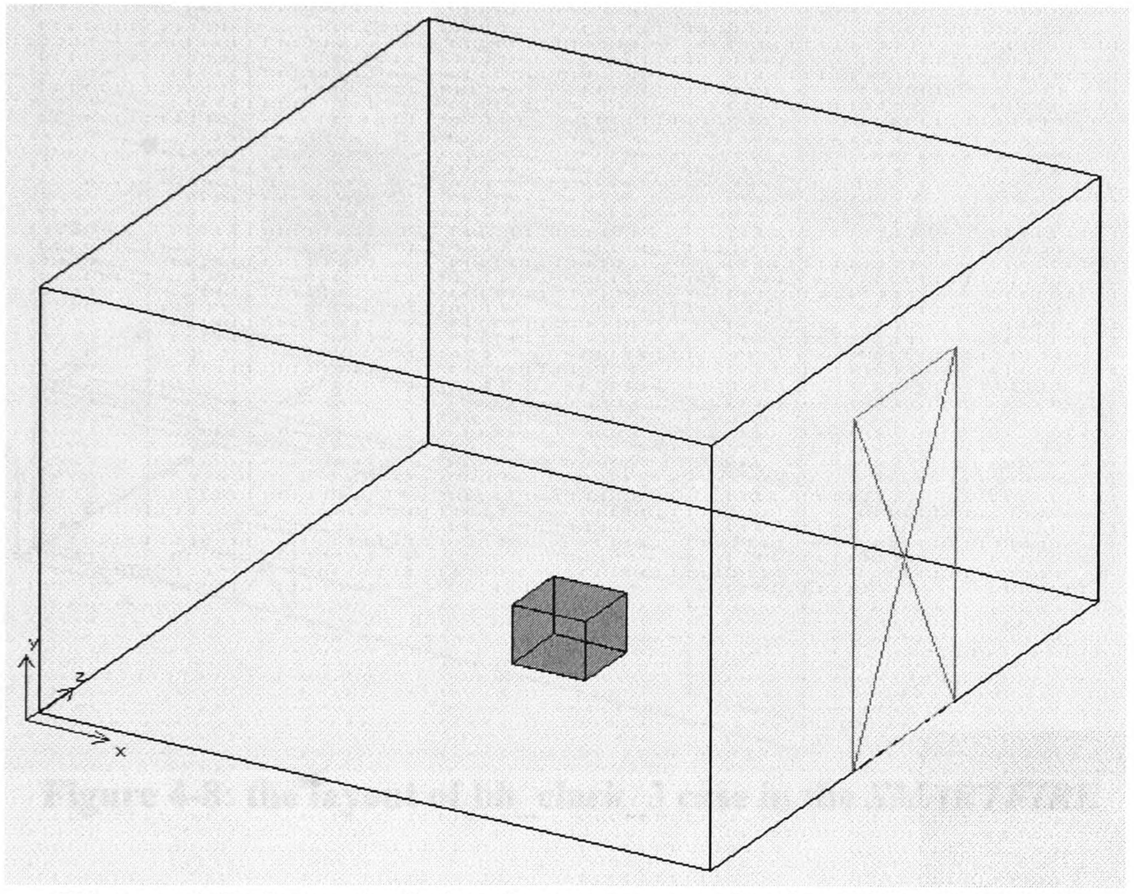


Figure 4-7: layout of simple room case in the *SMARTFIRE*

As seen from Figure 4-7, the first case was a small room (2.8m* 2.8m* 2.2m) with a fire (0.3m×0.3m×0.3m) in the centre of the room, directly on the floor. The room had a single vertical vent (door =1.83m * 0.74m). The second one was a multi-storey building so called bh_clark_3 case, the building sized 15.8m×21.4m×11.2m and has several small vents located in different floors and ceiling. Fire object (2.0m×2.0m×2.0m) located in the second floor near west end. The layout of the bh_clark_3 case is shown in Figure 4-8.

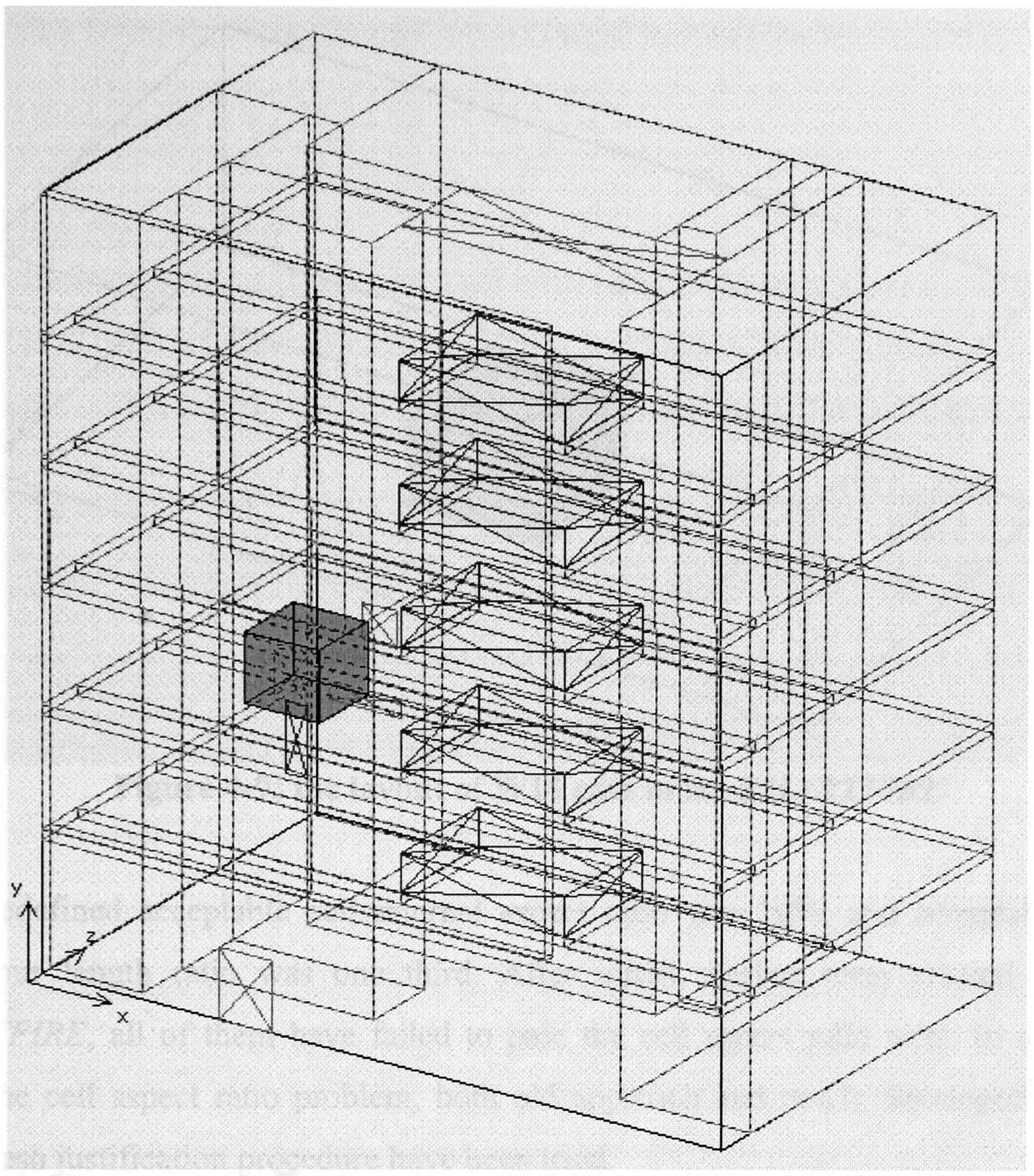


Figure 4-8: the layout of bh_clark_3 case in the *SMARTFIRE*

The third case is known as W14 case, the fire compartment measured 14.4 m × 7.2 m × 3.53 m and contained a door of dimensions 2.97 m × 2.13 m. The walls of the compartment were made of aerated concrete blocks with thickness of 0.3 m. The fire (2.4m×2.4m×1.4m) was located on the floor in the centre of the room as shown in Figure 4-9 below.

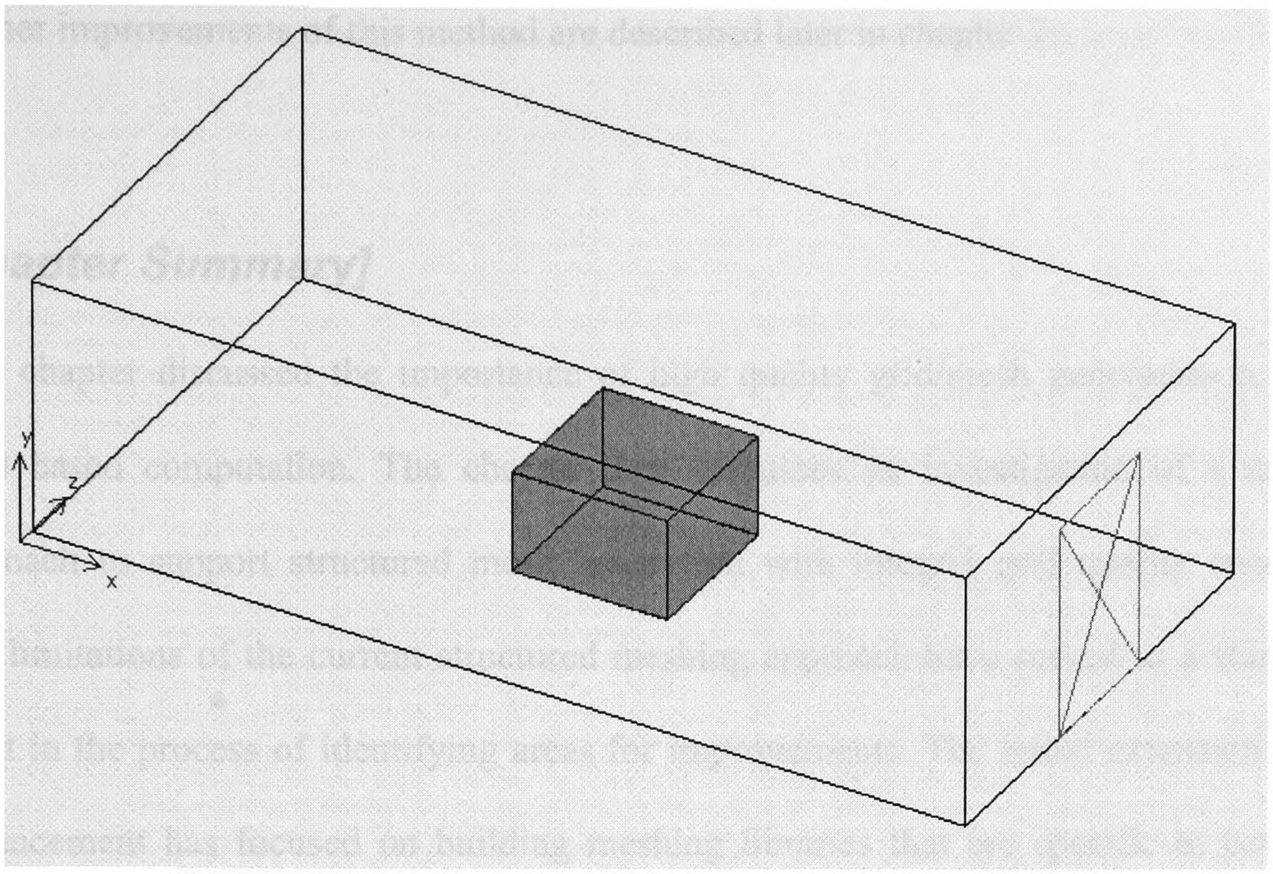


Figure 4-9: the layout of W14 case in the *SMARTFIRE*

The predefined acceptable cell internal aspect ratio was 20%, and acceptable cell neighbour length ratio was one third. After initial meshes were created by the *SMARTFIRE*, all of them have failed to pass the cell aspect ratio tests. In order to solve the cell aspect ratio problem, both old approach and newly developed block-wise mesh justification procedure have been tried.

Table 4-4 shows the overall mesh budgets by using this block-wised justification algorithm compared to the old approach used in current *SMARTFIRE* system.

Cases	Cell Budget (whole region mesh refinement scheme)	Cell Budget (Block-wise mesh refinement scheme)	Savings compared to use whole region refinement regime
A simple room	23184	18711	19.29%
bh_clark_3 (multi-storey building)	998070	506268	49.27%
W14 (part of a building)	397936	107712	72.93%

Table 4-4: Benefits of using a block-wised mesh justification
(Uses ACCEPTABLE_ASPECT_RATIO: 20% and ACCEPTABLE_LENGTH_RATIO: 33.3%)

Further improvements of this method are described later in chapter 7.

[Chapter Summary]

This chapter discussed the importance of high quality grid/mesh generation to the CFD based computation. The chapter then discusses an investigation of a novel approach to support structured mesh generation with integral grid quality control. The limitations of the current structured meshing approach have served as a starting point in the process of identifying areas for improvements. The initial extension and enhancement has focused on building meshing libraries that are specific to certain identified and distinct classes of geometry. This was identified as necessary because the original meshing library was based on rules and configuration settings for the meshing of a simple room scenario. In order to use this classification of geometry, the universe of fire scenarios is divided into a number of typical categories according to the geometry type and fire source characteristics. A case recognition algorithm has been investigated to automate the matching of an arbitrary scenario to the geometry categories and hence to select the most appropriate meshing parameters from the corresponding library. One of the major problems which has been identified with initial automatically generated meshes, is that of cell aspect ratio problems. In order to mitigate this problem, a dedicated mesh justification algorithm has been devised. Validation and testing of the new method has demonstrated that the newly developed block-wise mesh justification can solve the cell aspect ratio problem more effective and efficiently than the previous approach. Although these investigations have been specific targeted on structured meshing techniques, especially, for the hybrid structured meshing approach, which is still widely used by FFM and other general CFD system. These considerations are appropriate to most structured meshing systems and, furthermore, the proposed approach for the initial meshing would be applicable for unstructured meshing systems too.

5 Development of Local adaptive mesh refinement

[Chapter Overview]

In the previous chapter, a new approach was described, to extend and enhance a structured meshing system's ability for constructing a "reasonable" or "good" quality initial mesh for arbitrary fire scenarios. This approach was mainly knowledge based. As has already been stated, the main intention for this research is to emulate an Experts' ability to run fire simulations successfully even if previous attempts have failed. In order to emulate how an Expert would achieve a suitable quality for the mesh, the software system should be able to make 'run-time' adjustments to the mesh to suit the precise meshing requirements of the scenario to be modelled. This is considered to be a performance based approach.

In this chapter, the meshing strategy is used to progress from the generalization of the scenario (i.e. the case classification) to a detailed specification – which gives the mesh distinctive requirements according to the problem that will be simulated. This is accomplished using mesh refinement that uses localized quality feed back from the simulation state/results to direct the mesh improvements to those areas where they are most needed. This has been investigated using adaptive local re-meshing a progressively more refined mesh. The computational complexity of CFD simulations and the typically protracted run-times mean that it is important to consider the issue of computational expense associated with the meshing. It is vital to keep the mesh cell budget to a minimum level without adversely affecting the solution quality, so the re-meshing refinement has to add cells sparingly to achieve the desired mesh quality. This approach to meshing is based on adaptive finite element techniques commonly used in the Finite Element Analysis.

5.1 Introduction

In the previous chapter, the characterisation of typical fire scenarios was described, and case recognition was used to facilitate the automation of selecting the 'most appropriate' set of parameters and rules, as defined in the corresponding mesh library. A number of new mesh libraries (for various fire scenarios other than room based ones) have been added to the extended *SMARTFIRE* meshing library. However, because the parameters/rules set in these meshing libraries are mainly taken from the static view of the geometry type and problem specification of the scenario, they are merely guidelines that can help to produce a "reasonable" initial mesh.

Most fire related phenomena are generally not static, since, given sufficient heat, fuel and oxygen, a real fire will go through various stages of ignition, smouldering phase, growth stage, fully developed period and decay phase. There may also be rapid or gradual fire spread and transitional effects from the failure of materials affected by the fire. There are considerable differences between fires due to the nature and time scales associated with each phase. These in turn are controlled by the exact chemical and physical nature of the fuel load, the size of the enclosure, the degree of ventilation, type of fire initiation and the history of conditions that the fire experiences. These factors mean that the numerical solution of fire problems is difficult. Conventional approaches that calculate solutions only on a prescribed mesh, can fail to adequately resolve all of the fire phenomena, can have excessive computational costs, can fail to complete or – most critically – can produce incorrect results.

A typical CFD based analysis would proceed 1) from the creation of a mesh and selection of configuration settings, 2) to the generation of a solution, 3) to an assessment of the solution accuracy, and finally 4) to the analysis of the results. Experience is the traditional method of determining whether or not the mesh and configuration will be optimal or even adequate for a particular simulation scenario. Furthermore, even a seemingly small change to the simulation scenario could have a drastic influence on the meshing and configuration needed for a successful outcome.

Adaptive procedures try to automatically refine the mesh and/or adjust the configuration in order to achieve a solution that has a specified accuracy, in an optimal fashion.

5.2 Adaptive Techniques

Various adaptive meshing methods [Babuska-83] [Arney-90] [Dorr-84] have been studied previously. The commonly used adaptive meshing procedures include:

- Local refinement and/or coarsening of a mesh (h-refinement).
- Relocating or moving a mesh (r-refinement), and
- Locally varying the polynomial degree of the basis (p-refinement)

These strategies may be used singly or in combination. R-refinement alone is generally not capable of obtaining a solution to a specified accuracy. If the mesh is too coarse, it might be impossible to achieve a high enough degree of precision without adding more mesh cells or altering the basis. In some senses p-refinement is the most powerful. Exponential convergence rates are possible when the meshing solution is smooth. It is apparent that h-refinement is by far the most popular and widely-employed technique. H-refinement can also increase the convergence rate [Flahety-00]. These adaptive procedures are mainly applied to Finite Element Methods, while the *SMARTFIRE* is built based on Control Volume (FCV) methods. Both Finite Element analysis (FEA) and Control Volume simulations require high quality mesh generation. The mesh refinement techniques for FEA are completely applicable for FCV formulation in our context. Usually posterior error estimates [Verfurth-96] are necessary to terminate an adaptive procedure. However, optimal strategies for deciding where and how to refine a mesh or to move a mesh or to change the basis, are rare (for more detailed discussion of adaptive mesh refinement, please refer to the related section in chapter 3).

5.3 An adaptive local mesh refinement method for the SMARTFIRE meshing system

The adaptive mesh refinement algorithm is based on general principle of adaptive techniques and it has been optimised for use within the structured meshing architecture of the *SMARTFIRE* Environment. This mesh adaptation ensures that the adaptive meshing is the most suitable, efficient and effective that can be applied to the mesh, using the existing meshing techniques. The adaptive mesh refinement procedure consists of these parts : (i) construction of initial base mesh, (ii) local refinement of the base mesh in regions where resolution is inadequate, and (iii) correction and justification of the refined mesh when it becomes overly distorted (i.e. if aspect ratio problems are identified). This approach utilizes earlier works on the case classification, case recognition and block-wise mesh justification, and exploits the block-wise structure of the *SMARTFIRE* mesh.

5.3.1 Initial base mesh construction and mesh distortion justification

The efficiency and robustness of our adaptive mesh refinement strategy are dependent on our ability to generate a suitable initial base mesh and to correct/justify the base mesh when it becomes overly distorted (i.e. the tolerance on cell aspect ratios cannot be satisfied) due to the local mesh refinement. An initial base mesh that has too few cells will result in excessive refinement, while a properly constructed initial base mesh can reduce the need for many further refinements, thus increasing the efficiency of the mesh refinement procedure. An effective mesh distortion correction and justification algorithm is needed to prevent the additional errors caused by the possible mesh distortion that can be introduced during local mesh refinement, therefore ensuring the robustness of the adaptive procedure.

Most adaptive algorithms either do not pay enough attention to the initial base mesh construction or make the initial base mesh construction itself an iterative process, for example, a commonly used approach to initial mesh generation is to use the error indicators computed by a trial solution to determine an initial base mesh that approximately equi-distributes the error indicators. This usually starts with a coarsest uniform mesh and, subsequently, local mesh refinement is used until the prescribed tolerance is attained. Obviously, this is not ideal, as the initial mesh construction itself may require a number of solver iterations thus bringing extra costs to the adaptive mesh refinement process. Our initial base mesh is generated using rules and uses the earlier work on case classification and case recognition. For example, by giving a test case to the new software, with the newly deployed case classification and case recognition techniques, the software is capable of recognising in which category the test case belongs and consequently retrieves the most appropriate meshing parameters from the corresponding mesh library to construct a suitable initial base mesh. In this way, the initial mesh is already built using the best available knowledge. This helps to minimize the need for further refinements (For details about case recognition and extended libraries for meshing, please refer to the previous chapter).

The progressive mesh refinement process will inevitably make the aspect ratios unsatisfactory at some stage. The base mesh will become distorted and it is likely that additional errors or stability problems will be introduced. An effective adaptive mesh refinement procedure must be able to deal these issues. Otherwise, the process could become self defeating, since errors reduced by the refinement could be cancelled out by the additional errors caused by distortion of the base mesh. Further refinements on the now distorted based mesh could result in even more severe mesh distortion and this will most likely eventually break the refinement process (i.e. the errors are seen to become bigger after refinement). The usual solution to the mesh distortion problem is to regenerate a new base mesh which will have the same number of cells as the old one, and is generated using the same procedure for creating an initial base mesh. The problem with this approach is that, if the method used to generate the initial base mesh is not guaranteed free from aspect ratio problem, then the regeneration of new based mesh may not be able to solve the problem. Furthermore, even with such a method available, then regeneration of a new based mesh would take too much away of the good work already done by the local refinement procedure.

The proposed approach to solve this problem is that, instead of re-developing a mesh generation method which is guaranteed free from the cell aspect ratio problem (which is very difficult to do considering that the initial base mesh should not be too fine), a "stand-alone" algorithm should be designed and implemented, to look after the cell aspect ratio problem. This should repair the base mesh at any point of the adaptive mesh refinement process, as and when a problem occurs. In this way, the effectiveness and flexibility of the adaptive procedure would be enhanced. As discussed in the previous chapter, the block-wise mesh justification algorithm (for details, please refer to the previous chapter) developed in this research, is able to solve the cell aspect ratio problems efficiently and effectively. It was not clear if it would be possible to use the block-wise mesh justification algorithm directly here, since this is entirely dependant on how the refinement method would work. The choice of refinement method will inform the decision on how to ensure cells quality under mesh refinement.

5.3.2 Methodology for Choosing Candidates for Refinement

Local error estimates are often used as refinement indicators and to produce solutions that satisfy either local or global accuracy specifications. Successful error estimates have been obtained using h-refinement[Berger-84], where the difference between solutions on different meshes is used to estimate the error. However, this type of error estimation is expensive as it requires generating a second solution – on a different mesh. It is better to make use of the solution itself to derive the estimates of the error in a suitable norm. This can be achieved by using a posterior error estimate. A-"posteriori" literally means "derived by reasoning from observed facts", and the posterior error estimates can provide accuracy assessments that are necessary to terminate the adaptive procedure. The error indicator can either be an estimate of the local discretization error or another function that is large where additional resolution is needed and small where less resolution is required. Fortunately, with the existing *SMARTFIRE* solvers, a numerical method is available for calculating approximate solutions and error indicators of each mesh cell. A global error estimate, in a particular norm, can be computed by summing each nodal error contributions as,

$$\boxed{E = \frac{\sum_{i=1}^{n} (e_i)^2 \times v_i}{V}} \qquad \textbf{(5.1)}$$

Where n is the number of cells in the mesh and E is the global error indicator.
e_i is the local error indicator at element i and v_i denotes the volume of the i th cell.
V is the domain volume.

As shown in equation (5.1), the method of determining where adaptivity is needed is to use e_i as an enrichment indicator. It is assumed that large errors come from regions where the local error estimate e_i is large and even more so when the cell volume is large as well, hence, this is the natural location where the mesh should be refined. By using a combination of the interrelated global error tolerance and local error tolerance, the global mesh quality is assured and, at the same time, only regions that are likely to cause problems are actually refined, which makes the adaption more efficient and effective.

5.3.3 Refinement method

A common strategy for refinement of an element of a structured quadrilateral-element mesh is achieved by bisection. This requires mesh lines running all the way through the mesh to the boundaries, in order to retain the four-neighbour structure as shown in (a) of Figure 5-2. This strategy is simple to implement and has been used with finite difference computation [Zhang-96].

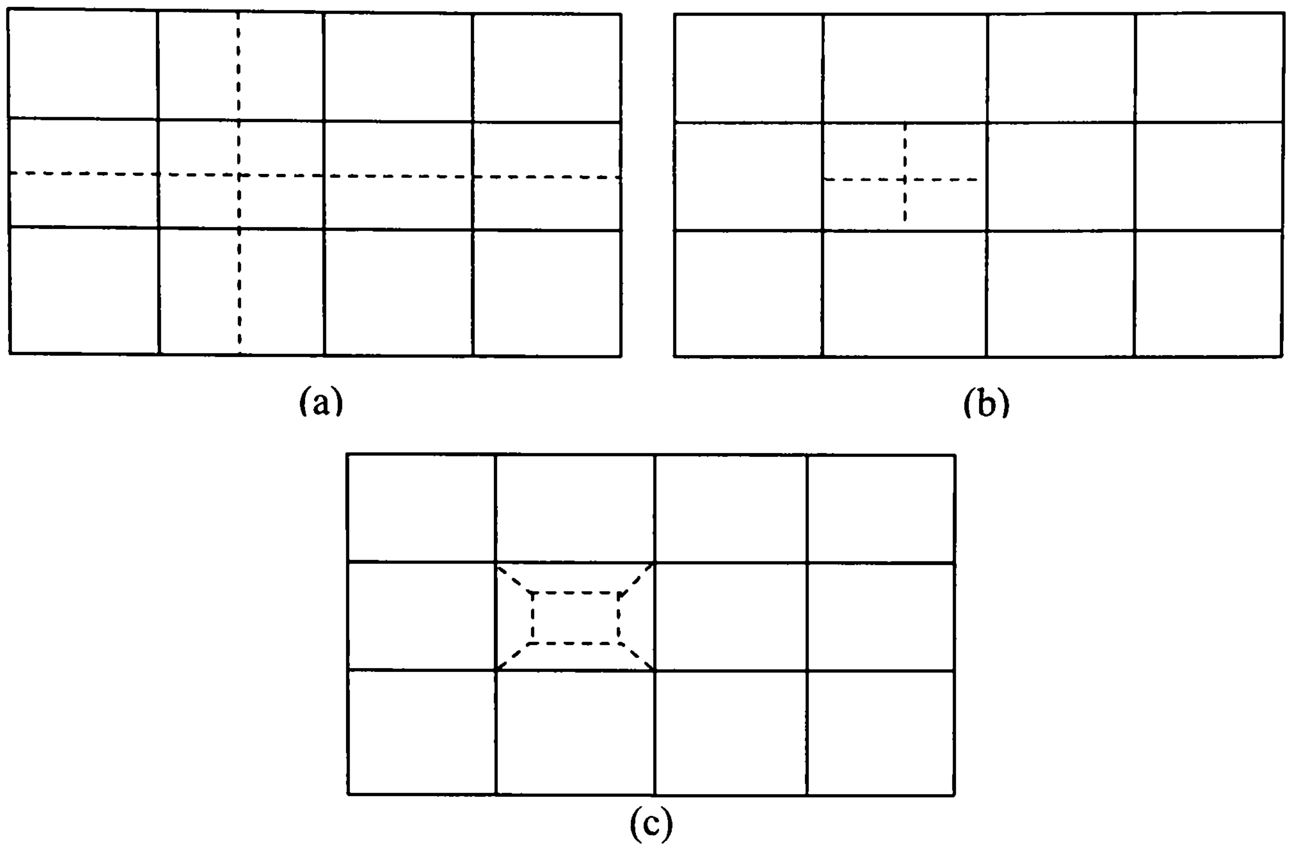


Figure 5-2: Bisection techniques for a cell in a structured rectangular-element mesh (a) creating mesh lines running between the boundaries, (b) introducing short mesh lines by creating irregular nodes, and (c) using fully unstructured refinement.

However, this clearly refines many more elements than necessary. The customary way of avoiding the excess refinement is to introduce irregular nodes where the edges of a refined element meet at the mid-sides of a coarser one as shown in (b) of Figure 5-2. This approach would produce a mesh that is no longer structured and the standard method of basis construction would create discontinuities at the irregular nodes [Flahety-00]. This technique would be suitable for FEA and could also be used in an unstructured control volume CFD code. One form of refinement which would avoid the introduction of discontinuities would be to use refinement with hypercubes, (as shown in (c) of Figure 5-2) although it is highly likely that this would introduce accuracy and stability issues due to the poor shape and aspect of some of the sub-cells. This negative effect on the cell shapes would make it harder to assess the quality of the mesh and hence would invalidate the feedback mechanism from the CFD simulation to the meshing. This is noted as an area that should receive further attention in future research because of the large potential savings of more localised, unstructured, refinement.

To find a simple but effective refinement method for use in a structured meshing system, a suitable refinement method was developed which could be directly incorporated into the existing meshing strategy. Essentially, the structured meshing system transforms the input geometry specification into a knowledge based form, and then uses a combination of meshing rules (appropriate to similar classes of fire simulation case) and parameters from the meshing library to generate a multi-block structured mesh. Expertise embedded in the meshing control library determines the importance of certain qualitatively different regions (blocks) in the geometry, and how to divide the geometry into such regions (blocks). Individual meshing parameters are determined for each region (block). In order to test the concepts investigated in this chapter, the techniques were implemented and tested in the SMARTFIRE automated meshing system. Figure 5-3 shows an example of a *SMARTFIRE* block-wise generated mesh.

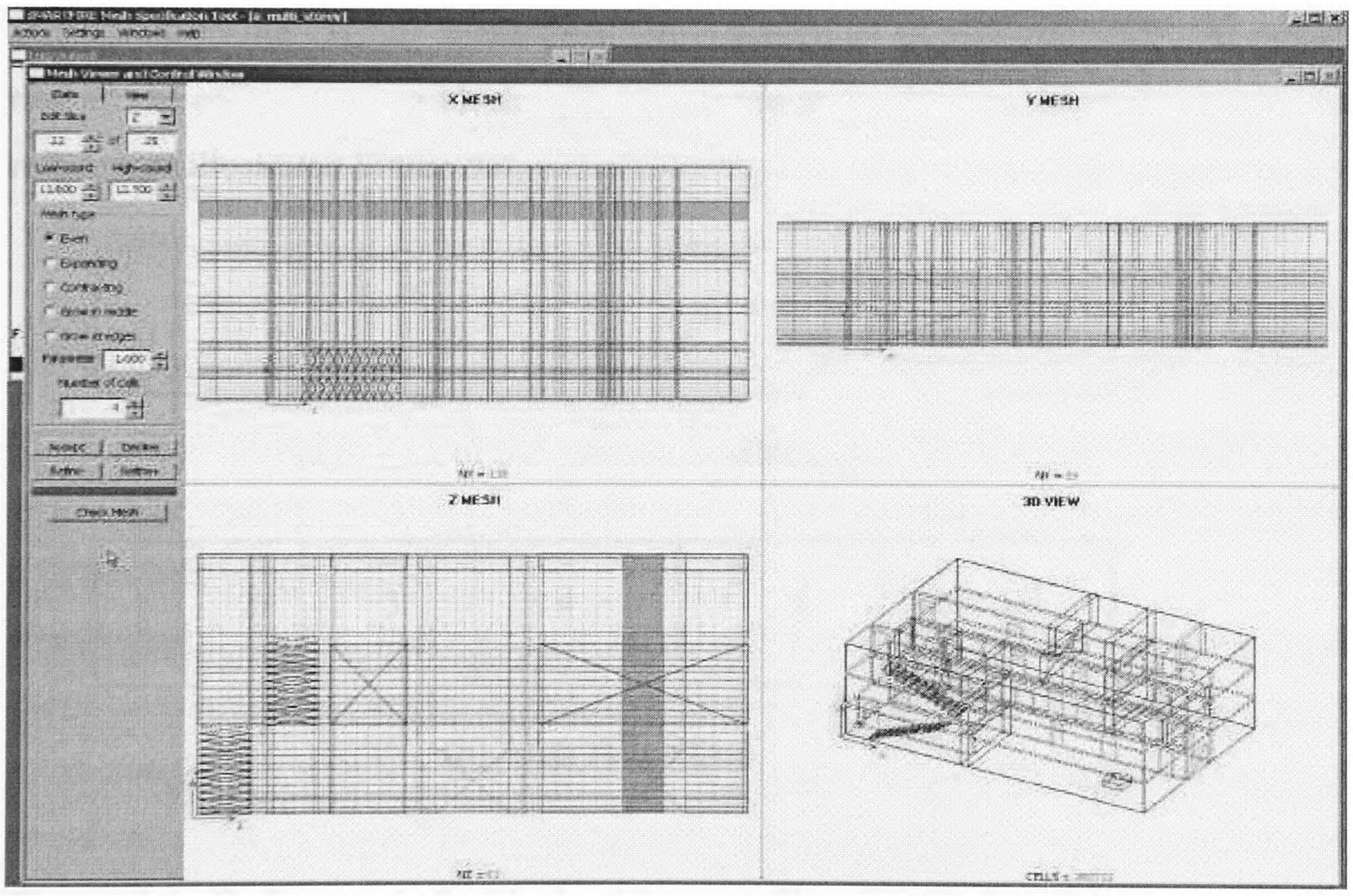
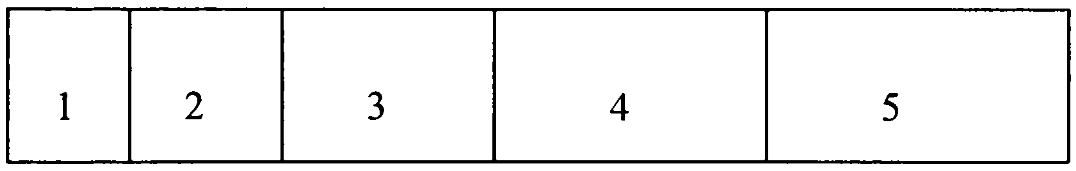


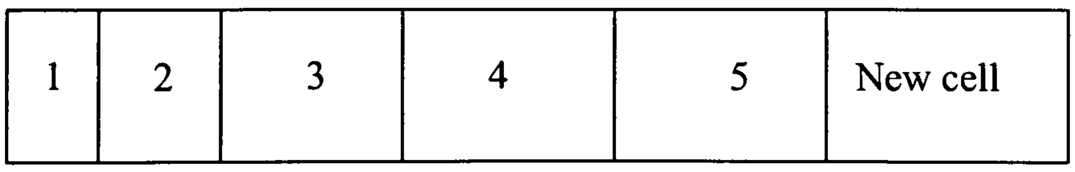
Figure 5-3: Interactive Meshing Tool showing a mesh that has been generated for a multi-storey building simulation case.

The existing meshing strategy has proven to be well behaved over many years of usage by *SMARTFIRE* users. Obviously, local refinement method must make sure all the best features of the meshing are not jeopardised while taking advantage of the new localized refinement techniques.

As seen from Figure 5-3 above, the number of blocks is dependant on the complexity of the geometry. Basically each individual object will occupy at least one block. In the case of overlapping objects, each overlapping object can occupy more than one block. In this way, it is possible to ensure that, within a particular block, the cells have roughly the same requirements as determined by their physical properties (i.e. if the block is the fire block, then all the cells within the block will have a meshing strategy suitable for cells in a fire). This gives opportunities to treat blocks as units to do refinement rather than using individual cells as units for refinement. This means that meshing can concentrate on those blocks which have a greater percentage of problem cells needing to be refined. It is then possible to optimize the refinement to add a minimum number of cells (usually one cell at a time is sufficient for most cases), and hence to concentrate on providing the best possible cell distribution within the chosen block. This change of distribution (please refer to previous chapter for more detailed discussion on the existing mesh distribution algorithm in *SMARTFIRE*) will ensure that all the cells sizes in a block will decrease to some degree. An example of such refinement illustrated Figure 5-4.



(a) Block of 5 cells before refinement



(b) After refinement

Figure 5-4: Refinement of a block with expanding distribution type

This has several advantages over simple bisections of the individual cells.

- This approach does not suffer from extra repair work when local refinement causes discontinuities at the irregular nodes.
- After whole block refinement, the global error level trend is downwards and the local continuity is guaranteed with only minimum cell additions.

- The earlier block-wise justification algorithm is still valid, when the overall aspect ratio is still not satisfied.
- The fact that, when using bisection refinement for a cell, there is no proof of the optimality of enrichment in the vicinity of the largest local error estimate, which means that after bisection of the largest error cells, it still may be necessary to use bisection of its neighbouring cells as well. While the block-wise refinement, takes advantage of the overall multi-blocks structure and makes all cells in the target block smaller in size by adding only the minimum necessary number of cells. Hence the target block is refined with a high degree of confidence of minimizing the global errors.
- This local refinement procedure mainly adopts the h-refinement techniques as used in meshing for FEA. However this method, by using the refinement method as described, effectively combines some r-refinement as well. Regarding figure 5-4 again, the actual refined mesh is effectively the result of pre-refined mesh moving to the left where it was assumed it was the regions of large variation in the solution of the current block in the first place when it was decided to use expanding cell distribution type. So the implemented method uses a h-refinement combined with a "static" version of r-refinement without running into the difficulties of altering the Solver (as for a "pure" r-refinement, the solve has to offer at least some sort of interpolation/mapping functionality to derive variable values of the moved mesh from the original mesh before moving).
- The cell re-distribution is at the heart of the refinement method. Strictly speaking, it should be noted that this refinement strategy is a static re-meshing refinement rather than a dynamic refinement – which would require that data associated with the old cells would have to be interpolated in to the new cell distribution. (The disadvantage of this approach is further discussed below) Using this approach, it is easy to retain a sufficient degree of smoothness in the cell distribution. From an r-refinement point of view, the cell points are not moved independently, but rather each point appears to be coupled at least to its neighbours. Also, this method ensures that grid points are not moved too far or too fast, which avoids the potential problems of oscillations.

- After refinement, the original block structure has been maintained with each block still representing a unique object physical property.

The disadvantages

- Mesh lines still run to the boundaries to retain the regular Cartesian structure, this inevitably means that refinement has used more cells than is strictly necessary.
- Many of the cell centres will be redistributed after the refinement. This will make it difficult to re-use values from the pre-refined mesh for the refined mesh.

5.3.4 Summary of the algorithm

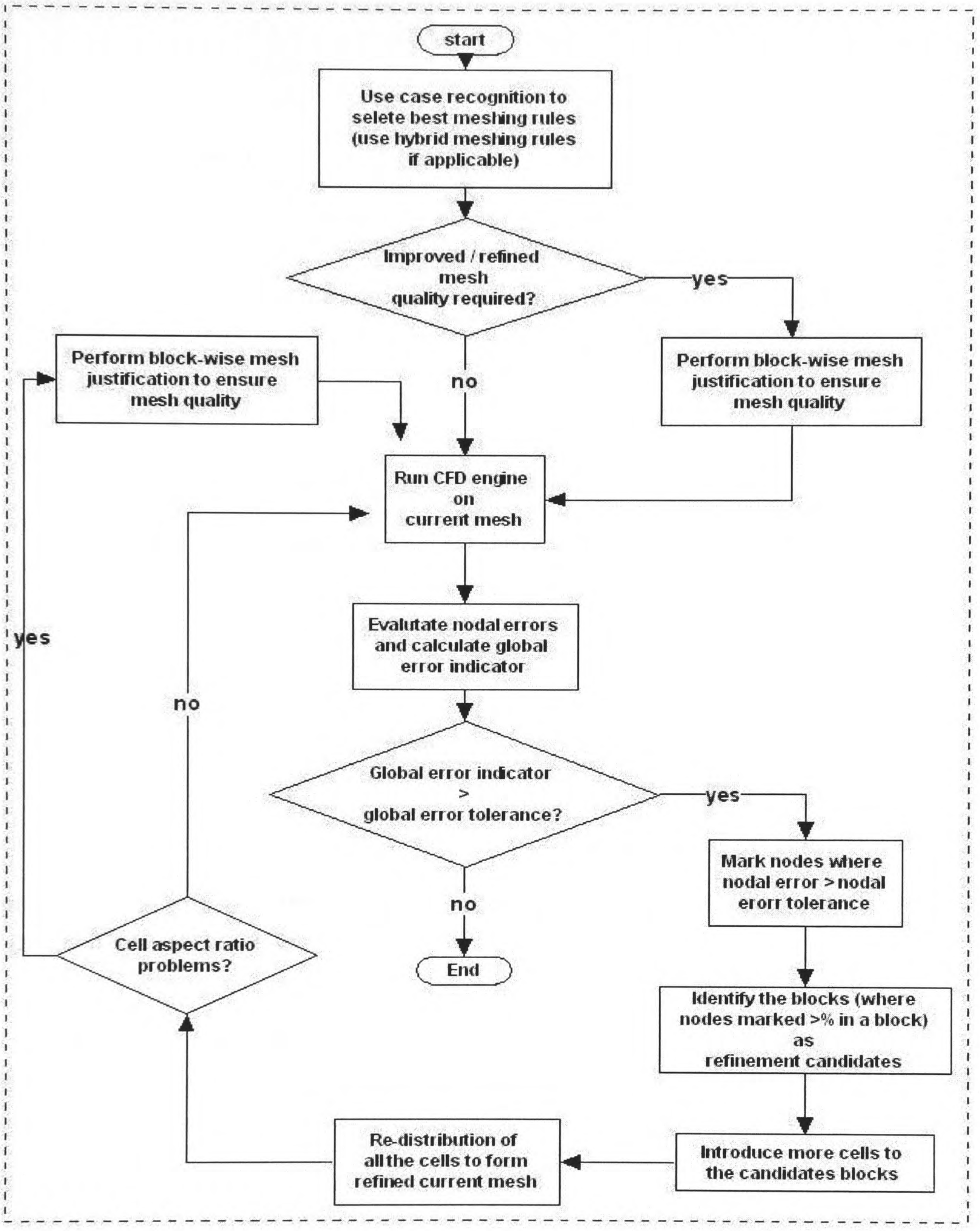

start
Use case recognition to selete best meshing rules (use hybrid meshing rules if applicable)
Improved / refined mesh quality required?
yes
no
Perform block-wise mesh justification to ensure mesh quality
Perform block-wise mesh justification to ensure mesh quality
Run CFD engine on current mesh
yes
no
Evalutate nodal errors and calculate global error indicator
Global error indicator > global error tolerance?
yes
Mark nodes where nodal error > nodal erorr tolerance
no
Cell aspect ratio problems?
End
Identify the blocks (where nodes marked >% in a block) as refinement candidates
Re-distribution of all the cells to form refined current mesh
Introduce more cells to the candidates blocks

Figure 5-5: Flowchart of adaptive local mesh refinement procedure for multi-block structured mesh

To summarise the adaptive mesh refinement procedure, a top-level description of the adaptive procedure is shown in Figure 5-5 above. And also a more detailed pseudo programming language is also given in Figure 5-6 below.

```
Procedure {adaptive h-local mesh refinement for multi-block structure mesh}

        Begin {Construct Initial Base Mesh}
                Use case recognition to select best meshing rules;
                * Select appropriate start up and mesh quality strategies;
                * Use hybrid meshing rules if applicable;
                * Perform Block-wise justification to ensure mesh quality;
        Repeat:
                Run CFD Engine on current mesh
                Evaluate Nodal Error for each node and calculate
        global_error_indicator

                If gobal_error_indicator > global_error_tolerance then
                    Mark nodes where nodal error > nodal error tolerance
                        If nodes marked >Δ% in a block then
                            Introduce more cells in Block
                            re-distribution cells(including newly added cells)
                        End if
                    Checking aspect ratio problems of the new mesh
                        If mesh becomes too distorted then
                            Perform block-wise mesh justification
                        End if
                Else
                    Terminate
                End If
        End

End {adaptive h local mesh refinement for multi-block structure mesh}

Note: * denotes optional selections
```

Figure 5-6: Pseudo-programming description of adaptive local mesh refinement for multi-block structured mesh

This procedure attempts to keep the global error indicator, which is the representation of the overall mesh quality, below a pre-defined global tolerance by introducing more cells into blocks where these exceed a percentage of nodes that have nodal error greater than a predefined nodal error tolerance. The separation of base mesh constructions, local refinement method and correction, and justification of the refined mesh, offers adequate flexibility for many future enhancements.

5.4 preliminary validation of the local adaptive mesh refinement method

The validity of the local refinement method was assessed using a simple two compartment case. This Fire experiment [Neilsen-00] considered was conducted at McLeans Island, in New Zealand to investigate the behaviour of pre-flashover fires and to validate zone and field modelling programs. The experiment was conducted to provide a more comprehensive set of data than the Steckler et al. experiments [Steckler-82] commonly used to validate zone and field modelling simulations.

5.4.1 General description of the Experiment

The experiment was conducted in a two-compartment structure(see figure 5-7); each compartment measured 2.4 m wide, 3.6 m long and 2.4 m high. A doorway, with dimensions 2.0 m high and 0.8 m wide, separated the rooms. The fire was placed in one room (here after referred to as the fire room). The room next to the fire room and separated by the doorway (here after referred to as the adjacent room), opened out to the atmosphere and had no soffit or doorway. The fire was of fixed size with a heat output of 160kW, centrally located in the fire compartment. The fire used LPG (liquid petroleum gas) as the fuel source.

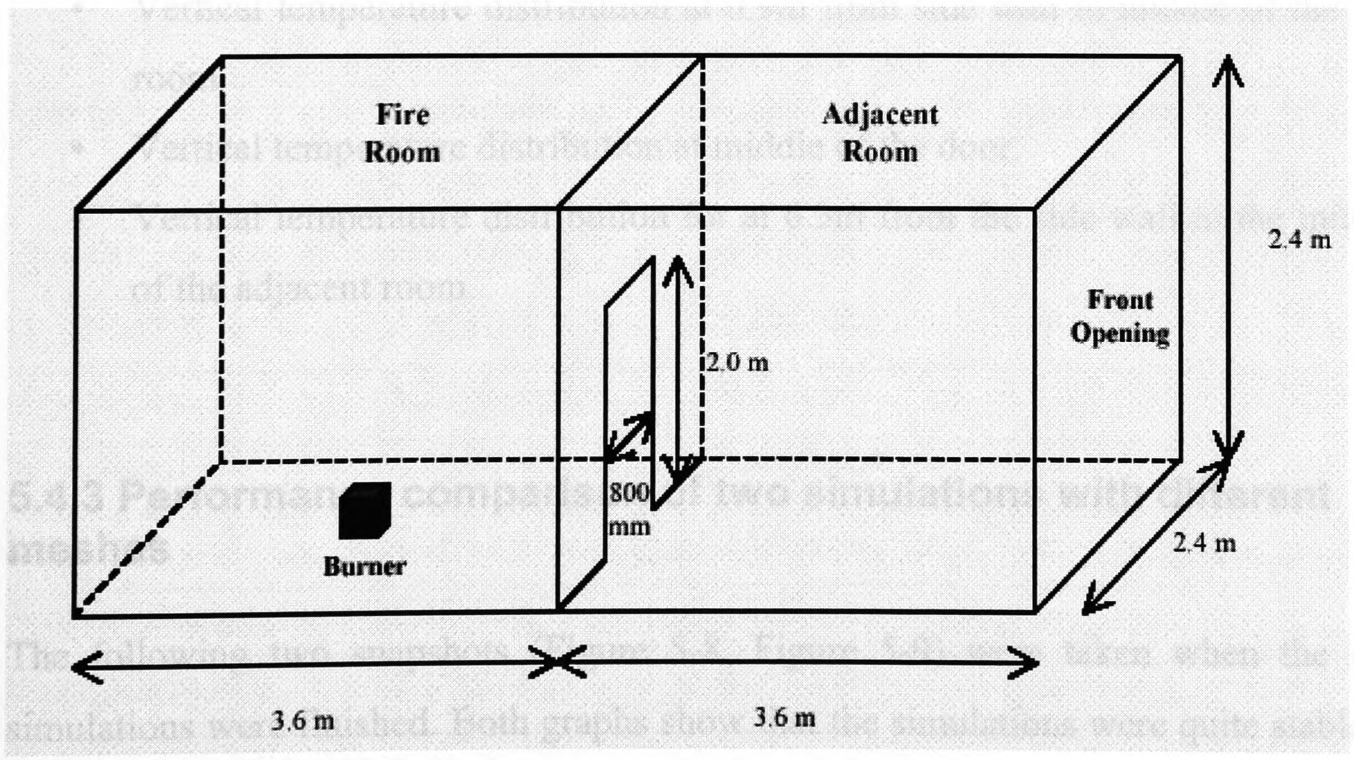


Figure 5-7: Illustration of the Layout for the Two-Compartment Structure.

5.4.2 Model Setup and boundary conditions for the *SMARTFIRE* simulations

Two simulations for the two rooms experiment were carried out. One (simulation a) used normal recommended mesh budgets of 56784 cells (i.e. 84*26*26); the second one (simulation b) used optimised local refined mesh budgets 31027 cells (i.e. 71*19*23) (from a coarse mesh). The specification was as follows:

- Flow model used. Combustion is disabled for simplicity. Radiation active.
- Initial temperature is 288.15 K.
- Wall material set to Gypsum Plaster.
- The total heat source is 160 kW.
- 60 time steps of duration 1 second with a maximum of 50 sweeps per time step.

The residual graphs were taken from the end of the two simulations to compare the comparative simulation stability and accuracy, between the cases using the local refined mesh and the usual recommended mesh.

At the end of the simulations, the following simulation results were stored for the purpose of comparing the simulation results with the experimental data:

- Vertical temperature distribution at 0.9m from side wall in middle of the fire room.
- Vertical temperature distribution at middle of the door.
- Vertical temperature distribution for at 6.3m from the side wall in the middle of the adjacent room.

5.4.3 Performance comparison of two simulations with different meshes

The following two snapshots (Figure 5-8, Figure 5-9) were taken when the two simulations were finished. Both graphs show that the simulations were quite stable as the trend of the residuals is reasonably downwards and the accuracies are similar. In order to examine the residual errors more closely, a table was created (Table 5-1) to compare each variable's residuals separately. It can be seen from the Table 5-1 that, for the residual of all of the important variables, a slight improvement (decrease) has been made for the case that was meshed using optimised local refinement. The differences between the convergences were not large. However, this initial result is very encouraging as the local refined mesh had only 54 percent of the overall cell budget of the normal recommended mesh and achieved the same/or slightly better performance in terms of simulation stability and accuracy.

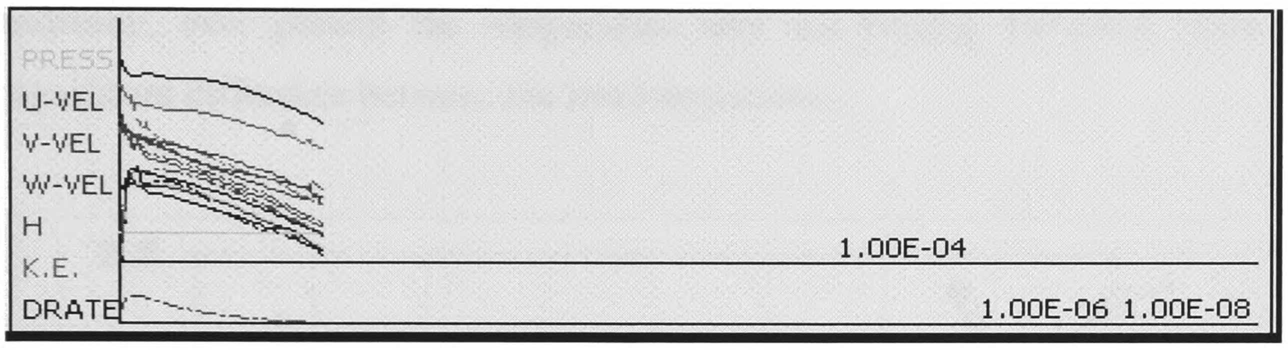


Figure 5-8: Residual graph at the end of simulation (a) with normal mesh

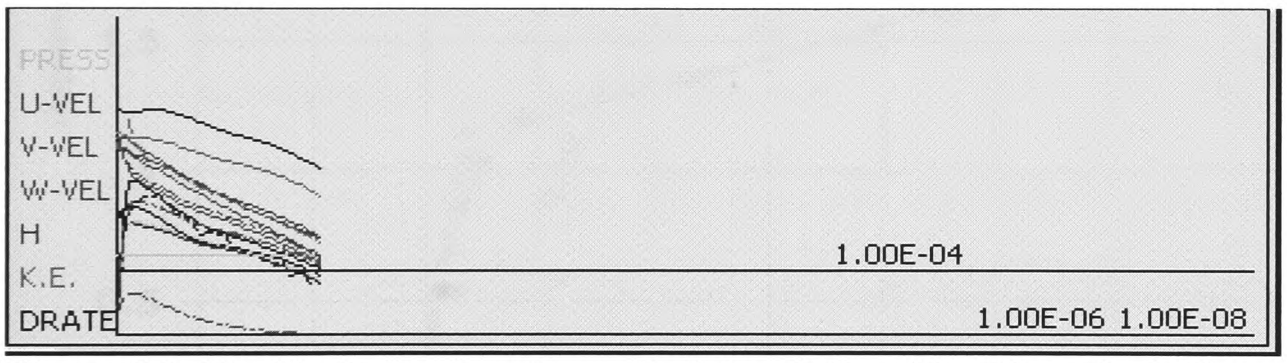


Figure 5-9: Residual graph at the end of simulation (b) with optimised mesh

Variable	Simulation a Residuals	Simulation b Residuals	Improved
Pressure	3.230E-04	1.776E-04	√
U_velocity	1.668E-03	3.382E-04	√
V_velocity	1.256E-03	3.661E-04	√
W_velocity	1.021E-03	1.790E-04	√
Enthalpy	4.104E-04	1.369E-04	√
Kinetic_energy	8.414E-03	1.563E-03	√
Dissipation_rate	2.150E-02	4.365E-03	√

Table 5-1: Comparison of residuals between simulations (a) with normal mesh and (b) with optimised mesh at the end of simulations for the two compartment experiment.

5.4.4 Temperatures near fire

The temperatures near the fire, in the fire room, are depicted in Figure 5-10. The results from the experimental data are significantly different from the two simulations. Both simulations predict constant temperature profiles for the lower layer and consistent with what is observed experimentally. The two simulations predict temperatures near ambient and both appear to under-estimate the lower layer temperature. The two simulations, both of which include the effects of thermal radiation, over predict the temperature near the ceiling. However there is little significant difference between the two simulations.

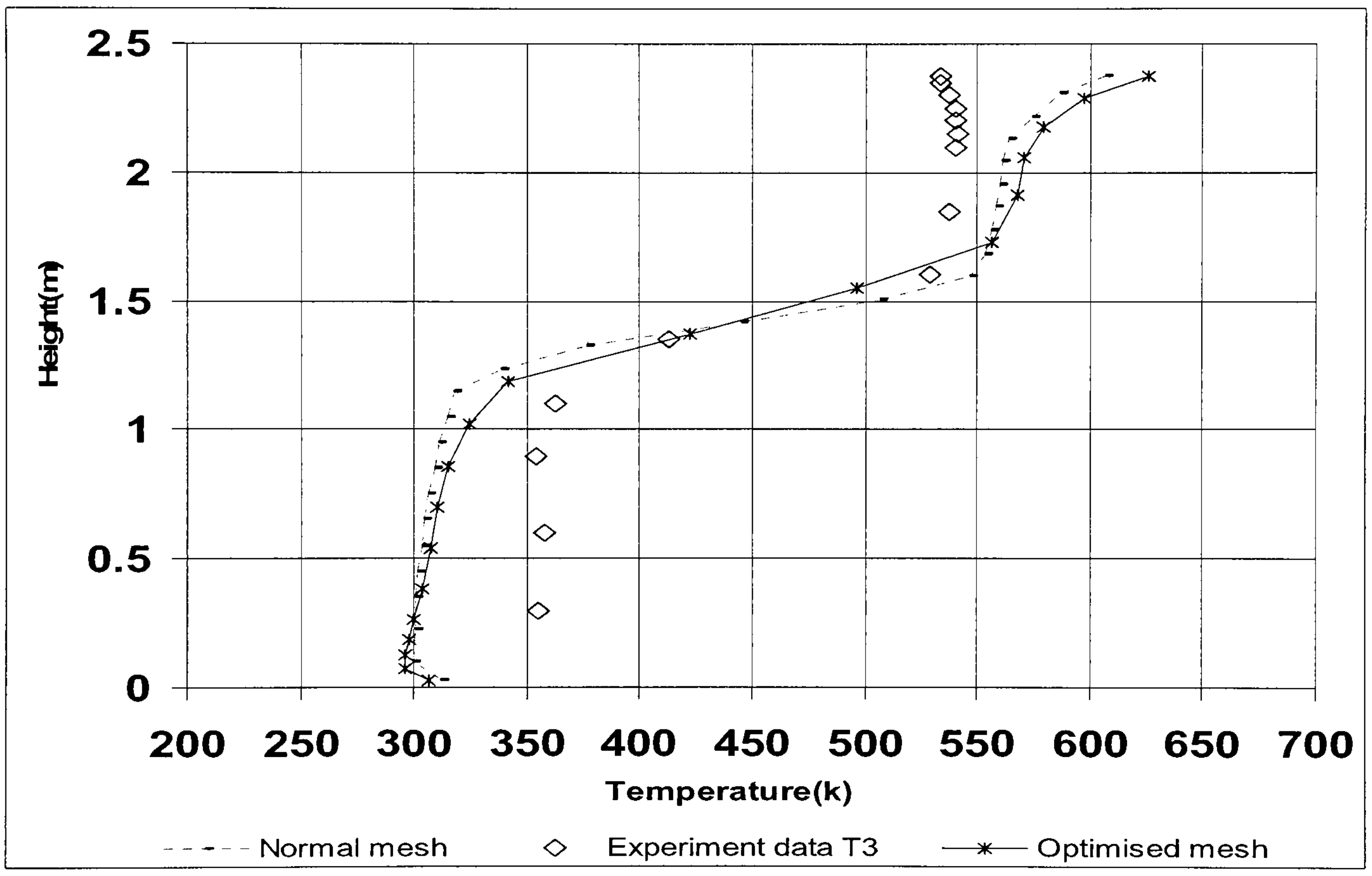


Figure 5-10: vertical temperature profile at 0.9m from side wall in middle of the fire room for two room experiments.

5.4.5 Doorway temperatures

Figure 5-11 shows the vertical temperature profile in the middle of the doorway. The temperature profiles for both simulations in the middle of the doorway shows very good agreement with what is observed for the experimental profile.

Temperatures in the simulations are also very similar to the experimental results, with little difference between the two simulations. Both simulations overestimate the temperature near the ceiling.

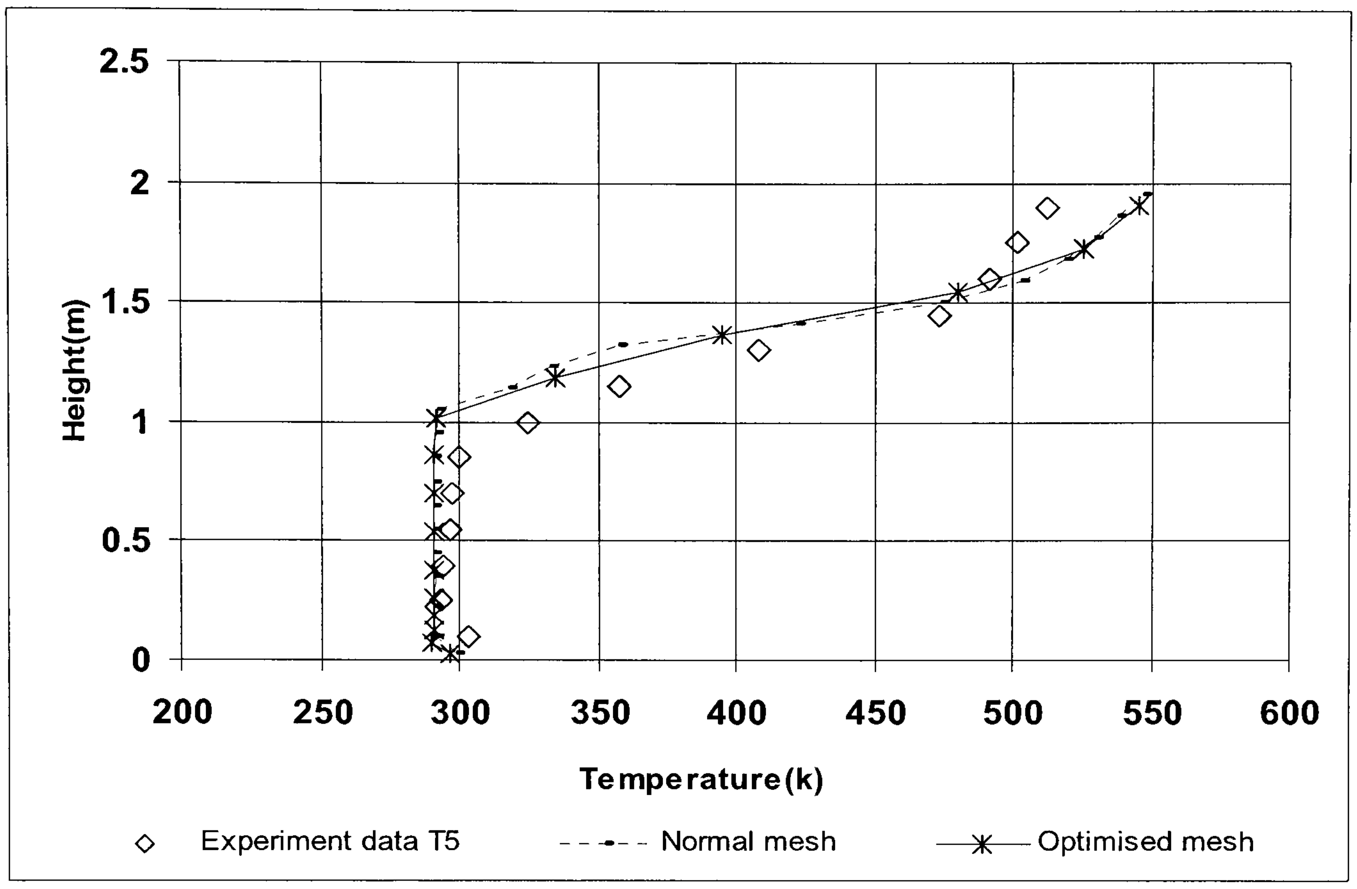


Figure 5-11: Vertical temperature profile at middle of the door for two room experiments.

5.4.6 Temperatures in the adjacent room

The vertical temperature profile in the middle of the adjacent room is presented in Figure 5-12. The simulated temperature profiles of the lower layer show a constant temperature with height, similar with what is observed for the experimental profiles. The predicted temperatures are in good agreement with the experimental data compared to the near fire and door way temperature predictions. The results produced by the two simulations are almost identical.

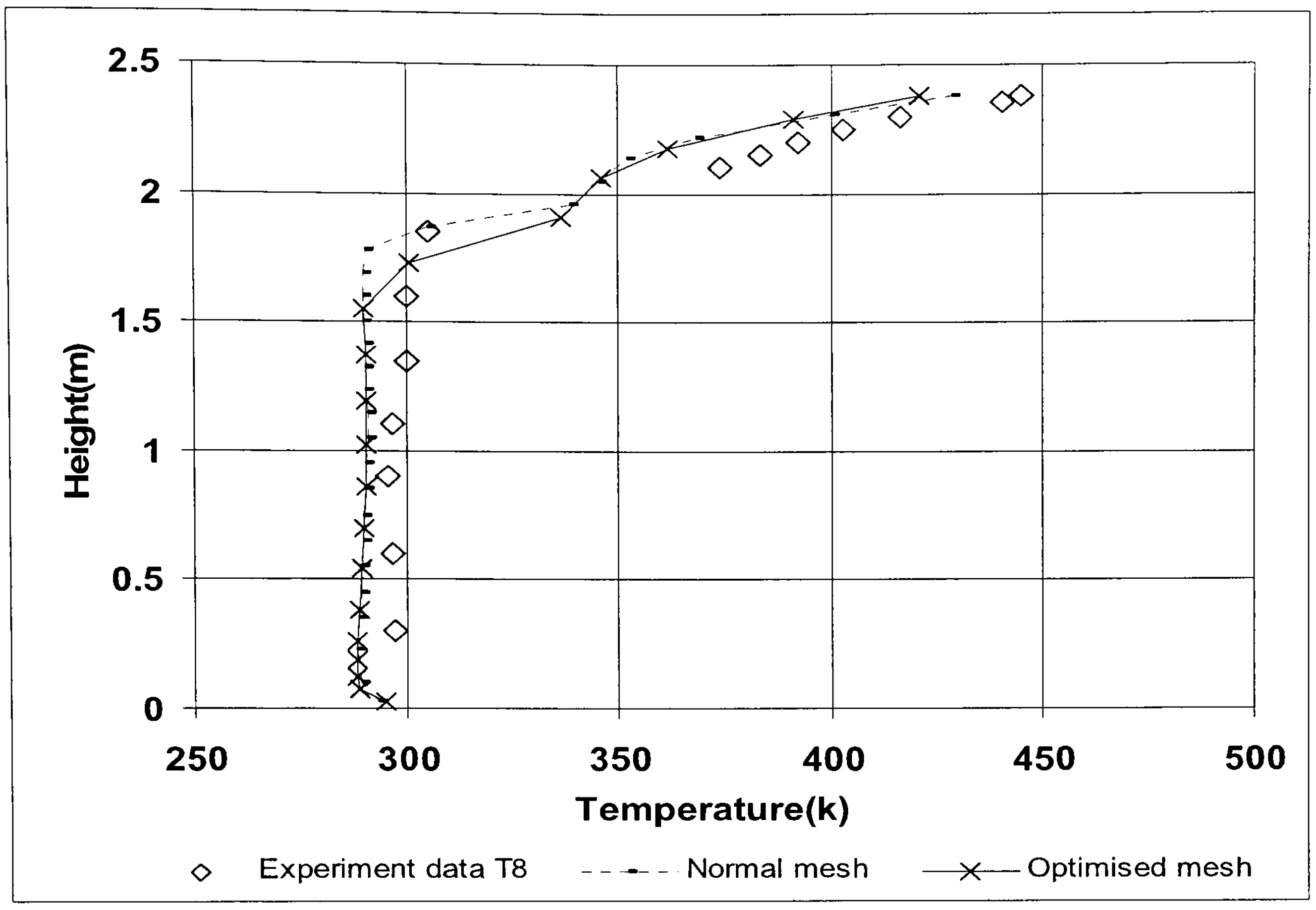


Figure 5-12: Vertical temperature profile for at 6.3m from the side wall in the middle of the adjacent room for two room experiments.

5.4.7 Remarks

The comparison of the results from two simulations with the experimental data resulted in these preliminary conclusions.

- Assessment using a coarse mesh solution can provide improved set-up and meshing.
- Local adaptive refinement procedures enhance the quality of the knowledge based mesh and thus can improve the solution performance.
- Results (from a single study) are no worse than for un-adapted cases and have benefits of time savings from the reduced mesh budget.
- Knowledge based local adaptive refinement methods provide a good learning tool for novice users.

[Chapter summary]

The proposed local adaptive mesh refinement method is based on adaptive techniques more commonly used in finite element analysis. The nature of the methods described in this chapter is that they require that the initial meshing solution is of sufficiently high quality to begin with. This justifies the earlier work (described in the previous chapter) which focussed on generating a high quality initial base-mesh. The investigation, described in this chapter, then attempts to maximize the efficiency and effectiveness of the meshing solution prior to conducting a production run. First of all, the meshing procedure started with an initial base mesh, which was optimised to remove poor aspect ration cells. This initial meshing stage was followed by a simple posterior nodal error estimation, which was computed using a CFD solver, to indicate the effective error distribution across the computation domain. The regions with higher percentage of nodal errors, above a prescribed tolerance, indicate if there is a need for a more refined mesh in a particular region. More cells are then introduced into such problematic regions and the mesh cells are redistributed to ensure that the cells maintain their quality. Again nodal errors are estimated for the refined mesh, as computed by the solver, and if there are still large errors, the refinement procedure is repeated. This quality test and refinement procedure is iterative. If the resulting mesh found to be overly distorted, between any two refinements, then the block-wised mesh justification procedure will be applied automatically. Finally, global error estimation is evaluated by summing all the nodal error contributions in a particular norm. The global error estimate is used in a termination test for the adaptive procedure. The chapter demonstrates the local refinement procedure on a simple two apartment case. These tests show improved simulation performance in terms of the simulation stability and accuracy, and also give benefits of time savings due to the fact that using

optimised local refinement method produced a high quality meshing solution, but with a significantly reduced mesh budget.

6 Prototyped Experiment Engine and initial test results

[Chapter Overview]

This chapter describes how all of the novel technologies investigated in this research, were integrated into an architectural framework and tested. This framework has been instantiated as a module of the *SMARTFIRE* FFM environment, in order to allow it to perform "trial and error" based searches of such factors as: appropriate case specification set-up, meshing and selection of control parameters – in order to increase the success rate of running the simulations. It is argued that this form of selected "trial and error" search is similar to the approach adopted by a human FFM Expert who has to run a new and unknown scenario in the minimum time. This novel framework includes a newly developed component called the Experiment Engine which links the existing Case Specification GUI and the Solver component, seamlessly. The EE is able to generate run-time experiments – using improved configurations and error/convergence information yielded from previous configurations – until a satisfactory configuration is found or all of the improvement options are exhausted. The prototype Experiment Engine (EE) has been designed to operate fully automatically and has proved to be very robust. The EE incorporates the local mesh refinement technique described earlier and the solution status assessment methods that were developed for another experimental solution control tool – called the Intelligent Control System (ICS was developed at the University of Greenwich). The framework has been tested on selected simulation scenarios and the initial test results show the validity of the local block-wise refinement regime and the effectiveness of the Experiment Engine concept.

6.1 Introduction

In order to support FFM software users, especially those who lack detailed CFD knowledge, but nevertheless have to use or make decisions based on FFM (e.g. Consultants designing buildings who may have limited CFD knowledge or regulators presented with CFD modelling results who have to approve designs but do not know if the model was run appropriately), a fully automatic, robust Experiment Engine integrated into the *SMARTFIRE* solution process has been developed.

For complex fire cases, even Expert users cannot always run the FFM simulation successfully in the first attempt. The premise behind the operation of the Experiment Engine (EE) is that an Expert user has the ability to construct simplified trial runs which can be used to get a feel-for and gain insight into the specific case to be modelled. The outputs of the trial runs can then be used to appropriately set up the cases, to fine tune the meshing, and hence to get the required solution. The EE attempts to emulate an Expert user's ability to be able to set up and take appropriate control actions as and when necessary, and to recover from any simulation failure. This process involves both an experimental and a production phase. In the experimental phase, the original problem is simplified in order to keep the overheads of performing experiments to a bare minimum. Once the experiments are complete, the mesh is adaptively refined (based on solution error norms) and control parameters are changed, if the solution had not properly converged, until the results are found to converge to a prescribed accuracy.

These initial Expert activities, during a simulation, can be abstracted as trial and error search guided by expertise, with feed back from the solution for the appropriate set up, meshing and control parameters for the given problem. This is the key strategy which has influenced the development of the Experiment Engine.

This chapter begins by outlining some of the features on trial and error techniques and their applicability to FFM is briefly discussed. The next section will describe the existing *SMARTFIRE* architecture and identify the need for the Experiment Engine to perform trial and error search for appropriate set up, meshing and control parameters. Secondly, the new *SMARTFIRE* solution process is presented. Thirdly, details about "trial" constructions and "error" assessments are presented. Finally, some initial test studies have been conducted to illustrate and demonstrate the benefits of using the Experiment Engine.

6.2 Trial and error search

Trial and error is a simple method for obtaining knowledge, both in terms of propositional knowledge and know-how. In trial and error searching, one tries an option to see if it works. If it works, then this is considered to be a solution. If it does not work – i.e. there is an error – then one tries another option.

Trial and error searches have a number of features:

- Solution-oriented: trial and error makes no attempt to discover why a solution works; merely that it is a solution.
- Problem-specific: trial and error makes no attempt to generalize a solution to other problems.
- Non-optimal: trial and error is an attempt to find "a solution", not "all solutions", and not "the best solution".
- Needs little knowledge: a trial and error search can proceed where there is little or no knowledge of the subject.

These features of trial and error searches are very similar to the problems facing FFM. First of all, the purpose of the FFM simulation is solution-oriented. E.g. it is not necessary to know why and how the configuration worked for the given problem, since only the solution itself (and its quality) actually matters. Secondly, because FFM simulations are problem dependant in nature, it is impossible to generalize a configuration which would be suitable for all scenarios. Thirdly, at the configuration stage of the simulation, the reliability of the simulation is more important than the performance of the simulation, although the Expert is able to significantly improve the performance by justifying the control parameters at some later stage of the simulation. Finally, because of the complexity of the CFD modeling involved, even the Experts may not always be certain whether the configuration applied would have the desired effect. In a "plain" trial and error search, options are simply tried at random and it is neither optimal nor infallible, however, with the expertise guided trial and error search, the option that is a priori viewed as the most likely one is tried first (for example, our earlier efforts about case classification and case recognition can be used to work out the a priori mesh), followed by the next most likely (e.g. the local mesh refinement algorithms is used to work out the next most likely mesh), and so on until a solution is found. The embedded expertise makes the trial and error search more efficient and the validity of the local mesh refinement algorithms will help to ensure the robustness of such a strategy, thus the Experts' activities in the initial stages of the simulation are closely modeled.

6.3 The existing SMARTFIRE architecture

SMARTFIRE is an open architecture, interactive CFD code with integrated Knowledge Based System (KBS) components that attempt to make fire field modelling accessible to non-Experts users.

SMARTFIRE is comprised of the following components:

- Case Specification Environment: An advanced friendly and supportive Graphical User Interface (GUI) that allows users to enter a geometry, to change the physics options and the simulation strategy to be used for the scenario being modelled. There is also an automated meshing system with a built-in manual mesh editing facility that is embedded within the Case Specification Environment. This interactive meshing tool enables the user to create a suitable mesh for a simulation that will be run in the *SMARTFIRE* CFD Engine (also known as the Solver). It should be noted that novice users do not generally need to change most of these controls or the automated generated mesh, since many of the parameters including meshing parameters have been assigned sensible default values or are chosen automatically. However, for more complex cases, the default parameters may not be appropriate. An automatically generated mesh may not be adequate and can seriously affect the accuracy of the final results. In the worst case, it may be impossible to obtain the final results as the simulation diverges or the solution crashes.

- The interactive CFD engine (or Solver): The numerical engine of the *SMARTFIRE* system has an integrated "run-time" User Interface that provides direct access to many of the underlying solution controls. Furthermore, there is support for planar data visualisation of the solution variables and errors (using a variety of different display techniques) and support for exploring the data both visually and numerically. These display and monitoring facilities use the most up-to-date versions of the solution data at any stage of the simulation. A user has full control overall the simulation process and can modify all control parameters.

These two components are implemented as separate modules (i.e. they are two separate processes, when *SMARTFIRE* is operating). This helps to provide flexibility and modularity in order to maximise the ease of the software management and maintenance. For example, any changes of one component will have minimum impact on the other component. However, the communication between the two components is one way from the GUI to the Solver (i.e. the only interaction between the GUI and solver is the output files from the GUI - the problem specification and other configurations defined in the GUI are stored in several files, which represent the physical problem definition and the mesh. The CFD engine takes these files as inputs to initialize and perform the computation. In a sense, the CFD engine knows nothing of the GUI since it is merely an input device. Moreover, the GUI and solver completely lose track of each other at run time and configuration changes in either module are not reflected in the other module.

In order to achieve the trial and error based search (by the mean of performing the adaptive procedure) it was proposed, first of all, that the communication between the components had to be bidirectional, because the GUI needs to know if an "error" occurs in the solver, and to then construct a new "trial" if necessary. Secondly, the components have to understand each other and keep track of each other at run-time. One approach to do that is to make the interaction between the two components directly, which means that the GUI has to able to communicate with the solver at run-time. Obviously, this requires the modification of the existing code, because the components were not designed to have direct run-time communication. Although it is quite possible to construct a solver that can interact directly with the geometry based problem specification, the modification of an existing solver may require the introduction of new data structures etc. (i.e. the run- time modification of the mesh by the adaptive procedure will cause all sorts of problems) thus forcing an extensive rewrite of the code. The expense and time required to do this is large and in most cases considered prohibitive, particularly for well established codes such as *SMARTFIRE*.

The alternative approach that was used with the existing *SMARTFIRE* code, which is the focus of this chapter, is to leave the existing components unaltered and to add a new component which is responsible for controlling the flow of information between the two modules and also acts as a translator between them. As a result, the Experiment Engine (EE) was developed to link the GUI and the Solver seamlessly and is responsible for communication between the two components, for example, when there is something wrong with the simulation state in the Solver, then this information is first passed to the Experiment Engine where the information is processed, if it is necessary to alter the case configuration, then the Experiment Engine will issue instructions for the GUI to try an "improved" case configuration (including a change of the mesh or/and control parameters) to see if that will solve the problem.

The actions taken by the GUI, in carrying out an instruction from the EE to improve the mesh, will follow the exact procedure that was developed earlier for local adaptive mesh refinement. Since this topic has been covered fully in the previous chapter, it will not be repeated here. Problems with the simulation state are determined using the solution state assessment techniques that were developed for the ICS. In the next section, the EE solution state assessment techniques and control parameter adjustment are discussed briefly.

6.4 Assessment of the solution state and control parameters adjustment

Having the "correct" problem set-up and adequate meshing does not guarantee that a required solution quality will be obtained, since control parameters also play a key role in obtaining a desired solution. Janes's ICS[Janes-02], using heuristic search techniques to find the almost-best set of control parameters through a dedicated evaluation function (which is able to assess solution state, e.g. convergence problems) which assesses the residual graphs before and after changes of control parameters are made.

Janes concluded that, based on the residual graphs, an Expert was able to differentiate between acceptable and unacceptable solution states and the control decisions can be based almost exclusively on the results of the residual graph assessment.

Typical residual graph features are shown in Figure 6-1.

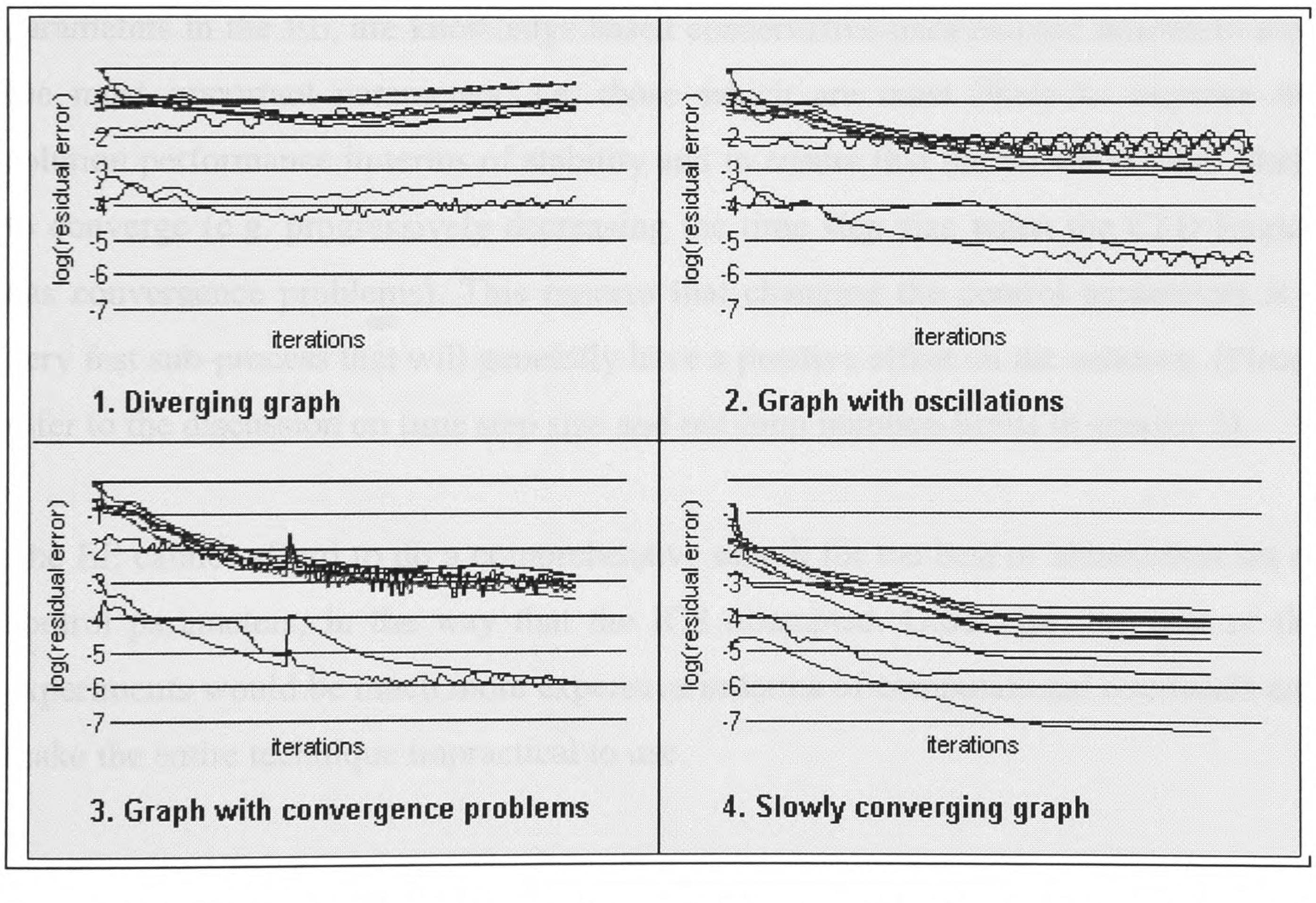


Figure 6-1: Example residual graphs reflecting specific simulation problems (as described in Janes Phd thesis)

Assessment of the solution state and the process justifying the control parameters has been implemented as a sub-process (a simplified version of ICS) that is embedded in the EE. Assessment of the solution state in the EE is derived from the method used in the ICS and extends it to include assessment of the validity of values of only those variables which are deemed to be most important to a CFD Expert (i.e. flame temperature should not exceed 1300°C, any temperature higher than 2000°C is incorrect in normal conditions [kumar-01]). This assessment also considers other fatal error indicators. The additions to the assessment method enable the EE to monitor the solution process thoroughly and to be able to detect any abnormal and/or fatal solution status and thus, to make any necessary changes to correct the problem. For example, in the event that a fatal error occurs, a new test case can be formed with a combination of suitable control parameters, mesh settings and problem set-up changes, (i.e. automatically selecting the best modes of operation in the further version of the EE, as and when the conclusive knowledge becomes available.)

Obtaining appropriate control parameters is one of the key outputs from the experimental process. However, unlike the ICS, the changes made to the control parameters in the EE, are knowledge based conservative ones and are limited to only the most important parameters, i.e. those which are most likely to improve the solution performance in terms of stability and to ensure that the Solver is more likely to converge (e.g. progressively decreasing the time step size when the CFD Engine has convergence problems). This ensures that changing the control parameters is a very fast sub-process that will generally have a positive effect on the solution. (Please refer to the discussion on time step size and out-loop iteration limits in chapter 3).

The EE cannot afford to do a comprehensive search for the best or almost-best set of control parameters, in the way that the ICS attempted. Otherwise, the cost of the experiments would be much more expensive in terms of computational overheads and make the entire technique impractical to use.

The purpose of the control parameter change, in the EE process, is mainly used to maintain a valid solution status rather than trying to find the optimal control parameters. The prime task of the EE is to select, apply and maintain reasonable set-ups, meshing and control parameter selections and to make these "decisions" as quickly as possible. Hence the EE is intended to give a much better chance for effective use of other optimization strategies (e.g. the ICS and/or Group Solver techniques – which also depend on having a high quality mesh) to succeed in the production run rather than in the experimental phase. This is mostly due to the fact that there is much more confidence in the set-ups and meshing produced by the EE, and the control parameters should be in a suitable range after the experiments.

In this way, the EE is not considered to be a substitute for techniques such as the ICS and/or Group Solvers [Ewer-99b] [Hurst-04], but rather, the EE is a management tool which decides when to deploy the ICS and/or Group Solver techniques to further optimise the solution process and to make sure that the ICS and Group Solvers work under their own optimal set of conditions – with full fault tolerance if there are problems.

In summary, the "adjustment of control parameters" sub-process is not intended to find a perfect set of control parameters. However it is necessary for the EE to be able to monitor the solution process and to make necessary control changes when there are convergence problems. In this way, it will help the whole EE process to produce proper set-ups, to use a suitable mesh, to form a production case with reasonable solution controls, and it is intended that users will eventually get the desired solution and solution quality with only minimal user intervention.

6.5 The new Experiment Engine integrated solution framework

The EE integrated *SMARTFIRE* solution process is an iterative process as illustrated in Figure 6-2.

The integrated EE framework links the original *SMARTFIRE* Case Specification Environment (GUI) and the *SMARTFIRE* Interactive CFD Engine components. These components can now communicate via data passed through a well defined message passing interface. In this way, it is possible to maintain the original Case Specification process and the CFD Solver process as independent but cooperating tasks. The two components interact with each other only when the EE is operating. This componentization has enabled rapid prototyping and the research and implementation of new development requirements as they have been envisaged.

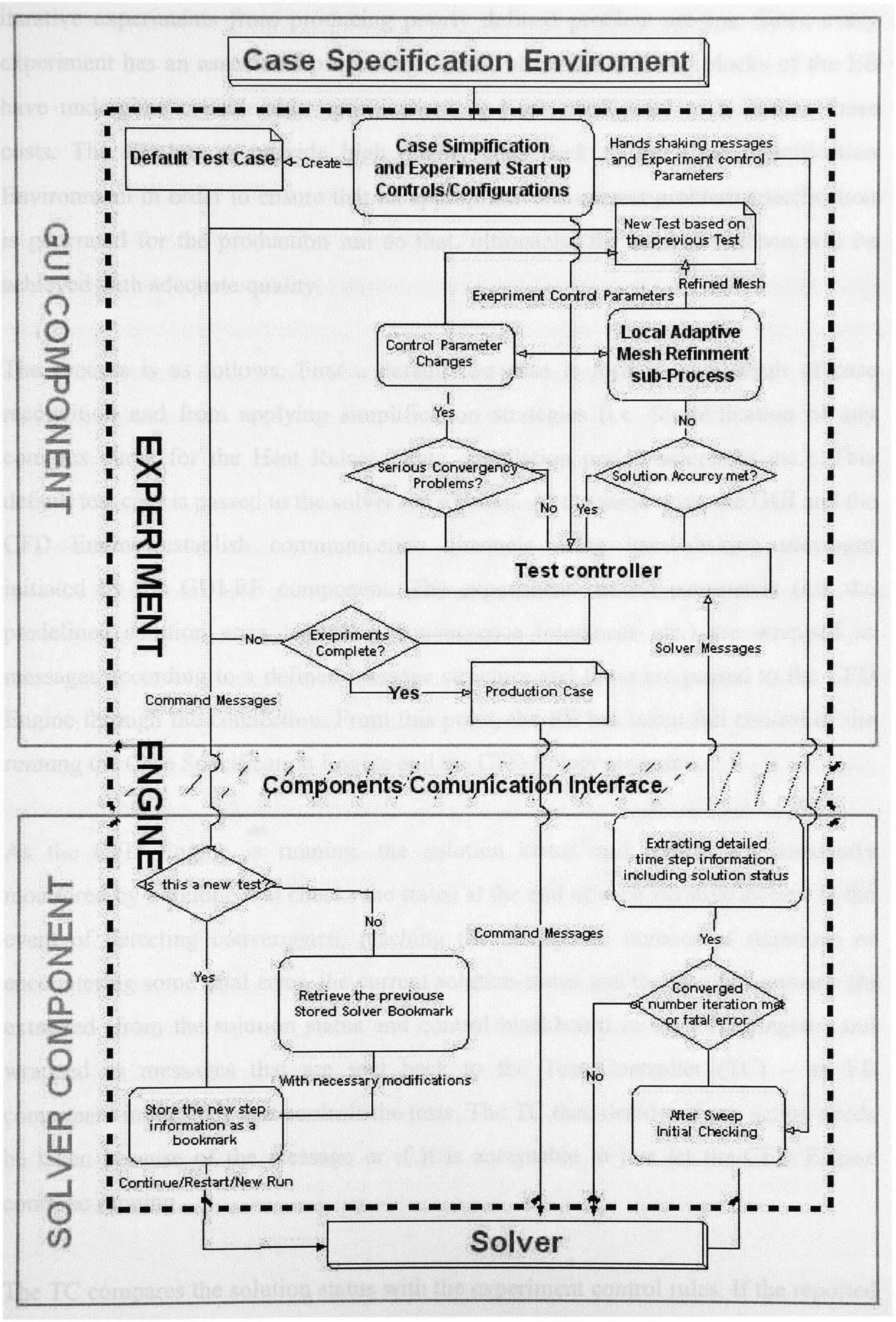

Case Specification Enviroment
Default Test Case
Create
Case Simplification and Experiment Start up Controls/Configurations
Hands shaking messages and Experiment control Parameters
New Test based on the previous Test
Refined Mesh
Exepriment Control Parameters
Control Parameter Changes
Local Adaptive Mesh Refinment sub-Process
Yes
No
Serious Convergency Problems?
Solution Accury met?
No
Yes
Test controller
No
Exepriments Complete?
Solver Messages
Yes
Production Case
Command Messages
GUI COMPONENT
EXPERIMENT ENGINE
Components Comunication Interface
Extracting detailed time step information including solution status
Is this a new test?
No
Command Messages
Yes
Yes
Retrieve the previouse Stored Solver Bookmark
Converged or number iteration met or fatal error
No
With necessary modifications
Store the new step information as a bookmark
After Sweep Initial Checking
Continue/Restart/New Run
SOLVER COMPONENT
Solver

Figure 6-2: Overview of Integrated Experiment Engine Solution Process

The EE architecture and mode of operation, has been carefully planned to avoid the complete failure of the CFD process. Various techniques are used to protect the iterative experiments from producing poorly defined problem set-ups. Since every experiment has an associated processing "cost", all of the building blocks of the EE have undergone considerable optimisations and are configured to minimize these costs. The EE has to provide high quality feed back to the Case Specification Environment in order to ensure that an appropriate and correct problem specification is generated for the production run so that, ultimately, the correct solution will be achieved with adequate quality.

The process is as follows. First a default test case is formed as a result of case recognition and from applying simplification strategies (i.e. Simplification of any complex curve for the Heat Release Rate, simulation period selections etc.). This default test case is passed to the solver for solution. At the same time, the GUI and the CFD Engine establish communication channels using hand-shaking messages initiated by the GUI-EE component. The experiment control parameters (i.e. the predefined solution error tolerance, convergence tolerances etc.) are wrapped as messages according to a defined message structure and these are passed to the CFD Engine through the connection. From this point, the EE has taken full control of the running the Case Specification Engine and the CFD Solver programs.

As the CFD Engine is running, the solution status and results are constantly monitored by a routine that checks the status at the end of each iterative sweep. In the event of detecting convergence, reaching the configured number of iterations or encountering some fatal error, the current solution status and the results summary are extracted (from the solution status and control blackboard in the CFD Engine) and wrapped as messages that are sent back to the Test Controller (TC) – an EE component in the GUI that controls the tests. The TC then decides if any action needs be taken because of the message or if it is acceptable to just let the CFD Engine continue running.

The TC compares the solution status with the experiment control rules. If the reported error is bigger than the predefined solution error tolerance, then the TC will activate the local mesh refinement sub-process. The result of this action will lead to a refinement of the current mesh according to the error distribution that was found on the previous unrefined mesh. This localized mesh refinement sub-process is the same implementation of the local adaptive mesh refinement method that was discussed in chapter 5.

In the event of mesh refinement taking place, the TC will construct a new test which consists of the newly refined mesh and any required changes to the solution control parameters (N.B. only the time step size is controlled at this time although it is completely possible for more complex control strategies to be implemented). This revision of the solution control depends on the quality assessment of the current solver status information and other critical variable values (e.g. pressure, velocity values, residuals, etc.)

If the TC decides that it is not necessary to refine the mesh, then it is possible to retrieve the previously stored restart files to make other necessary modifications (i.e. to apply a new time step size). So the results of the previous time step are used as a starting point for the re-run. In this way, it is not necessary to re-run the test from the start, hence, the EE is able to keep the cost of the experiments to a minimum.

When the CFD Engine starts to solve a new time step, a bookmark is saved. This will be used if the experiments need to be restarted from this point – due to subsequent poor solution performance or solution failure.

At the end of this iterative trial, the TC will check to determine that no more changes are deemed to be necessary, and the trial will be assumed to be successful at the prescribed end-simulation time. The set-up, used by production-run, is formed according to the original problem specification and the set-up, mesh and control parameters of the last successful EE test case. The production run can be executed automatically or the user can make further changes if they so wish. Finally the control of the simulation processing is handed back to the user in the CFD Engine.

6.6 *Initial tests and discussion of the results*

The first test case presents the results from using a moderately complex geometry with two fire sources, to examine the validity and robustness of the local block-wise mesh refinement procedure. The second study uses the EE to simulate a fire case that has high speed flows and a high output heat release rate. The second case has an extracting "inlet" that removes air from the corridor – representing a ventilation system (as shown in Figure 6-11) that extracts at a prescribed rate. The second case was seen to have convergence problems and is thus regarded as somewhat "unstable". Finally, the results are analysed and presented with initial conclusions on the effectiveness of the techniques.

It should be noted that, although these initial tests are all performed using Heat Release Rate models for the fire representation, there is no reason why a combustion model could not be used instead. The complexity and nature of the algorithms has no impact on the validity of the Experiment Engine monitoring and control regime. It is generally acknowledged that volumetric heat sources and heat release curves tend to produce less stable and less realistic temperatures than when using a combustion model. This makes it more important to ensure that the EE can work well with the less stable types of simulation scenario.

6.6.1 Validity and efficiency of the local block-wise refinement algorithm

In order to validate the efficiency and robustness of the local block-wise mesh refinement algorithm, a multiple room geometry (of size 5.0 m by 5.1 m by 5.0 m) was used – as shown in Figure 6-3. This has two fire sources with the same heat output, both activated at t=0 s. Both fires have growing Heat Release Rate (HRR) curves with peak heat outputs of 500.7 kW as shown in Figure 6-4. One fire is centrally located on the ground floor, and the second fire is on the first floor. Both floors have obstructions representing furniture such as chairs, a sofa and a bed (these items of furniture do not burn during the simulation period). There is a single vertical vent (a door of size 1.0 m by 2.0 m height) in the ground floor and a single vertical vent (a window of size 2.0 m by 1.0 m height) in the first floor. The two floors are connected by a staircase.

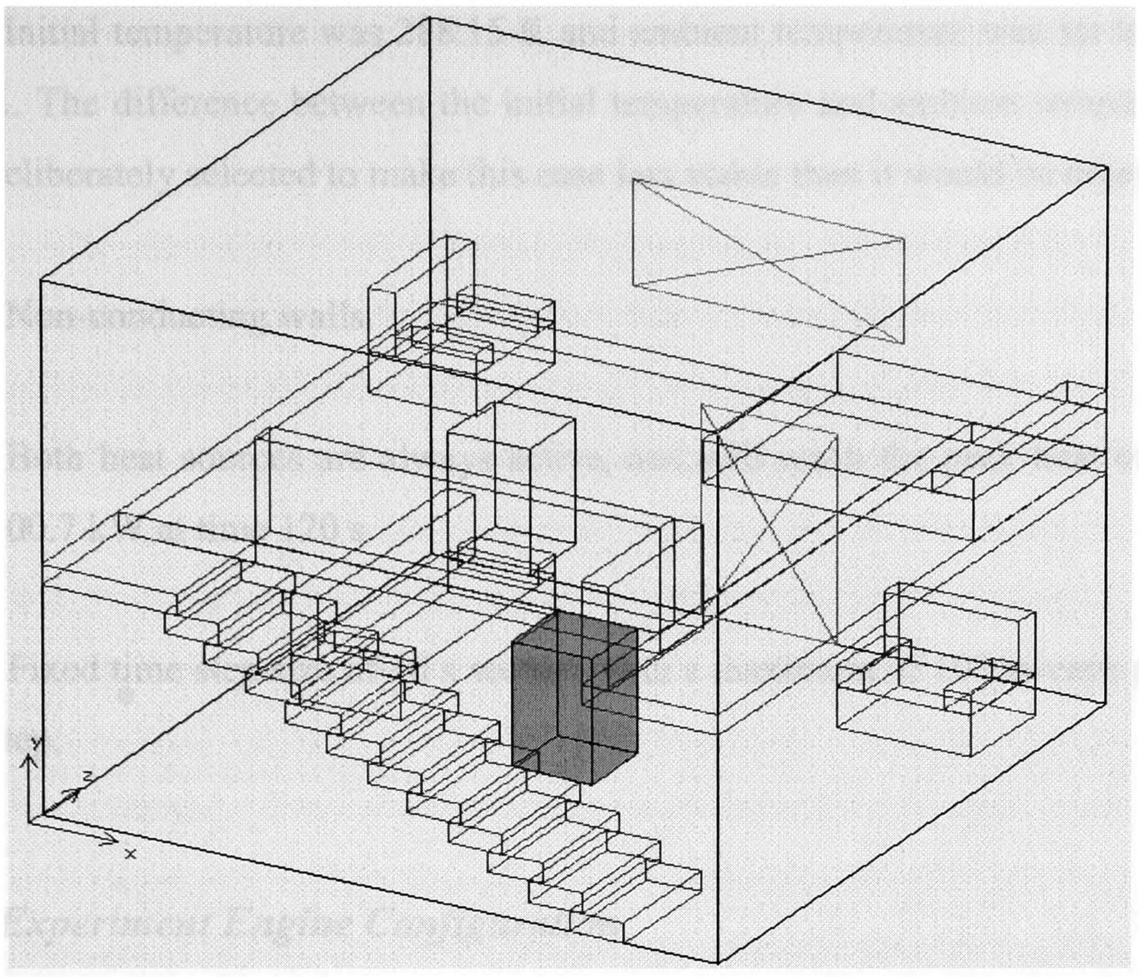


Figure 6-3: Geometry set-up for the multiple room complex case in the *SMARTFIRE*

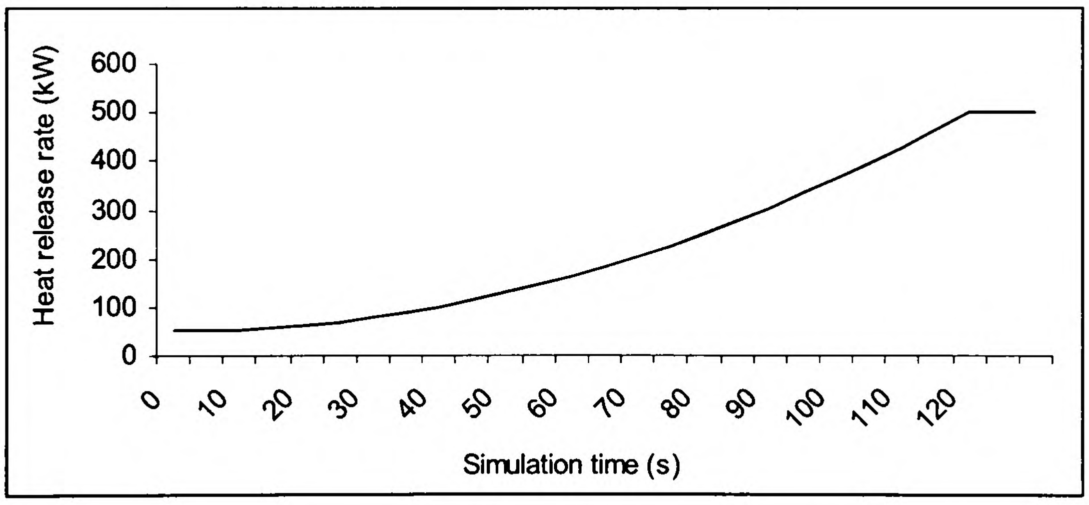


Figure 6-4: Heat release rate for both of the fire sources in the multiple room case.

6.6.1.1 Problem set up and boundary conditions

- Flow, heat transfer and radiation models are used. Combustion is disabled for simplicity – although it's use would not have affected the EE behaviour significantly.

- Initial temperature was 288.15 K and ambient temperature was set to 303.75 K. The difference between the initial temperature and ambient temperature is deliberately selected to make this case less stable than it would be otherwise.

- Non-conducting walls.

- Both heat sources are always active, and will reach the peak heat output of 500.7 kW at time 120 s.

- Fixed time step size of 10 s second, with a maximum of 100 sweeps per time step.

6.6.1.2 Experiment Engine Configuration

- Nodal error Tolerance is set to 1E -03.
- The Experiment Engine is activated from the beginning with an "Improved Mesh Quality" strategy, which means whenever the mesh became too distorted; the block-wise mesh justification procedure will be applied automatically.
- Each heat source is simplified so as produce an equivalent constant fire with a constant heat output of 120 KW.

Two sets of experiments (set A and set B) have been conducted using the EE. Each test in the experiment set only runs for a single time step that is simplified to be reasonably representative of the most unstable period of the simulation. New tests are then formed with progressively refined meshes until the maximum nodal error (continuity error) is below the predefined tolerance of 0.001.

6.6.1.3 Experiment set A starting with coarse initial base mesh

Experiment set A started with coarse initial base mesh of 25839 cells (i.e. 33*27*29), the mesh in Y direction is shown in Figure 6-5.

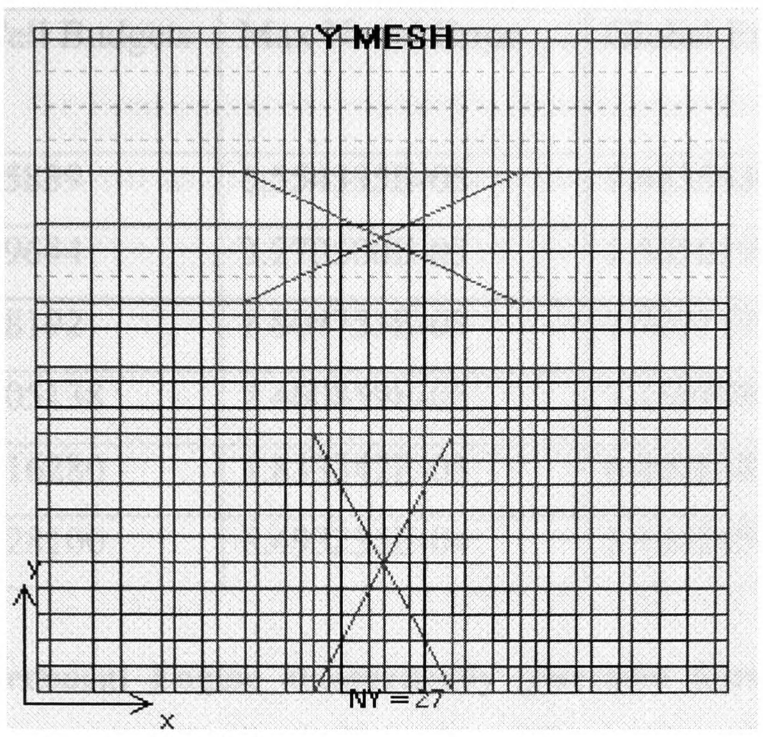


Figure 6-5: the coarse initial base mesh in Y direction for Experiment set A

After 6 times iteration of local refinement to satisfy the maximum nodal error tolerance of 1E-03, the final mesh was end up with a mesh budgets of 128100 cells (i.e. 60*61*35). The refined final mesh in Y direction is shown in Figure 6-6.

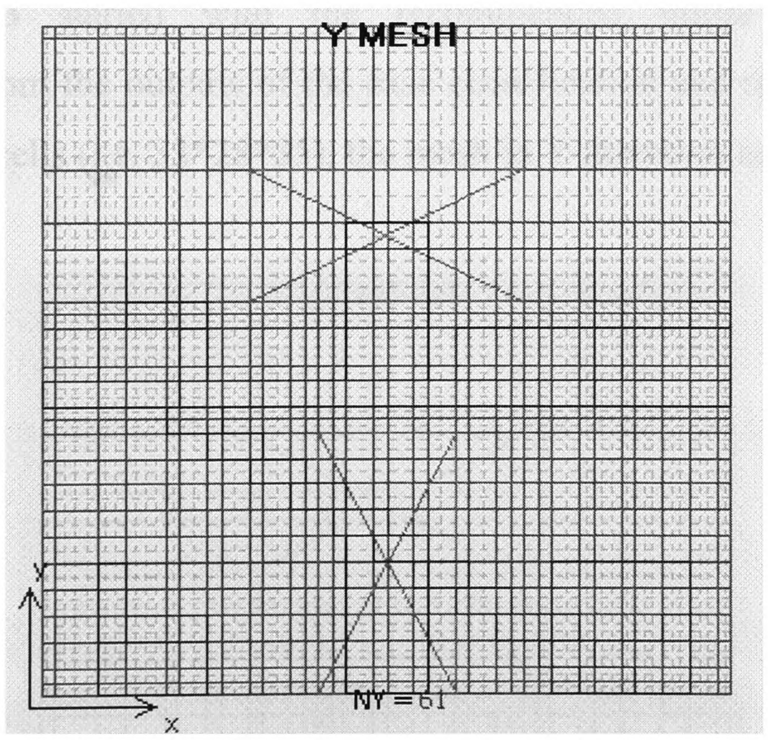


Figure 6-6: the refined final mesh in Y direction for Experiment set A

The detailed effects regarding the maximum continuity error by the mesh refinement procedure are shown in Table 6-1. The test number in Table 6-1 indicates the level of the local refinement, as after each local mesh refinement, the Experiment Engine generated a new test with the refined mesh automatically.

Test Number	Cell Budgets	Max Nodal Error	Global Error Indicator
1	25839	6.554335E-03	7.433563E-08
2	59644	3.270500E-03	1.565619E-09
3	88192	1.804753E-03	7.723117E-10
4	105138	2.460859E-03	7.159708E-10
5	116280	1.833342E-03	6.402814E-10
6	128100	8.493235E-04	5.104215E-10

Table 6-1: the Experiment Engine automatically generated tests with refined mesh based on continuity error distribution in Experiment Set A (started with a coarse initial base mesh).

6.6.1.4 Experiments Set B starting with recommended initial base mesh

Experiment set B started with the recommended initial base mesh (i.e. recommendation from the outputs of the case classification and recognition), the cell budget was 32340 cells (i.e. 33*28*35), the mesh in Y direction is shown in Figure 6-7.

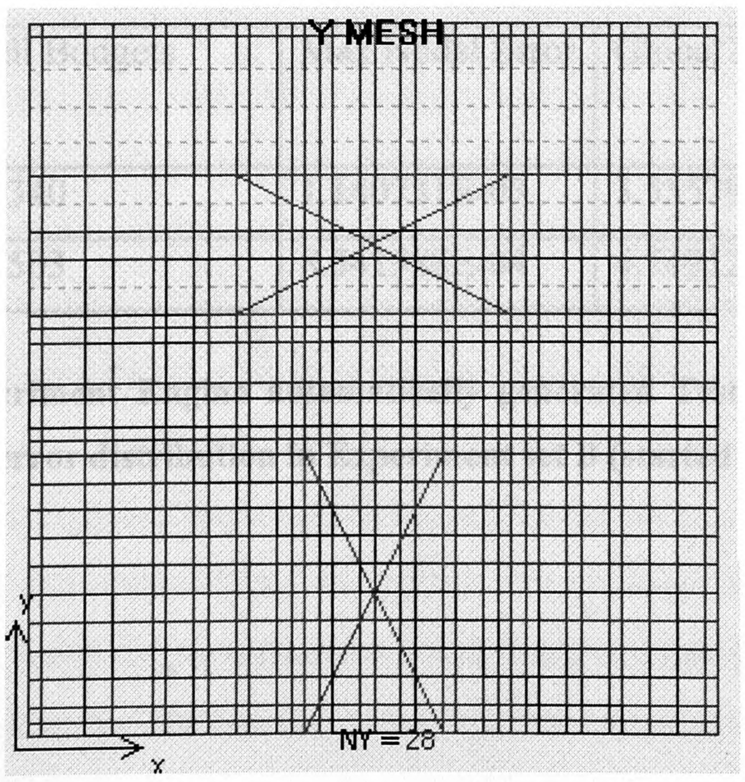


Figure 6-7: the recommended initial base mesh in Y direction for Experiment set B

After just two iterations of the local refinement to satisfy the maximum nodal error tolerance of 1E -03, the final mesh was end up with cell budgets of 52503 cells (i.e. 37*33*43). The refined final mesh in Y direction is shown in Figure 6-8.

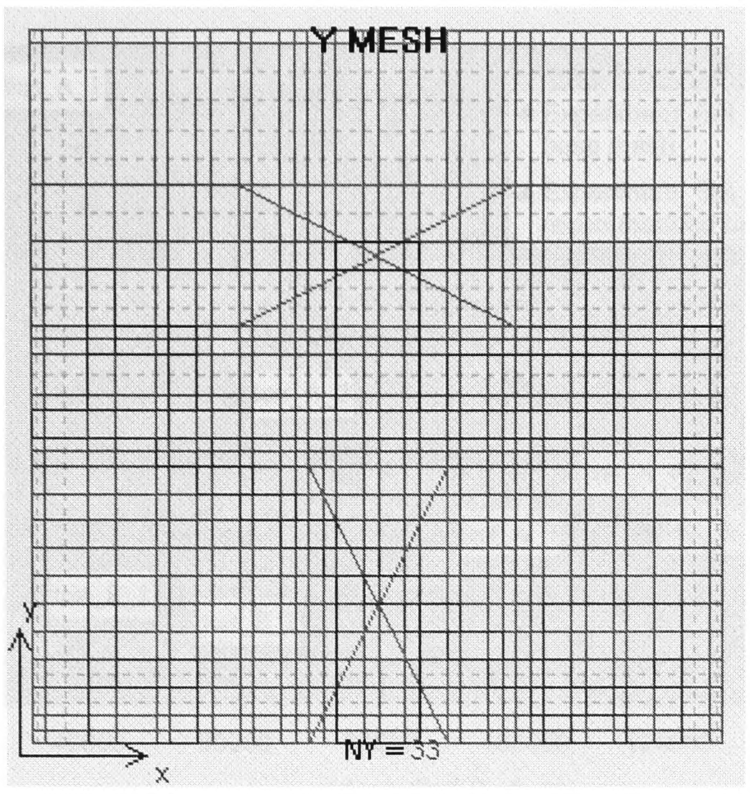


Figure 6-8: the refined final mesh in Y direction for Experiment set B.

The detailed effects regarding the maximum continuity error by the mesh refinement procedure are shown in Table 6-2.

Test Number	Cell Budgets	Max Nodal Error	Global Error Indicator
1	32340	1.140731E-03	3.555790E-09
2	52503	8.541988E-04	4.749122E-10

Table 6-2: the Experiment Engine automatically generated Tests with refined mesh based on continuity error distribution in Experiment set B (started with a recommended initial base mesh).

6.6.1.5 Comparison of the local mesh refinement performance on Experiment set A and B

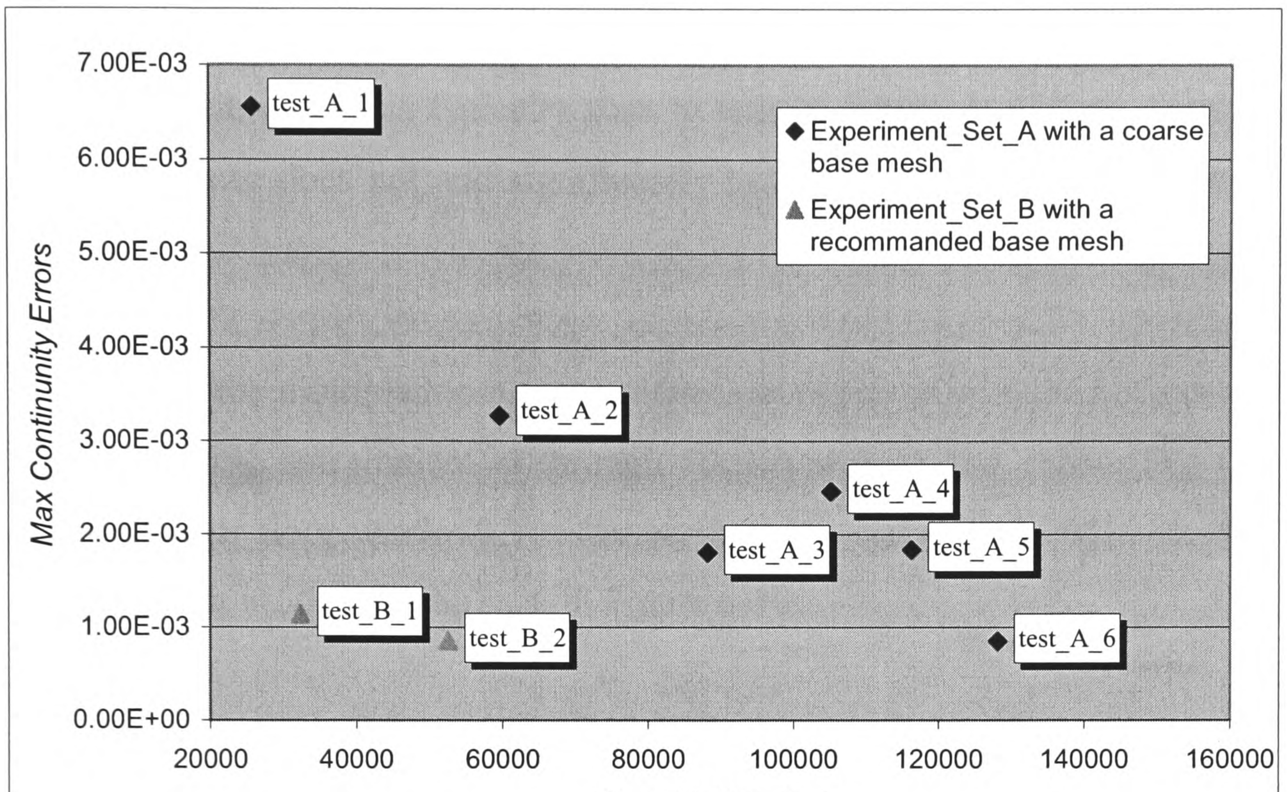


Figure 6-9: Comparsion of the local mesh refinement performance on Mulitple room complex case with different quality of the initial mesh

In both sets of the experiments, the local refinement procedure performed well in terms of achieving the goal of decreasing the maximum continuity error. The robustness of the local refinement procedure is shown by taking the coarsest initial base mesh (in A tests) and still being able to refine it until the prescribed error tolerance is met. However this approach takes more refinement iterations than using the initial recommended base mesh (in B tests). It should also be noted that, test_A_4 has introduced higher errors than in test_A_3 as seen in Figure 6-9, Conversely, the local refinement procedure performs very well with an initial recommended mesh in B tests. It took only two iterations (test_B_1 and test_B_2) of the local refinement before reaching an acceptable mesh which has all nodal errors below the prescribed error tolerance. This demonstrates that the path to the qualified mesh, starting from a good quality initial mesh, is significantly better than starting from a poor quality initial mesh. This justifies the earlier work on the case classification and case recognition work as important pre-cursors to high quality meshing.

6.6.1.6 Remarks

The total cost of experimentation is two time steps for the B tests and 6 time steps for the A tests. The total expense of these extra set-up tests will clearly depend on the total simulated time but this typically runs to tens or hundreds of time steps and so this set-up expense does not add significantly to the overall processing time in this case.

The solution for the multiple room case proceeds as expected with the hot gasses from the fire developing entrainment in through the lower door and exhausting a plume through the upper floor window. The temperature iso-surfaces at 200, 300, 400 and 500 degrees C (at t=50 s) can be seen in Figure 6-10.

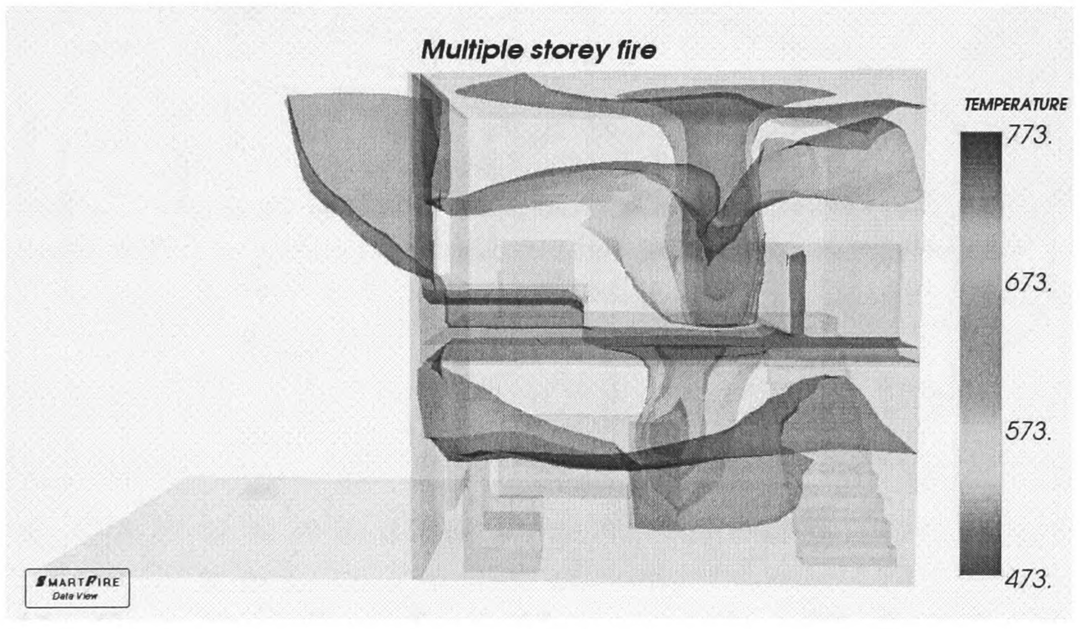


Figure 6-10: Temperature iso-surfaces for the multiple room fire scenario.

6.6.2 Corridors with extraction case

The following example demonstrates the EE supporting a corridor scenario with forced mechanical extraction (an extracting ventilation system) with small vents blowing downwards into all of the rooms. It had previously been observed that this case was problematic to converge using the default solution configuration.

6.6.2.1 The geometry

The corridor scenario is a single floor of building with a number of rooms on each side of the corridor. The corridor runs along the full length of the building. Figure 6-11 shows a wire-frame view of geometry. The dark green (crossed) squares are the duct openings representing the ventilation system. The building covers an area of 10.9 m by 9.0 m with a height of 2.4 m.

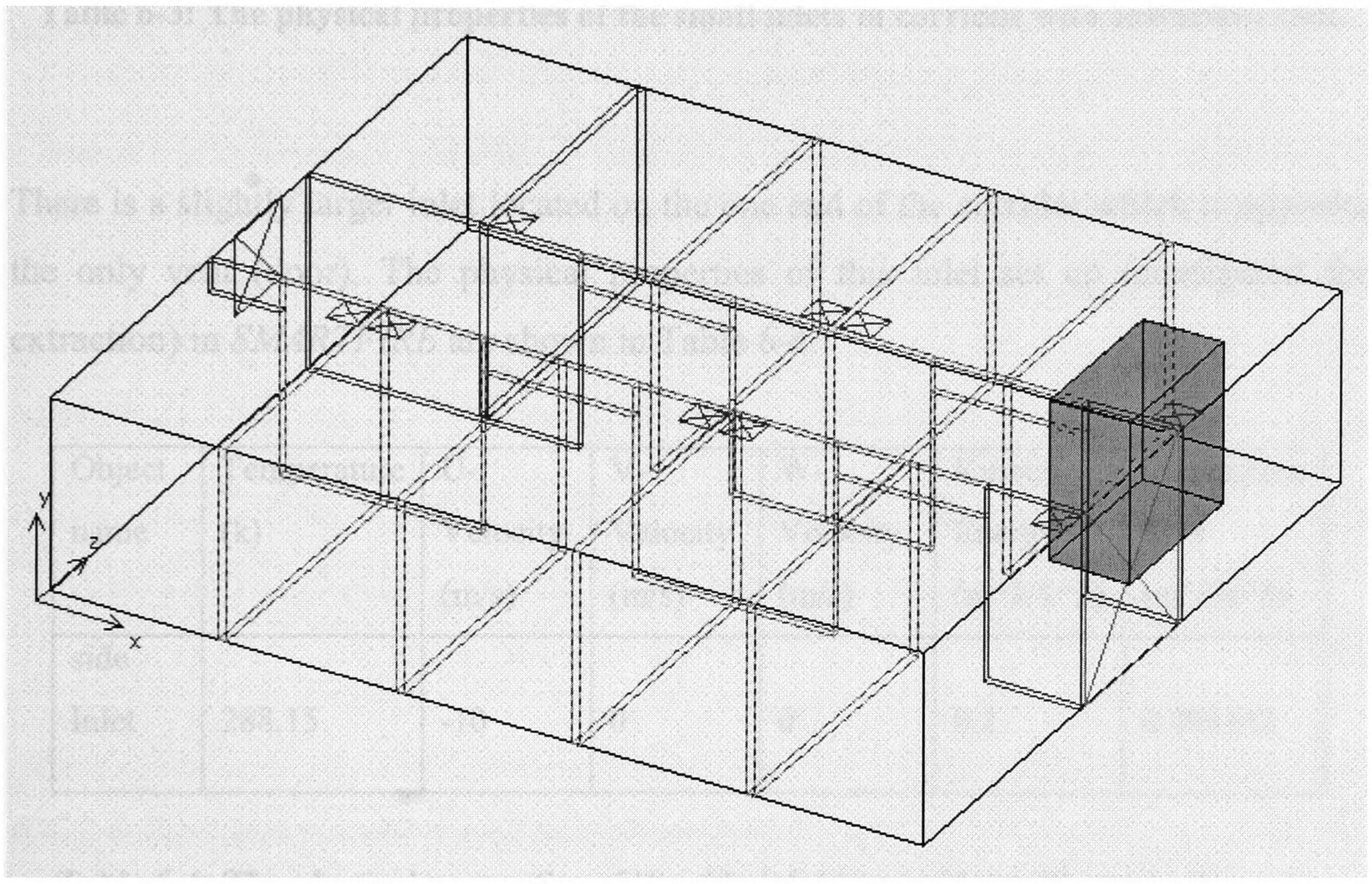


Figure 6-11: Geometry view of corridor scenario with forced extraction in *SMARTFIRE*

6.6.2.2 Vents and inlets

As seen from Figure 6-11, the only vent (door) to the outside of the building is located on one end of the corridor which is near the fire region and is approximately 2.4 m * 2.0 m. The vent was assumed to be open during the entire simulation. There were several small inlets (0.4 m * 0.4 m) located in the ceiling of the rooms along the main corridor.

Table 6-3 shows the physical properties of the small inlets set up in the *SMARTFIRE.*

Object name	Temperature (k)	U-Velocity (m/s)	V-Velocity (m/s)	W-Velocity (m/s)	Kinetic Energy (m^2/S^2)	Dissipation Rate (m^2/s^3)
Small Inlet	288.15	0	-0.075	0	1.125e-05	3.1e-07

Table 6-3: The physical properties of the small inlets in corridor with extraction case

There is a slightly larger inlet located on the one end of the corridor which is opposite the only vent (door). The physical properties of this inlet set up (configured for extraction) in *SMARTFIRE* are shown in Table 6-4.

Object name	Temperature (k)	U-Velocity (m/s)	V-Velocity (m/s)	W-Velocity (m/s)	Kinetic Energy (m^2/S^2)	Dissipation Rate (m^2/s^3)
side Inlet	288.15	-10	0	0	0.2	0.298142

Table 6-4: The physical properties of the side inlet in corridor with extraction case

6.6.2.3 Material properties

The floor of the test case is assumed to be composed of non-conducting material, and the ceiling and all surrounding wall are composed of concrete. The physical properties of the non-conducting material and concrete set ups are shown in Table 6-5 and Table 6-6 respectively.

Material name	Conductivity (Wm^-1K^-1)	Specific Heat (J kg^-1 k)	Density (kg m^-3)	Laminar Viscosity (Pa s)	Thermal expansion coeff (k^-1)	Molecular weight (kg kmol^1)
Non_ Conducting_ Material	0.0000	10000.0000	10000.0000	10000000000	0.0000	0.0000

Table 6-5: The physical properties of non-conducting material in the corridor with extraction case

Material name	Conductivity (Wm^-1K^-1)	Specific Heat (J kg^-1 k)	Density (kg m^-3)	Laminar Viscosity (Pa s)	Thermal expansion coeff (k^-1)	Molecular weight (kg kmol^1)
Concrete	1.4000	880.0000	2300.0000	10000000000	0.0000	0.0000

Table 6-6: The physical properties of concrete in the corridor with extraction case

6.6.2.4 Fire specification and model set up

The single fire source was represented as a large volumetric fire that has an αt^2 growth rate with α=0.113 and a peak Heat Release Rate (HRR) of 1328kW at 108 seconds of simulation time as seen from Figure 6-12.

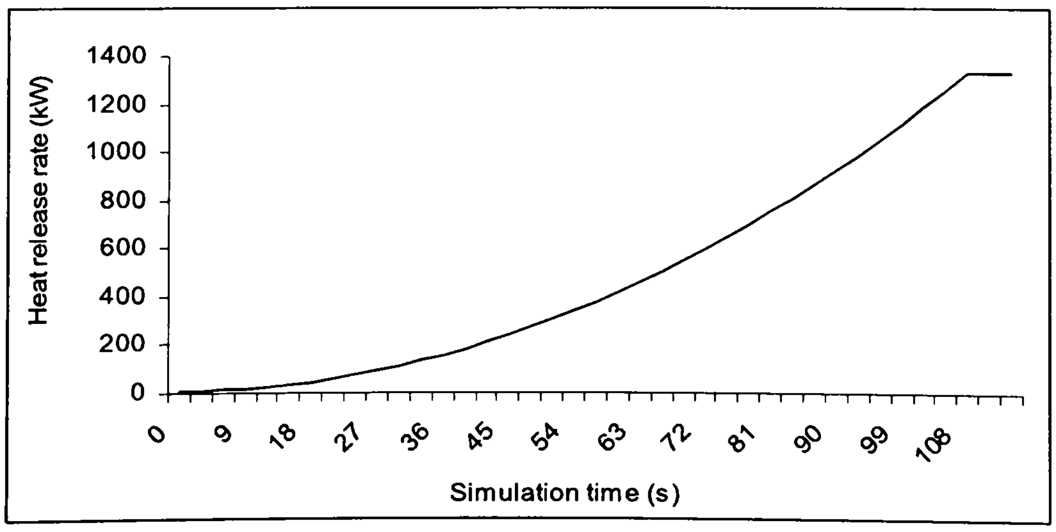


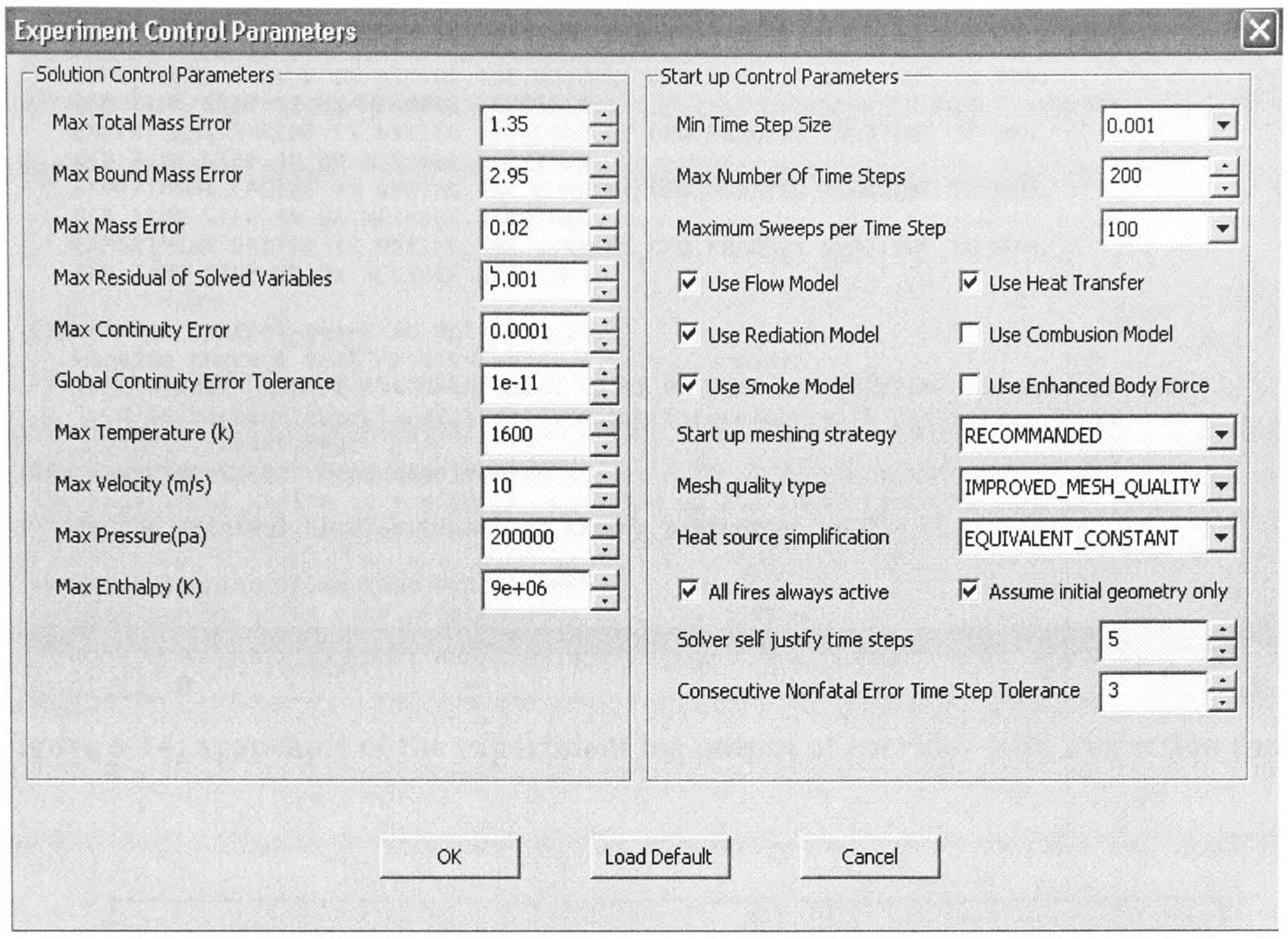
Figure 6-12: Heat Release Rate curve of the fire source in the corridor case.

This scenario uses flow, heat transfer, radiation and smoke models. For simplicity, the combustion model is not used, since there is no data for the precise nature of the fire source except for a given HRR curve.

6.6.2.5 Experiment Engine Configuration.

- Nodal error Tolerance is 1E -4.
- The initial base mesh is set to be the recommended.
- The Experiment Engine is activated from the beginning with an "Improved Mesh Quality" strategy which means whenever the mesh becomes too distorted, block-wise mesh justification procedure will be automatically applied.
- Heat simplification strategy is set to Equivalent Constant with means the HRR is set to an "average" constant value of 450 kW for the test. This is dynamically generated by the EE only during the Experiment phase of the simulation.
- The Minimum time step size is set to 0.001 and the maximum sweeps is set to 100.
- Time allowed for experiments time was set at a quarter of the actual simulation time.

A snapshot of the experiment engine configuration panel is shown in Figure 6-13.


Experiment Control Parameters

Solution Control Parameters

Max Total Mass Error	1.35
Max Bound Mass Error	2.95
Max Mass Error	0.02
Max Residual of Solved Variables	0.001
Max Continuity Error	0.0001
Global Continuity Error Tolerance	1e-11
Max Temperature (k)	1600
Max Velocity (m/s)	10
Max Pressure(pa)	200000
Max Enthalpy (K)	9e+06

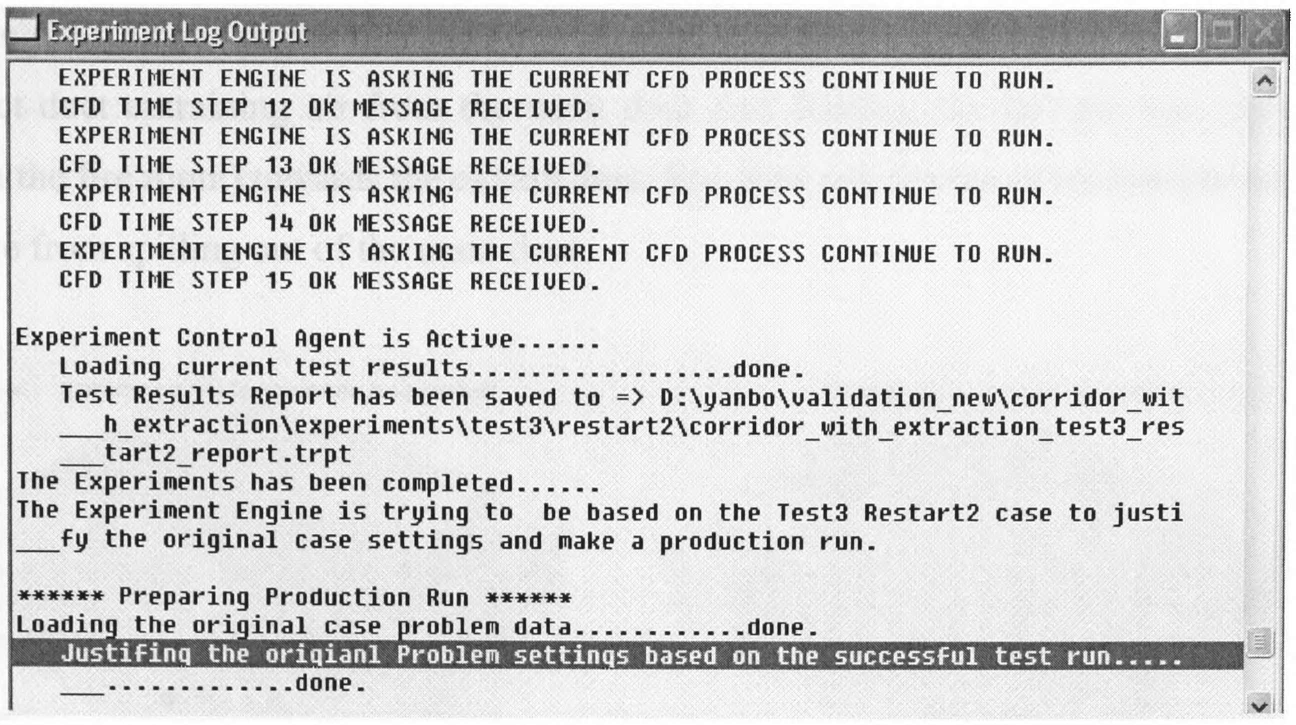
Start up Control Parameters

Min Time Step Size	0.001
Max Number Of Time Steps	200
Maximum Sweeps per Time Step	100
✓ Use Flow Model	✓ Use Heat Transfer
✓ Use Rediation Model	Use Combusion Model
✓ Use Smoke Model	Use Enhanced Body Force
Start up meshing strategy	RECOMMANDED
Mesh quality type	IMPROVED_MESH_QUALITY
Heat source simplification	EQUIVALENT_CONSTANT
✓ All fires always active	✓ Assume initial geometry only
Solver self justify time steps	5
Consecutive Nonfatal Error Time Step Tolerance	3

Figure 6-13: Experiment controls for the corridor with extraction case.

6.6.2.6 Results

The experiments start with an initial base mesh of some 10044 cells (i.e. nx=54, ny=6 and nz=31) and an initial time step size of 10 s, after 3 tests and two restarts, in the last test directed by the test controller of the Experiment Engine, the experiments end with a final mesh of 28512 cells (nx=72, ny=9 and nz=44) and a final time step size of 2 s, which was then adopted for the production run. It was observed that the required convergence was achieved in each time step in the production run. The following snapshots and figures are used to show how the EE progresses the solution, the success of the experiments process, the comparison of the initial mesh and final (most refined) mesh and the residual graph of the production run at the end of the simulation respectively.



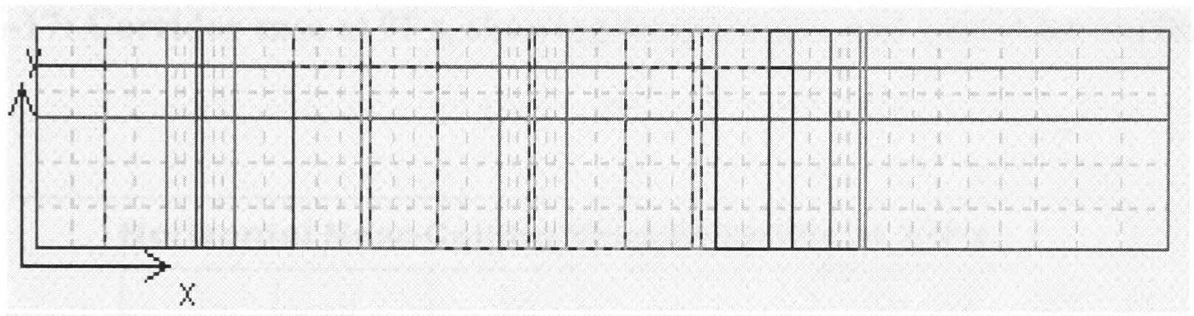
```
   EXPERIMENT ENGINE IS ASKING THE CURRENT CFD PROCESS CONTINUE TO RUN.
   CFD TIME STEP 12 OK MESSAGE RECEIVED.
   EXPERIMENT ENGINE IS ASKING THE CURRENT CFD PROCESS CONTINUE TO RUN.
   CFD TIME STEP 13 OK MESSAGE RECEIVED.
   EXPERIMENT ENGINE IS ASKING THE CURRENT CFD PROCESS CONTINUE TO RUN.
   CFD TIME STEP 14 OK MESSAGE RECEIVED.
   EXPERIMENT ENGINE IS ASKING THE CURRENT CFD PROCESS CONTINUE TO RUN.
   CFD TIME STEP 15 OK MESSAGE RECEIVED.

Experiment Control Agent is Active......
   Loading current test results..................done.
   Test Results Report has been saved to => D:\yanbo\validation_new\corridor_wit
   ___h_extraction\experiments\test3\restart2\corridor_with_extraction_test3_res
   ___tart2_report.trpt
The Experiments has been completed......
The Experiment Engine is trying to  be based on the Test3 Restart2 case to justi
___fy the original case settings and make a production run.

****** Preparing Production Run ******
Loading the original case problem data............done.
   Justifing the origianl Problem settings based on the successful test run.....
   ___.............done.
```

Figure 6-14: snapshots of the experiment log output of corridor with extraction case.

(a)



(b)

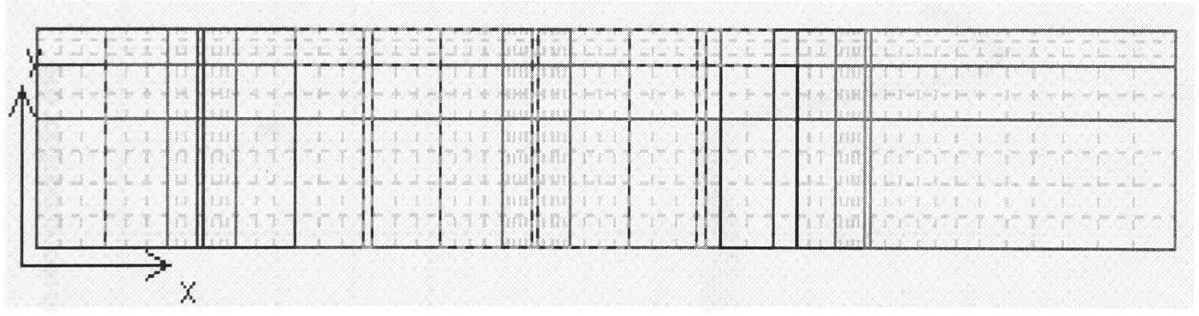


Figure 6-15: Comparison of the (a) initial and (b) final/most refined y-mesh for the corridor scenario with extraction.

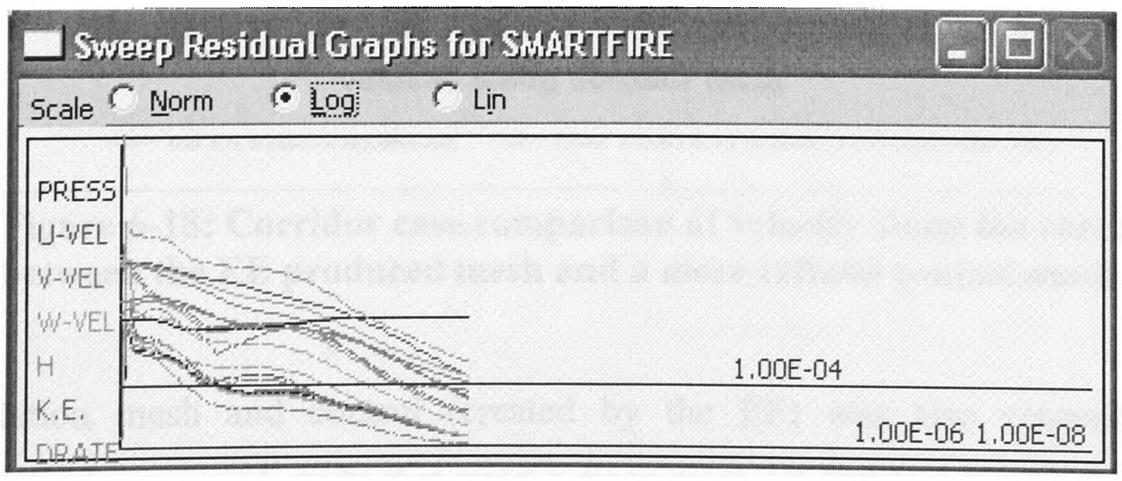


Figure 6-16: residual graphs at end of the production run for corridor scenario with extraction.

Figure 6-17 below shows that the simulation proceeds as expected with the corridor extract duct entraining air from the main door and drawing the hot gas and smoke (from the fire room) towards the extract duct. The flow rate for the extraction prevents smoke from spilling out of the main door.

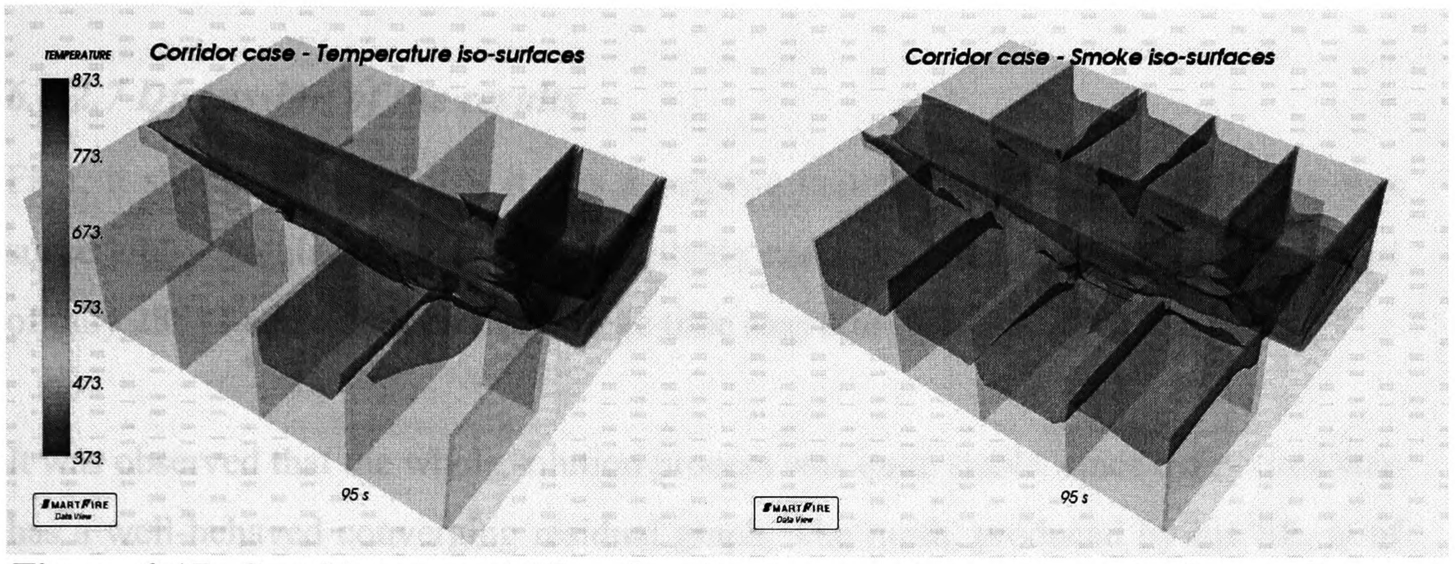


Figure 6-17: Corridor case at 95 s, showing temperature and smoke iso-surfaces.

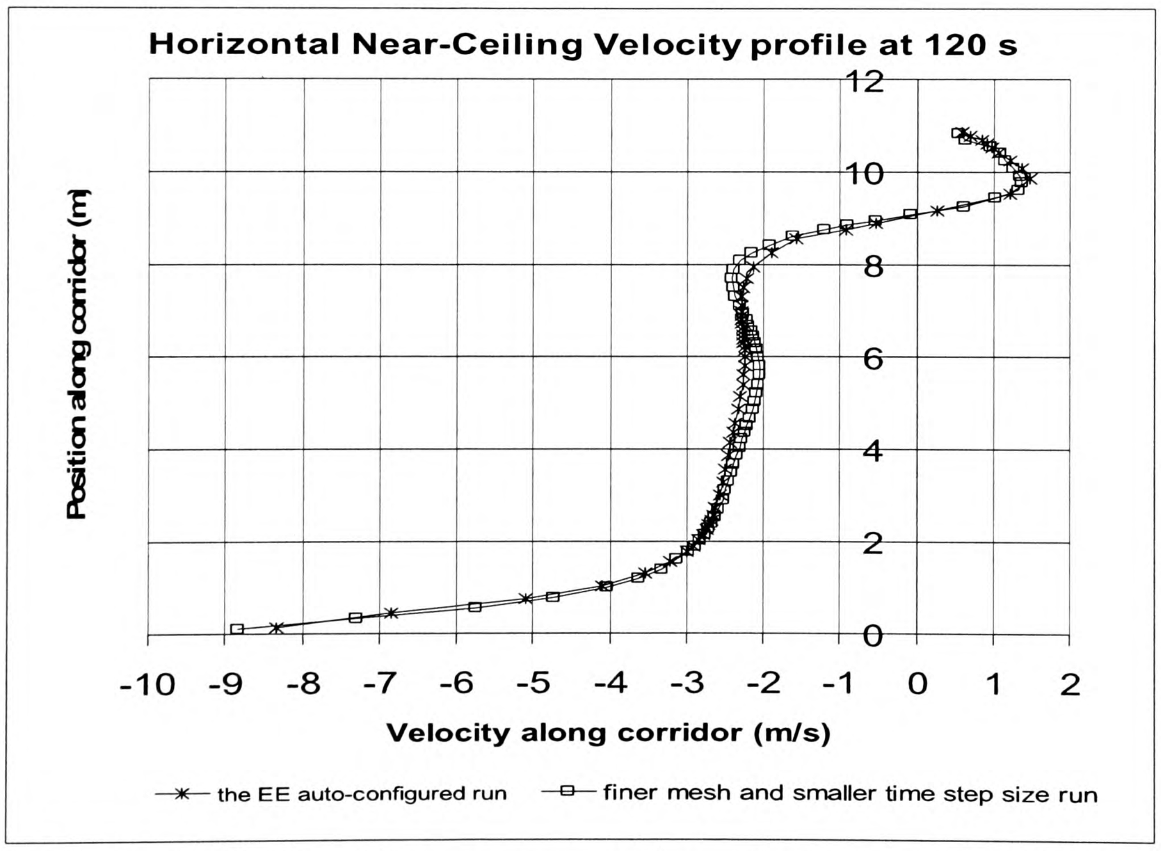


Figure 6-18: Corridor case comparison of velocity along the corridor between the EE produced mesh and a more refined control mesh.

The simulation mesh and control (created by the EE) was also compared for consistency with a control study that used a finer mesh of 40480 cells (with nx=80, ny=11 and nz=46) and a smaller time step size of 1 s. Figure 6-18 shows the data comparison of horizontal velocity (along the length of the corridor) with position, at the end simulation time of 120 s. This is in a horizontal slice near to the ceiling. The graph shows good agreement between the two sets of results, indicating that the EE solution has not suffered from using a less refined mesh and a larger time step size.

6.6.2.7 Discussion of the results

First, it should be noted that it was a surprise that this formerly unstable case was successfully simulated using a comparatively coarse mesh (a final total mesh budget of only 28512 cells) and relatively large time step size of 2s.

It was observed that the whole solution process was quite stable, since every time step has a well-behaved converging residual graph. The mesh, produced by the EE, is of good quality since fine resolution cells have been applied to the near fire region and the region near the extraction duct, with all other regions remaining as coarse as possible (as shown in Figure 6-15(b)).

Second, the control strategy for the EE is to generate an appropriate mesh first, and then to find the corresponding time step size. The mesh refinement is set so as to make sure that the maximum continuity error will be well below the predefined error tolerance. This is followed by a time step size search starting from 10 s, which is almost the largest time step size that an Expert user would choose (or be able) to use in any complex fire scenario at reasonable cell budgets. Theoretical study has show that, to get a stable simulation the largest CFL (Courant_Friedrichs_Lewy) number anywhere in the flow field must strictly obey,

CFL < CFL critical Where, $\mathrm{CFL} = u\Delta t / \Delta L$

u: characteristic speed,

Δt: time step size,

ΔL: control volume size

Using Courant limit, it is possible to determine an acceptable ΔL, and corresponding Δt, for the local flow speed.

The experiments were performed on a desktop computer with an Intel Pentium 4 3.0 GHz CPU with 2.00 GB of RAM. The total CPU time for all of the experiments was 3h 22m 51s and the CPU time for the production run was 5h 24m 08s (i.e. a total combined CPU time of 8h 46m 59s for the EE supported simulation). Hence, the overhead of performing the experiments, for the corridor scenario with forced extraction, is approximately 38.5% of the complete processing time. This is tolerable considering that no user intervention was required during the simulation. The unsupported control study took a total of 12h 21m 10s. Thus the EE (even including the time taken for the experiments) saves approximately 28.8% of the run-time of the control simulation.

6.7 Initial conclusions

The analysis of the results substantially enhanced our understanding of the control techniques. It was established that the new solution process worked correctly and the Experiment Engine is quite robust and effective. In the Experiment Engine enabled test runs, appropriate set up and meshing was ensured and the desired accuracy of the solution was obtained. This is a very important step towards providing users with complete support during the whole simulation process. At this stage, the prototype Experiment Engine has not yet actively controlled the production run, after the appropriate set up, meshing, and control parameters established in the experimental phase of the simulation. And the Experiment engine has successfully emulated an Experts' ability to run an arbitrary new scenario in a limited way (i.e. to make the set up and meshing better through trial runs). In practice, an Expert is able to significantly improve the performance by justifying the control parameters at the production stage of the simulation and also making sure that the convergence is always achieved. Therefore, further development of the Experiment Engine (discussed in the next chapter) should focus on looking for performance gains by applying run time justification of the solution control parameters and to help to guarantee the simulation stability and convergence assurance in the production phase of the simulation. The ultimate aim of the Experiment Engine research is to always provide an acceptable CFD solution without the need for human monitoring or intervention and be able to provide assurance of obtaining a solution under extremes of problems encountered.

[Chapter summary]

This chapter described the instantiation and testing of a new framework that successfully integrates a fully automatic Experiment Engine into a Fire Field Modelling solution process, and presents the initial tests of the *SMARTFIRE* prototype version of the Experiment Engine. The results of two selected test cases show the validity of the local block-wise refinement regime and the effectiveness of the Experiment Engine. A key achievement is the delivery and demonstration of an automated control system that ensures overall simulation stability and reliability.

The following summarizes the key test results:

- Assessment using a coarse mesh solution can provide improved set-ups and meshing quality.

- Starting with a "good" quality of base mesh (using Knowledge Based meshing techniques) reduces the need for refinement and, hence further increases the system efficiency.

- Local adaptive refinement procedures enhance the quality of the Knowledge Based meshing and thus further improve the performance.

- The integration of the Experiment Engine into the solution process offers a considerable enhancement in terms of simulation reliability and (in both test cases) is able to provide the user with the requested solution accuracy.

- The prototype Experiment Engine is (in a limited way) able to emulate an Experts' ability to run a new scenario in a logical and efficient manner and is fully automatic.

7 Further Enhancement and extension to the Experiment Engine as a fully robust and automatic control technique to take control of the whole fire simulation process

[Chapter Overview]

In the previous chapter, it was established that the prototype Experiment Engine was able to emulate the Experts' ability to make appropriate set up and meshing decisions, for arbitrary scenarios, according to a user defined preference for the required solution quality. It also highlighted where the Experiment Engine fit into the existing software architecture of a FFM system, and discussed the new software architecture that was developed to accommodate these new techniques. Initial tests have been conducted and the test results have been thoroughly analysed in order to examine the validity of the new techniques and to look for any areas of potential improvement. It was encouraging to note that the new architecture performs correctly and that the local h-adaptive mesh refinement was effective. These successes allow attention to be focussed on improving, extending and making the most of these new techniques so that generation of successful fire simulations can be made even easier.

In this chapter, the Experiment Engine concept has been further investigated in order to replace the existing ICS component. This was considered necessary due to the inherent limitations of the ICS methodology. It became apparent that the Experiment Engine should now continually monitor the solution process, after achieving an appropriate set up and meshing solution in the experimental phase of the Experiment Engine process. Furthermore, it was considered vital that the EE should then make run time adjustment to the solution control parameters as and when they are necessary. Consequently, the fully automatic EE enabled simulation can actively respond to any transient events and heat release changes in the production phase of the simulation and hence can help to guarantee convergence – when it is possible. In this way, the Experiment Engine is able to thoroughly support the simulation process from start to finish, and make the whole process as reliable as possible, so as to free the user from having to monitor, or interact with, the simulation. As a result, the simulation process has been made more robust and the Experiment Engine enabled simulation process can obtain solutions to a prescribed accuracy, with minimal user intervention.

7.1 Introduction

Simulations of fire scenarios are very complex numerical problems requiring considerable computing power. Significant Expert knowledge is necessary to correctly set up a problem (as discussed in the previous chapters) and choose appropriate control parameters. Formerly, this was acceptable when the scenarios were small and the required expertise was always at hand, for example, for the small scale room based scenarios, the Experts typically know what the control parameters should be in order to adequately solve the problem. In addition, a modern interactive CFD code (e.g. *SMARTFIRE*) contains a sophisticated interface that gives real-time access to all the data during the simulation. A user has full control over the simulation process and can access and modify all of the control parameters. Consequently, an Expert can substantially reduce the execution time and improve the accuracy of the results by continuously or regularly watching the progress of the simulation and making necessary adjustments as, and when, required. In the case that a solution fails, the user can use the restart facility to resume the solution from the last saved bookmark point where the solution was still acceptable so as to avoid having to start over from the beginning of the simulation. However, the advancement of simulation software complexity, along with increasing computer power and the increasing capacity of memory chips, means that it is becoming possible to simulate larger and ever more complex fire scenarios. These scenarios often consist of high heat output defined with complicated heat release rate curves and with potentially numerous transient events that can occur during the simulation (e.g. breaking windows, opening doors, activated extract fans). Unfortunately, such cases have turned out to be more difficult to control, because the nature, location, and duration of the transient events are often not known in advance and these simulations typically take days to complete (in some large cases, the processing time is measured in weeks). For these cases, the control actions are even more necessary since the simulation will go through distinctly different periods (i.e. sudden large increases in heat release rate and/or have different transient events happening), obviously there is no single set of control parameters which will be optimal or even "adequate" for the whole of the simulated period. Also it is unreasonable to expect that any Expert is ready to spend days (or weeks) in front of the computer diligently monitoring the solution status, adjusting control parameters for optimal performance and correcting problems as they develop. Thus the only real alternatives are first, to set up a very conservative set of control parameters which, it is hoped, will produce acceptable quality of results but may result in excessive computational costs and extended processing time. Second, to use Expert knowledge to make educated guesses to pre-configure different sets of control parameters applied for different period of the simulation and hope that the simulation will pass these difficult periods in terms of simulation stability, maximising convergence rates of all time steps and maintaining optimal performance. Naturally this situation had encouraged researchers to take the approach of providing code interactivity and automated solution control. The following two sections describe two previous attempts to provide automatic solution control in *SMARTFIRE* namely Rule-based Relaxation Adjustment System and the Intelligent Control System (ICS), as they provide valuable ground work for the current research and as the starting point for investigating the new solution control techniques that have been investigated during the current research and are now embedded in the Experiment Engine.

7.2 Simple Rule-based Relaxation Adjustment System

One of the first attempts to address the control problem in ***SMARTFIRE*** was made by Ewer[Ewer-98]. A prototype dynamic control module within the early ***SMARTFIRE*** system was developed to investigate the potential for automating the process of dynamically monitoring and controlling the solution of a particular class of fire simulations. The system's main goal was to improve the convergence rate and thus improve the performance of CFD codes and to attempt to avoid divergence. Ewer designed his control system to monitor the local convergence behaviour and then modify the linear and false time step relaxations based on a set of production rules. The rules were fairly simple and the control decisions were based on the most recent residual errors as indicators of convergence state. The system was able to demonstrate significant saving in run-times on a simple 2D scenario when compared with a non-controlled simulation using default control parameter settings. However, it was observed that there is also some danger of de-stabilising the solution as a more complex 3D scenario was actually de-stabilised by the solution control module. It was predicted that these problems are caused by the lack of monitoring of persistent trends in the solution convergence behaviour and Ewer acknowledged that further research was necessary before an automated control system could provide tangible improvement on more complex and arbitrary 3D cases. Nevertheless, Ewer provided significant grounding work for developing future solution control system. First of all, the research demonstrated it is possible for a run-time control system to reduce simulation time and improve the solution stability. Second, the residual errors were proved to be a main indicator of the solution state and are reliable for control decision makings. Third, the previous work indicated a continuing research direction for the subsequently development of the Intelligent Control System.

7.3 Janes's Intelligent Control System

Janes started his research into solution control techniques based on Ewer's Rule-based Relaxation Adjustment System.

7.3.1 Description of ICS

In his initial work, Janes focused on identifying the deficiencies of Ewer's approach and, based on these deficiencies, he developed more sophisticated residual graph feature extraction algorithms for more accurate and comprehensive assessment of the solution state, while the overall principle of Ewer's control system was unchanged. He also tried to formalise the Expert knowledge and document the control techniques that were typically used by CFD Experts when using Fire Field Modelling. This included definition of convergence and divergence, the level of importance of each solved variable, the influence of various residual graph features on an Expert's control decisions and identification of different solution states during the simulation. As a result, the definition of convergence and divergence has been clarified as residual errors of all variables should be below a prescribed tolerance level and PRESSURE and VELOCITY (i.e. the flow modelling variables) are the most important in assessing the stability for the simulation. However, Janes failed to uncover a consistent set of "globally applicable" rules for controlling a CFD simulation since various Experts used slightly different corrective techniques. In addition, it was also confirmed by Experts that, apart from residual graphs in the current time step, the residual graphs from previous steps and the physical condition in the domain are also very important in making any control parameter adjustments. Based on this additional knowledge, Janes developed a second version of Rule-based Relaxation Adjustment System. The main improvements on the original system were, first of all, a more comprehensive trend assessment algorithm was deployed. Second, four different control phases in a single time step were identified and different control rules developed for each of them. Consequently, a more sophisticated control action scheduling was implemented. However, despite these improvements, the control actions were still limited to relaxation adjustment applied repeatedly within the current time step and the residual graphs remained the exclusive source of information about the solution state. It was believed that, with the new enhancements, the new system should be able to provide tangible benefits for complex 3D cases. Unfortunately, the new system was still unable to control arbitrary 3D cases effectively. This was demonstrated by the fact that, even in the best cases, the observed performance improvement was lower than 7%. It was concluded, after thorough analysis, that the major factor contributing to the failure of this control strategy was the "kick effect" that occurs immediately after the control changes were made. This introduced an instability and thus degraded the convergence performance. Eventually, janes decided to move away from Ewer's original control architecture and associated control actions, and this resulted in the development of the Intelligent Control System. The fundamental difference between the two systems was that the ICS made all of the solution control parameter adjustments between time steps. In this way, changes to the control parameters do not introduce adverse effects on simulation stability. Consequently, residual graphs can be assessed by an additional "global" view of the whole time step and are thus more reliable than by local residual gradient alone. And this also made it possible to adjust additional control parameters, for example, the time step size between the time steps. Janes then went on to devise a set of tests to investigate the effects of various type of control actions applied between time steps during the simulations. All these tests used scenarios based on a Steckler type case but with different heat source locations and different fire sizes, etc. In total, there were 5000 experiments performed, in which each type of control action was tested 250 times, and this was used for statistical analysis of convergence speed with the aid of visual analysis of the residual graphs. Janes derived some general observational rules from this study, he was then able to characterise three categories of control actions namely those relating to speed-up, divergence recovery and oscillation removal. These were later used in Goal-driven search plans to provide a substantial reduction in the number of experiments required to search for an optimal path at scheduled times during the simulation. The overall ICS structure is based on heuristic search techniques to find a near-optimal set of control parameters at each check point in order to reach the goal in the shortest possible time thus reducing the simulation time. The performance of the control system was demonstrated using various room based fire scenarios. In the 54 kW heat release rate case scenario, the ICS system was able to reduce the number of iterations required by 60% compared to non ICS controlled runs. However, in the scenario with a 250 kW heat release rate case, the ICS system failed to provide any significant improvements and the simulation was, in fact, slowed down by almost 50%. This was partially due to the fact that some of the relaxation parameters (for some of the variables) had reached their maximum and could not be modified further while others were still being relaxed. This demonstrated that the system was not infallible and could not guarantee performance improvement in every case. Also it is worth noting that in most ICS controlled cases, the search costs remained computationally expensive and used many more sweeps than the actual sweeps used to produce the simulation results. Janes attributed these problems to an inherent feature of heuristic systems. Nevertheless Janes's ICS system had made a big step forward from Ewer's Rule-based Relaxation Adjustment System, since the ICS was able to make significant savings in some cases for more complex 3D scenarios.

7.3.2 Benefits and limitations of ICS

A brief summary of the benefits and limitations of Janes's approach is presented below to serve as a starting point for the new control system developed within the Experiment Engine.

Janes's major contributions:

- The research showed that it was possible for an automated system to be able to emulate human expertise to maximise the performance gains while at same time improving the solution reliability. It also provided benefits for ease of use and efficiency of the numerical software.
- Janes believed that all the modifications to the control parameters should be applied between time steps. In this way, changes to control parameters do not introduce any artificial irregularities during the time step and therefore the residual graph can be assessed by the techniques similar to the ones used by human experts.
- The research produced a set of comprehensive graph feature detection routines to assess the state of the residual graphs.
- The research proved Ewer's belief that the residual error were adequate as the main indicators of the simulation state and that most of the information relevant to the control decisions can be retrieved from the residual graphs until

further knowledge is available on how the physical conditions in the domain can inform the decision of changes of control parameters.

The main limitations of the ICS system:

- ICS system was based on heuristic search techniques and has inherent limitations. For example, in most testing cases the search costs was actually higher than the actual computational cost used in producing the simulation results, regardless of whether the ICS provides theoretical time savings or not compared to the unsupported simulation.
- The investigation on the effects of various control actions were not thorough enough to be applicable to any arbitrary fire modelling scenario, since the tests were merely conducted on small scale room based scenarios. Such scenarios do not represent the more typical complex scenarios that will be met by CFD Fire Field Modelling users.
- The ICS uses combined changes of various control parameters. These could be caught out and unable to handle exceptional events. Such exceptional events can leave the ICS unable to respond quickly enough to damp out the problem. It has also been noticed that, once the solution has taken a wrong path, it can be very difficult to recover, hence the simulation is likely to become invalid or to fail completely.
- It is not advisable to change too many control parameters at once, as it becomes difficult to analyse which of the changes have positively influenced the solution.
- An appropriate evaluation function is absolutely vital for heuristic search based techniques. Unfortunately, the evaluation used in ICS system has a potential flaw as it was unable to determine the necessary priorities and importance levels of the various features in the residual graphs, consequently the function that produced a final assessment based its decision on merely a simple combination of the sub-assessments of each feature.
- Control decisions were based on the results of an incomplete control parameter changes study.
- All these factors contribute to the ICS providing inconsistent behaviour to diverse cases and cannot guarantee the success of controlling the various types

of simulations. More generic control decisions, backed by Expert knowledge, are needed for the development of a more robust and flexible system.

7.4 A new approach for the control system

The identification of deficiencies in the ICS made it clear where to improve the ICS system and what to do in order to achieve a better and more generic control system. At this stage, it is apparent why the ICS has not delivered the quality of solution control that was hoped for.

7.4.1 Limiting the control parameter changes to one at a time

A control system that relies on heuristic techniques to determine the optimal control parameters in CFD applications has inherent limitations. First of all, the basic idea of heuristic search is that, rather than trying all possible search paths, attention is focused on paths that seem to be getting nearer to the goal state. Of course, one generally cannot be sure that a given path is really leading nearer to the goal state. But it might be able to "guess" that this as a reasonable path. Heuristics are used to help make that guess. In this sense, "heuristic" is a technique which will sometimes work, but not always. When a heuristic solution does not work, we then have to deal with that issue. Unfortunately, this feature of a heuristic system is not really acceptable for controlling CFD simulations because the choice of an inappropriate control parameter (produced by a heuristic search) can potentially lead to disastrous results as it can introduce simulation instability problems and, in the worst case scenario, it can even crash the CFD simulation. Second, the search costs are too high and it is very difficult to keep these at an acceptable level if multiple searching paths exist. This was demonstrated by the fact that search costs in the ICS system were significantly higher than that actual computational cost used in producing the simulation results for most test cases. Although some later ICS enhancements were made in order to minimize the search costs, typically the high search costs remained. This can be attributed to the fact that ICS system was trying to modify several parameters at once; this has inevitably increased the search space dramatically. Furthermore, the effects of combined control parameter changes are still largely unknown. Therefore, for the new control system, the control parameter changes should be limited to one at a time. This will also narrow the search space significantly, and consequently the search costs are kept to a minimum even with no heuristic.

7.4.2 Knowledge acquisition

Now the question is which control parameter should be changed by the new control system during the simulation. This control parameter has to be chosen very carefully and has to be well understood for its effects on the solution when changes are made. In a CFD simulation, one of the worst problems is that a simulation that has been running for days (or weeks) can suddenly become unstable and it will not even be possible to get a solution after all the processing time that has been spent. In this sense, the most important factors for the CFD simulation are the solution stability, accuracy and reliability. Compared to these factors, the simulation processing speed is somewhat less important. Needless to say that the most important factor for a CFD user is that each simulation that should get a solution (i.e. is well posed), will be able to do so. Of secondary importance is the concern about how long it takes to obtain that solution. So the new control system has, first of all, to make sure that at the end of the simulation, a solution is produced which is the best that can be obtained from the simulation according the user's preferences and within the resource limitations. Second, the new control system should make the simulation run as quickly as possible without loss of solution accuracy. With these requirements, the current research focused on establishing precisely which facts were known and could be used for the purpose of the automated control of CFD simulations. Fortunately, after consulting with a number of CFD Experts, it was determined that there was at least one parameter that is well understood, namely the time step size. There was general consensus of opinion among the Experts on the effects to the simulation stability after making time step size changes. Generally speaking, there is a limit on the magnitude of sudden physical (or chemical) changes which a numerical code can handle. Excessive sudden changes of critical solution conditions (i.e. heat release rate changes or other changes caused by transitional events) are the largest problems affecting the success of a CFD simulation. On one hand, the time step size is a measure of how much the simulation can progress in one time step of computation. On the other hand, it is also a means of controlling/limiting the magnitude of the physical changes that are applied to each time step in the simulation. This gives us a clue in how to ensure the stability of the CFD simulation. when the magnitude of sudden physical changes approach the limits that the numerical code can handle, then a smaller time step size should be applied to break the excessive changes into smaller portions such that each portion is within the limitation that the code can handle. The same logic applies to a bigger time step size. A bigger time step size may make the simulation advance faster as larger time periods are passed in each time step. However it also may make the simulation less stable depending on the magnitude of physical changes during that time step. We can sum up the Expert knowledge as follows,

- Increasing the time step size can lead to divergence but may also speed up the overall simulation.
- Decreasing the time step size tends to stabilize the simulation but not necessarily by slowing down the simulation (this seemingly contradictory fact is explained in the following section).

Moreover, the Experts believed that there is an optimal time step size range in each particular stage of the simulation depending on physical conditions in the domain at that time. The time step size should be adjusted to match both the physical conditions in the domain and the rate of change of those conditions, from time to time. Based on the assumption above, it is possible to say that by always using an appropriate time step size, during the simulation, can help to guarantee that the simulation is stable. Once the solution stability and accuracy are assured, then it is possible to consider the simulation speed issue.

7.4.3 Analysis and interpretation

Until now, no mention has been made of the one other factor which contributes to the simulation accuracy and speed. This factor is the number of sub-iterations within each time step, also known as sweeps or the outer-loop iterations. It is often believed that increasing the outer-loop iteration number can be a remedy for the loss of numerical accuracy resulting from using an overly large time step size. However, large sweep numbers typically result in much longer CPU time (please refer to the relevant section of literature review in chapter 3) since each sweep has typically a similar computational expense – regardless of the time step size used. There is a trade-off between the time step size and number of sweeps in order to achieve numerical accuracy required with minimum computation effort. It is assumed that each sweep costs the same amount of CPU time with a fixed case set up (i.e. the mesh is not changed during the simulation). So the numbers of total sweeps used in the simulation can be the measurement for assessing total cost of the simulation and thus calculating the total time that the simulation will run for. For example, two different time step sizes of 1 second and 2 seconds were used for one time step (assuming that in both cases the convergence can be achieved in the time step), and the maximum number of sweeps is not specified (i.e. the time step can use as many sweeps as needed to converge to the prescribed error tolerance). The processing using a 1 second time step needed 40 sweeps to achieve convergence, while the 2 second time step needed 100 sweeps to achieve the required accuracy. Then it is possible to say that, on average, the 1 second time step size test performed better than the 2 seconds time step size test in terms of saving simulation time, since it only used 40 sweeps to simulate 1 second of simulated time, while in the second test, 50 sweeps (i.e. 100 / 2) were used to simulate 1 second of the simulated time. This demonstrates that the bigger time step sizes "may" speed up the simulation, however smaller time step size do not necessarily slow down the simulation. It is known that simply applying an appropriate and optimal value of the time step size with an adequately large or uncapped number of sub-iterations, it is possible to guarantee the simulation stability and accuracy, and have benefits of speeding up the simulation. This makes it clear that the new control architecture should be build around applying time step size changes according to the simulation state with a dynamic corresponding number of sweeps. These changes are applied between time steps the time step size has to be consistent throughout a time step. Furthermore, the heuristic search is no longer necessary because the changes are comparatively simple – either going up or going down for the time step size. However it is not a guaranteed that the change of time step size will give a resulting speed up for the simulation, since this depends on physical conditions in the domain at time, whether the new time step size is more appropriate then the old time step size (it can be worked out very easily by comparing the ratio between the time step size and required number of sweeps to achieve the convergence), the suitability of all other control parameters and more importantly the search costs. Nevertheless, this technique was suitable for designing the new control system based on these observations and consequently, the new control system will help to ensure the stability and full convergence of the simulation whenever this is possible within constraints of the scenario and the available resources. Furthermore, the new control system will not suffer from the high search costs that the ICS did, because the search space is much smaller, hence the system has a better chance of speeding up the simulation. There is another consideration for designing the new control system so that it has a very limited search domain. First, because the typical fire simulations are so complex and transient in nature (e.g. in terms of changing heat release rate and consequent transient events etc.) so at any time during the simulation, the control parameter is likely to be optimal for only a short period of the simulation. So the idea behind the new control system is only to look for a performance improvement when the solution is stable at time, and limit the search costs to an absolutely minimum by not applying the search extensively, if the possible improvement can be easily/inexpensively determined then apply the change, otherwise the current control parameter will be kept. This will help to ensure that the simulation can go forward as quickly as possible. This is similar to the Chinese idiom which says that "it is not big problem with moving slowly compared with not moving at all" (i.e. during the search period). It is not sensible for the ICS to spend a lot of time on searching when the solution control is already near the optimal path, since the search only uses fairly "local" information (i.e. the most recent time step). The danger of a local search (without history) is that it can be overly sensitive to temporary instabilities that will soon be passed. The new control system is quite simple but its application is designed to be highly efficient, because the limitation to changing one parameter at a time, means that a change of control will have a more assured effect in terms of simulation stability. The new control system can quickly respond to transitional solution behaviour since the single dimension of search path will lead the time step size down-to the required level as the solution becomes unstable or up when the simulation becomes stable again.

7.5 A simpler, direct and yet better evaluation function and a sophisticated convergence forecast function for determining the simulation state

7.5.1 A better evaluation function

It is believed that an appropriate time step size should be adjusted to the physical condition in the domain at that time. But there is no easy way to determine the solution state by simply observing the solved variables, because it is easy to incorrectly interpret the stability of the solution state. The heat release rate curve may provide useful information, but it does not tell the full story (e.g. it cannot indicate when all of the transient events will occur, etc.). Hence, a broader and more generic view is needed to determine the instantaneous solution/simulation state. It is thanks to the earlier research by Ewer and Janes, that indicated the following conclusions:

- The residual graphs provide essential information about the simulation state. Therefore, the control decisions can be based almost exclusively on the results of the residual graph assessment.
- The residual graphs can also be analysed after adjustments have been made to assess the results of any control actions

The combined evaluation function, in the ICS system, was based on the above conclusions. This took account of three major factors that could be determined from the residual graphs, namely convergence speed (number of sweeps required to attain convergence), presence and magnitude of irregularities (finding sections of graphs where the residuals differ significantly from the global trend) and presence and magnitude of oscillations (finding sections of graphs that exhibit strong periodic variations in residual values). Unfortunately the earlier ICS research was unable to propose a best solution about how to combine these three separate measures into a single final assessment. The design of the final evaluation function, in the ICS, was actually based on a guess at the method of combination, rather than understanding the underlying priorities – as its author has indicated – and therefore putting a question mark over the quality of the assessment result. In order to eliminate this ambiguity, in the new control system, a much simpler and more direct approach is taken. Ultimately the major requirement for the residual graph is that it has reached the required error tolerance and assessing how quickly/costly it has been achieved in terms of the number of sweeps used. Considering the simulation as a whole, it is necessary to determine how efficient and effective the solution control is in the current time step in terms of the ratio between time step size and number of iterations used (the convergence rate can be represented as sweeps PER time step size). If the convergence has not been achieved, in a reasonable number of sweeps, then the time step size needs to be changed to a smaller time step size. Conversely, if the convergence is obtained easily, then it might be worth trying a larger time step size for increased processing performance. This might seem really simple but the ICS could not perform this task, because the predefined error tolerance may not be appropriate. For example if the predefined error tolerance is unachievable in all the tests produced by heuristic function in ICS, whether it is because the mesh was not fine enough or the control parameters have reached their limits, etc., if the ICS had evaluated the residual graph this way, then in many situations, the evaluation function would not be able to assess which test is better. In the new control system this is not an issue. There are a couple of points that need to make clear.

First, how can the Experiment Engine be sure that the error tolerance level is being set appropriately in the control system? To answer this question, one needs to consider the Experiment Engine as whole, it is not only just a control system like ICS, since it also performs other tasks before becoming the controller for the production run. In the first stage of the Experiment Engine controlled fire simulation, the system is initially responsible for making sure that an appropriate set-up is used for the case. In particular, there is a test for the suitability of mesh which includes assessing the solution quality on the test mesh during the toughest time period (this can be identified by looking at the heat release curve) in the simulation to see if any further refinement is necessary and, if so, then local mesh refinement is applied. After the mesh testing stage of the Experiment Engine controlled simulation, it can comfortably be stated that the mesh is likely to be appropriate for the problem that is being simulated. This is a critically important part of what the Experiment Engine does, because if the mesh is later found to be not good enough during the production run, then it will be very costly to refine the mesh and re-start the processing from the beginning. Technically the Experiment Engine is able to refine the mesh at any simulation stage, including during the production run, but this is obviously not an ideal solution.

Following the mesh test, there is another test that the Experiment Engine performs before the simulation proceeds into the production stage, and this is the time step size test. This intermediate stage of testing applies different time step sizes in a range from the "largest" time step size which is "reasonable" for the given problem to the "smallest" time step size which is within the user's resource limitations. During this stage, the user's preference for the error tolerance level is also tested, and this is justified if it is not reachable within the resource limitations. After these tests, the Experiment Engine is able to determine the effect of applying different time step sizes and to establish the initial time step size to use when starting the production run.

Secondly, the new control system is designed to only look for the speed improvement when the simulation is stable and converged to the required accuracy. So full convergence is always tested during the production run and any performance oriented modifications, of the time step size, will not compromise the solution accuracy. To put these considerations into the design of a simpler, direct evaluation function, when the control system looks for changes of current time step size in order to speed up the simulation, the results of any control actions to be applied and the original time step size are evaluated by using the new evaluation function. The assumption is that at least one of these will converge to the required error tolerance level (for example, the original run before the changes must be converged, otherwise the Experiment Engine would not do searches to improve the simulation speed). If two or more of these tests have converged, then the evaluation function simply has to look at the ratio of the time step size and number of sweeps used, as the indicator for the convergence rate. So the new evaluation will always give a definite answer to whether the modified time step size was capable of giving the promised speed-up. This direct evaluation function can also help the Experiment Engine to make a decision to terminate unsuccessful speed-up attempts much earlier. For example, when the Experiment Engine decides to test for speed improvement at a scheduled time, then the test on using a different time step size (either bigger or smaller than the current one) has to be tried to see if it is possible. At this stage, the evaluation function will inform the Experiment Engine that the speed index (time step size / sweeps) calculated for the current time step size. And then the Experiment Engine will tell the convergence forecast function (being described later) to watch for both the convergence and the target (the speed index). If either the convergence is not reachable or the target is unbeatable then the Experiment Engine can terminate the test early, hence saving precious compute time.

7.5.2 A sophisticated convergence forecast function

The new evaluation function sounds quite simple but it has a sophisticated convergence forecast function – which is constantly monitoring the simulation and is able to give accurate information about the simulation state in terms of convergence and stability. This is vital so that the new control system will be able to function correctly and efficiently. It is also absolutely crucial for implementing the proposed flexible number of sub-iterations. A top-level description of the convergence forecast procedure is shown in Figure 7-1 below.

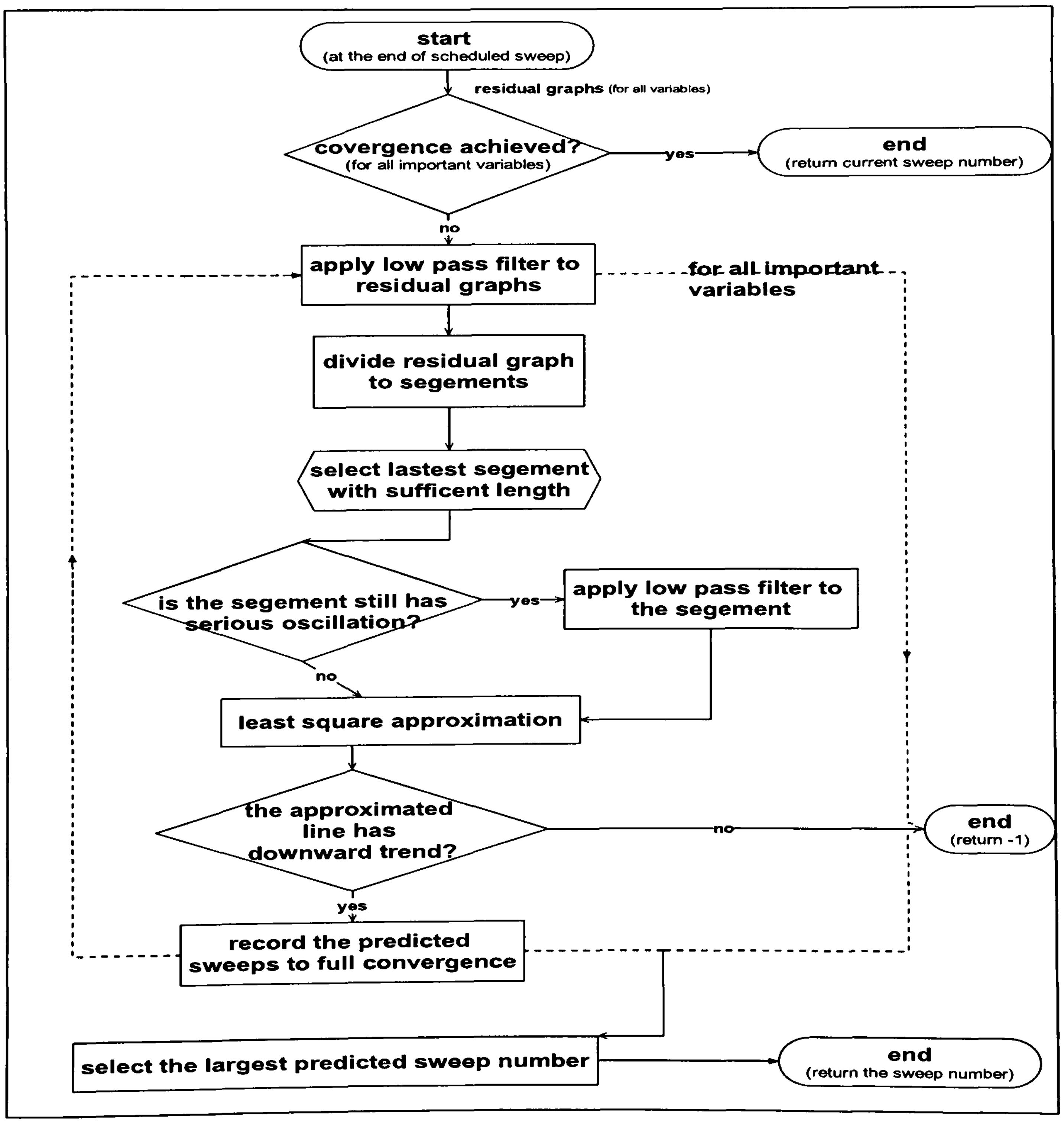

start
(at the end of scheduled sweep)
residual graphs (for all variables)
covergence achieved?
(for all important variables)
yes
end
(return current sweep number)
no
apply low pass filter to residual graphs
for all important variables
divide residual graph to segements
select lastest segement with sufficent length
is the segement still has serious oscillation?
yes
apply low pass filter to the segement
no
least square approximation
the approximated line has downward trend?
no
end
(return -1)
yes
record the predicted sweeps to full convergence
select the largest predicted sweep number
end
(return the sweep number)

Figure 7-1: Flowchart of the convergence forecast procedure

The graph extraction routines, developed in the earlier ICS research, were also carefully examined and reused wherever applicable to serve the purpose of implementing the convergence forecast function. For example, these legacy routines were used to identify the presence of the oscillations and irregularities and are still useful but have been put it into a different context. After further consulting with Experts, it is believed that the presence of the oscillations and irregularities in the residual graph are an inherent part of the simulation process, but they are providing very important information in the design of an accurate convergence forecast function. The Experiment Engine control system needs to be able to make a decision about whether the time step should be terminated early (because for example the residual graph appears flat during a long enough period and hence it is very reasonable to assume that it will not converge using the current time step size) or whether to give the time step some extra sweeps to make it converge to the required level, otherwise it would be not converged if the predefined number of sweeps is reached just before the convergence. These smart Experiment Engine control decisions have to rely on an accurate convergence forecast function which is constantly monitoring the behaviour of the residual graph. Certain mathematical techniques (e.g. Fourier Transform algorithm) are applied to smooth the graph when there are irregular oscillations, and then to help produce a more accurate global trend in the residual graph. It should also be noted that the irregularities are also an indicator of the convergence and stability status. It has been observed that, in the first 30 sweeps or so in the time step, the irregularities do not tend to cause problems since subsequent iterations tend to have less and smaller irregularities, and so do not particularly affect the convergence forecast. However when the upward trend in the irregularities persists as the iterations approach 50 or more sweeps, then the current time step will be terminated early by the Experiment Engine. It was first thought that the convergence forecast function might not be good enough in assessing this kind of residual graph, i.e. it might make mistake, because if enough sweeps were configured then the graph may recover from the irregularities and converge to the required level. But this tends to forget that the main purpose of the convergence forecast function is to minimise the chance of even a single sweep of the computation is being wasted and to identify convergence, divergence and any possible instability – as early as possible. In this sense, re-examining the example indicates that, it is actually an economical decision made by the Experiment Engine based on the information provided by the convergence forecast function, because it is of almost no benefit to hope that the time step will recover from the irregularities in that situation. Firstly, it takes some sweeps to recover from the oscillation and then even more sweeps to allow the solution to converge to the required level. If it had been classified as not converged, then the Experiment Engine would try a smaller time step size immediately to make the solution converged, and generally the smaller time step size requires comparatively fewer iterations to achieve a converged solution. Comparing with the recovery sweeps needed to overcome the irregularities, there is an overall saving in simulation time. Furthermore, if the Experiment Engine chooses to recover from the irregularities and indeed achieved convergence later, it is known that this is still a potential danger because long periods with existences of irregularities still hint that the current time step in the simulation is not stable enough. If the Experiment Engine later applies any performance oriented changes it is likely to fail thus wasting more CPU time. Therefore this is no real harm by simply classifying this sort of oscillation as not converged in the convergence forecast function. The sophistication of the convergence forecast function and effectiveness of the new evaluation function ensure the validity and efficiency of the new control system. In this way, the Experiment Engine controlled production run will be converged wherever it is possible, with near minimal computational effort.

7.6 Further minimizing the costs of the production run

In order to minimize the costs of the Experiment Engine controlled production run, it was necessary to examine the sub-modules within the new control system to identify if any further improvements could be made. Essentially, the new control system consists of two modes of action namely monitoring and control. The first one is where the monitoring module constantly monitors the solution state by continually examining the residual graphs, and evaluating the convergence forecast function (described earlier). The second module is responsible for any control decisions made by Experiment Engine and this is based on the information obtained from the first module. In this module, there are two distinct categories of actions which are applied by the Experiment Engine depending on the exact simulation state. The first category of action is to change the time step size to a smaller one to recover from simulation failure, divergence, and any possible instabilities of the simulation. As the changes of time step size are only going down in one direction, so the required level of time step size can be quickly achieved and, with the sophisticated convergence forecast function, it should not waste too many sweeps even when the decrease of time step size requires several levels of change. This is due to the fact that, if the time step size being tested is not appropriate, it will be terminated early by the Experiment Engine. So this class of simulation problems do not make the control actions behave any differently. The second category of the action is to search for speed improvement for the simulation. This test has to be performed in both increasing and decreasing time step size directions because it is believed that for a certain stage of the simulation, there will be an optimal range of the time step size, so either direction could bring speed improvement. So it is reasonable to try to change the time step size in both directions. The effects of the time step size changes will compete with each other and comparing with the simulation speed before the change, it is down to the new evaluation function to decide which change gives better performance. The Experiment Engine will always pick up the best one in terms of giving benefits to the simulation speed. However, it is known that Experts can often make the right decision about whether to increase the time step size or to decrease the time step size by accessing the simulation state without having to try the other option, so there is a possible improvement for the Experiment Engine to eliminate the unnecessary searches, thus saving more simulation time spent on tests. The following two examples are used to illustrate how the unnecessary search costs are removed by the Experiment Engine in emulating this Expert ability. The first example is when the convergence has been achieved with a very small number of sweeps, 10 sweeps in this case while using 1 second of time step size. Normally the Experiment Engine will try two different time step size of 2 seconds and 0.5 second to seek possible improvement of the simulation speed. However, using expertise, the Experiment Engine knows that the simulation achieved a very fast convergence in this situation, and this indicates that the simulation must have been very stable, so it is highly unlikely a change of the time step size to 0.5 second will have any tangible benefits. Based on this judgement, the Experiment Engine will only try time step size of 2 second and to see if the speed improvement can be obtained. The same logic is applied to the second example, when the number of iterations used to achieve the convergence is relatively large, for example 70 sweeps or more. In this situation, the Experiment Engine will only try a smaller time step size to see if speed up is possible. In addition, if the particular status gives no easy way to eliminate one of the two possible time step size changes by looking at the residual graph, the new evaluation function also has embedded logic of preferring smaller time step sizes provided that these do not degrade speed performance, (i.e. if 50 sweeps used while using 1 second of time step size and 100 sweeps used by using 2 seconds of time step size to achieve the same convergence, the Experiment Engine will favour the one using a 1 second time step size). This is due to the fact that simulations using smaller time step sizes are generally more stable than those using bigger time step sizes. This strategy should bring longer term benefits, because the simulation is more stable, thus less recovery procedures will be invoked, hence saving simulation time as well. Furthermore, because the Experiment Engine is only looking for the simulation speed improvement, when it is converged, the convergence speed index (defined by time step size used divided by the sweeps used to achieve the convergence – a measure of the simulation speed – whereby bigger convergence speed index has better simulation speed) can also be used to determine early termination of an unsuccessful search for speed. For example, if the current time step size is 1 second and 25 sweeps have been used to achieve the convergence, then the current simulation speed index is 1/25, then if the Experiment Engine decides to try time step size of 2 seconds to see if any performance gains can be made. In this situation, the current speed index of 1/25 is the target, if the search run using time step size of 2 seconds approaches 50 sweeps and still has not achieved convergence (even though it is tending towards convergence) then the Experiment Engine can terminate this unsuccessfully search immediately, because it is not going to achieve a better simulation speed (i.e. $1/25 > 2/51$). This logic can be scaled up easily. Any new search run will have a target convergence speed index, which is the current best convergence speed index, as soon as the Experiment Engine can determine that the new search run is not able to achieve a bigger (i.e. better) convergence speed index than the target one, it will stop the search run immediately. In this way, the potential wasted sweeps, due to unsuccessful searches, are kept as a minimum.

There is another factor which also affects effectiveness and efficiency of the control system in searching for speed improvement of the simulation or recovering from simulation faults. Because the control system was primarily developed to help to guarantee the simulation stability and accuracy of the solution, so any control parameter changes applied are conservative in nature, it does not perform an aggressive performance oriented search. Otherwise, even though the time step size is the only control parameter which the controls system is trying to modify from time to time, the potential search space is still big and the search costs will be very difficult to control. Instead, the control system only tries to modify the time step size to one level (i.e. up a level or down a level) at time, consequently the difference between the time step sizes in adjacent levels become very important factors which will surely affect the effectiveness and efficiency of the control system. At first, it seems to be very reasonable to apply a relative changes to the current level to a degree (i.e. 20%, 50%, 100% etc). However, for example, if the Experiment Engine adopted 100% as the degree of the difference between the two adjacent time step size, then there are times when it is difficult even for human experts to be sure that the degree of the change is appropriate or sufficient to make a difference in simulation stability or the speed performance of the simulation.

Consider, if the current time step size is 2 seconds, and the Experiment Engine decides to reduce the level of the time step size, then the new time step size becomes 1 second. This is fine because, in practice, the 1 second of time step size and 2 seconds of time step size are frequently used in different simulations according to the nature of the problem and the accuracy requirement. However, if the current time step size is 0.025, then the next level down by 100% will be 0.0125, then it is difficult to be sure if these two time step sizes will really make a difference to the simulation in recovering from a simulation fault or having the simulation speed improved. The situation becomes even more confusing if 20% or 50 % were adopted as the degree of the difference between the two adjacent levels. The uncertainty about the consequence of the changes being made by the Experiment Engine is a costly factor, because if these attempts to change, fail then the search time is wasted. Fortunately, in practice, only a limited number of different time step sizes are frequently used by Experts, for example, 10s, 5s, 2s, 1s, 0.5s, 0.2s, 0.1s, 0.01s and the time step sizes in between these values are used very rarely, therefore it is safe to assume that all these values listed can be regarded as distinct from each other and will make a difference on the simulation when applied. Therefore, the Experiment Engine will only apply these values listed should it decide to change the time step size and potential wasted search costs are largely limited thus saving simulation time.

The other consideration of the design of the new control system is to prepare for the worse case scenario, so that if or when this occurs, it is known how costly this will be. The worst scenario for the new control system is that any searches intended to speed up the simulation are not be able to recommend an action or the actions taken fail to provide reasonable period of speed improvement. These problems can be summarized as: (1) after a search is conducted, the resulting action from the Experiment Engine is to carry on with the current time step size without modification, or (2) after the modification is made, then at the next scheduled time, the Experiment Engine sets the change back to the previous one, which results in switching the time step size up and down frequently.

These issues led to the decision that the frequency in searching to speed up the simulation is at least every 10 time steps, so roughly in the worst case scenario, it will still be less than 20% (2/10, two possible wasted searching time steps used by 10 time steps of normal simulation run) of cost of an un-controlled simulation. Although the nature of changing the time step size will be best suited for large, complex and event filled fire scenarios, the physical solution changes significantly thus requiring changes to the time step size. Conversely, with a very stable simulation, the new control system will probably not be operating at its best. By limiting the costs in the worst case scenario, we can make the new control system more generic and practical in use. This means that we know that even when the new control system could not effect a speed-up for the simulation, then this is still not too costly.

The EE will definitely ensure full convergence whenever this is possible, and the user will also get the benefit of being freed from the task of continual or regular solution monitoring and still get the required quality of solution.

7.7 Other areas of improvement

As discussed in chapter 5, the local mesh refinement method developed in this research is to treat blocks as refinement units rather than have the individual cells as units for further refinement. The method used to choose candidate blocks for refinement is when the cell nodal error is greater than prescribed nodal error tolerance. Then cell quality check is performed (for each block) in all three dimensions and the total number of bad cells (per block) is recorded. When a block has certain percentage of bad cells, then it is identified as a candidate for refinement. It was realised that, for some situations, the above method is not optimal and could be improved. For example, when a bad cell clearly has an aspect ratio problem, then this should be taken into account when selecting the candidate blocks for refinement. Instead of refining the blocks related to bad cells in all three dimensions, there are times when leaving one or two blocks in different directions unrefined gives overall better refinement results. Figure 7-2 (b) shows the refinement results in two dimensions using the improved method in which the y-direction refinement is omitted. Comparing with the refinement results in Figure 7-2(a) used the old method, the refinement results using the improved method clearly have benefits of saving cells and the refined cells tend to have better topology.

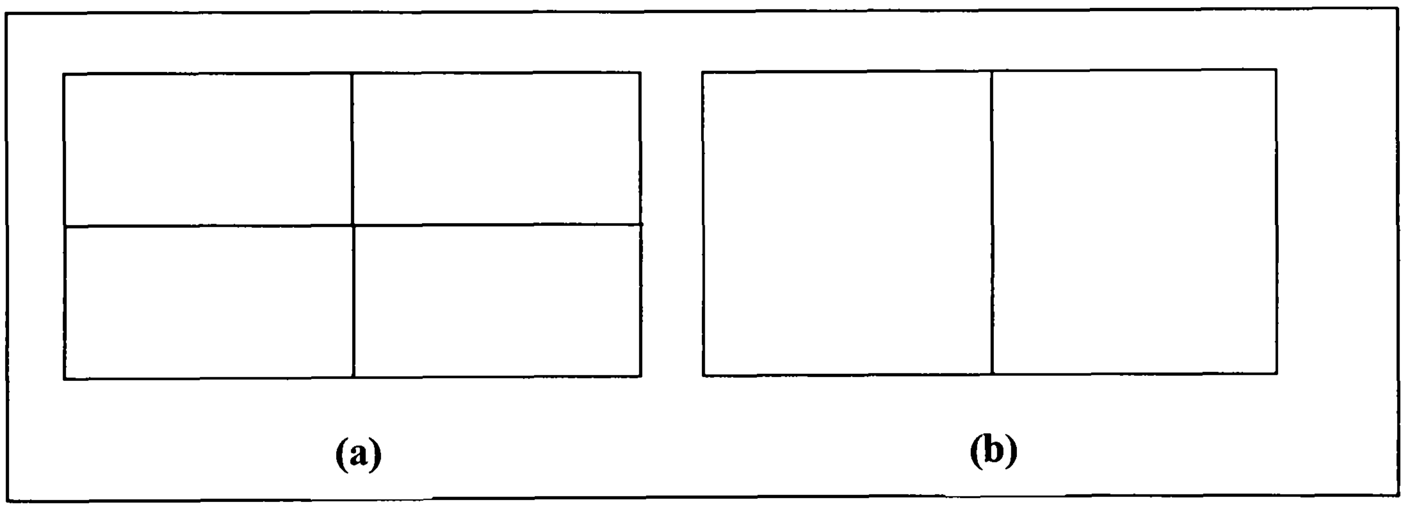


Figure 7-2, the refinement result by using the old method (a) and the refinement result by using improved method (b).

7.8 Summary of different stages in the Experiment Engine controlled simulation

With the newly developed EE control system, the whole simulation process will now go through three major stages. In each stage, the Experiment Engine control is focusing on a particular task, which is significant to the overall success of the fire simulations.

- Meshing test stage

In this stage, the Experiment Engine makes sure that the mesh is thoroughly tested and the time step size is in a suitable range. First of all, the EE selects a simulation period in which the physical conditions in the domain have substantial variations over time. This time period is identified from the variation in the heat release rate curve. A poor quality mesh would most likely have problems at the selected simulation period. Secondly, the Experiment Engine generates the initial test for the selected simulation period with a reasonable quality of mesh. If the original problem is too complicated then it may be simplified for the test. The test results are continually monitored by the Experiment Engine. If the Experiment Engine finds any problems due to the inadequate quality of the mesh, then the Experiment Engine automatically generates a new test with a refined mesh. Within the tests, the time step size may be changed to make sure that the test solution has an acceptable accuracy. This procedure is operated repeatedly until the selected simulation period has been concluded and significant flow has developed in the domain. Then the Experiment Engine can be reasonably sure that the mesh quality is good enough and the time step size is in a suitable range. Then the Experiment Engine controlled simulation will go into the next stage.

- Time step size study stage

In this stage of the simulation, the Experiment Engine tries different time step sizes to see what effects they have on the simulation stability and solution accuracy. Therefore the user preferred solution accuracy level can be also tested in this stage. If the accuracy level is not appropriate (either it is too low or unachievable within the resource limitation), then it will be adjusted to a suitable level according to the results from the time step size study. Finally, an appropriate initial time step size for the production run is selected and a suitable global error tolerance is established. At this stage, the Experiment Engine is able to automatically generate the production run with appropriate set-up and mesh which was thoroughly tested in the first two stages for the given scenario.

- Production stage.

  In this stage, the control system embedded in the Experiment Engine, is constantly monitoring the solution state, and takes any necessary actions to ensure that the solution converges to the preferred error tolerance whenever it is possible and is able to recover from any simulation instability. The EE also monitors for opportunities for possible simulation speed improvement so as to produce a good quality solution in the shortest time.

In conclusion, the Experiment Engine is able to take control of whole fire simulation process automatically and produce a required quality of solution with almost no human intervention.

[Chapter summary]

This chapter describes various enhancements that have been investigated and incorporated in the prototype Experiment Engine, with a particular emphasis on the new control system. These investigations have centred on allowing the Experiment Engine to emulate the Experts' controlling activities throughout the whole simulation process. The EE controlled simulation runs through three major conceptual stages during the simulation process. In the initial meshing stage, the mesh is thoroughly tested under difficult conditions (i.e. where the heat release rate changes most rapidly). This is probably not the only time that the mesh could have problems. For instance, when a hot flow (e.g. a plume or ceiling jet) is constrained by geometry, or an event occurs that opens up a new flow path (e.g. door opening or window break) will also potentially cause stability problems. In the initial mesh testing stage of running the Experiment Engine, these conditions are not tested and it is not known if or when they will occur, in advance – unless they are timed events. When the mesh is found to be inadequate for the required solution accuracy then local mesh refinement is applied to refine the mesh. At the end of mesh testing stage, an appropriate set-up, mesh and suitable range of the time steps size is established. In the time step size study, all possible time step sizes – within a suitable range – are tested and the optimal time step size (in terms of achieving the required accuracy) with best speed performance is chosen and used as the initial time step size in the production run. As previously mentioned, some of the problems encountered in the production run – for some complex scenarios – are almost impossible to test-for in the initial meshing test and/or the time step size study stages of the EE controlled solution process. For example, the mesh obtained from the mesh test stage may still not be of sufficient quality for some of the events that will occur later in the simulation. In order to compensate for this issue, it was decided that the EE should be made to continuously monitor the solution state of the simulation and make any necessary control decisions to change the control parameters in response to the physical condition changes (e.g. heat release rate change, transient events etc.) in the domain whilst also checking for any possible performance gains. If it is found that a problem persists (i.e. the problem is most likely due to inadequate quality of the mesh) despite control parameter changes made by the EE, then it will still be possible to refine the mesh automatically during the production run (although it is a very costly strategy to change the mesh at this late stage of the simulation, since the production run will have to restart from the very beginning, this might be the only resolution for a persistent problem).

In summary, the investigations towards extending the control offered by the EE system, have demonstrated that it is possible to give fully automatic and robust control of the production run. Furthermore, the prototype EE has shown that it is able to obtain results to the required accuracy without the need for user intervention and, rather surprisingly, in shortest possible time.

8 Investigating the performance and reliability of the EE concept for representative FFM studies

[Chapter Overview]

In Chapter 7, a number of potential enhancements and extensions were investigated and tested within the Experiment Engine prototype. The extension to the scope and architecture of the EE meant that it is potentially able to take control of the whole fire simulation process. The design for the EE has focussed on providing fully automatic operation whilst ensuring solution robustness. In this chapter, three significant fire simulation examples are studied in order to test if the EE does demonstrate the required effectiveness, efficiency and robustness. In order to test the Experiment Engine thoroughly, the selected scenarios compare the EE controlled simulation with that of an Expert user manually controlling the simulation (i.e. the Expert user makes changes to the control parameters at pre-defined simulation times). These are further compared with a non-controlled simulation (i.e. an attempt is made to use the sensible "default" control parameters, throughout the simulation). Detailed analysis of the results is presented with particular emphasis on the reliability/fault-recovery, simulation speed and achieved accuracy.

In order to start to test the validity and capabilities of the EE, it was first applied to the control of a moderately complex apartment fire scenario. In the second scenario, the Experiment Engine is used to control the simulation of a corridor cable fire experiment. Finally, the third test case – has been designed by an Expert user to have many transitional changes that are likely to cause simulation instabilities, but which is representative of the sort of simulation complexity that Fire Engineers would be likely to perform, This scenario models three-storeys of an apartment building geometry with fire spread between floors and a number of transitional events throughout the simulation. This challenging simulation is analysed to determine the Experiment Engine's robustness, efficiency and benefits, in particular, when it is applied for simulating more complex fire scenarios.

8.1 Introduction

For fire cases which are inherently stable, under the default control conditions and using the recommended mesh, it is simple for the user to start the simulation process and expect to get a reasonable solution. However, for the majority of fire cases, things are typically not so straight forward, especially for real life/complex fire scenarios, which commonly involve heat release rates changing rapidly in short periods of time and transient events like windows breaking, doors opening etc. These events can occur as a result of the changing physical conditions in the domain and hence, at unpredictable times. These events and conditions can very easily "break" the solution process. By this, it is meant that the solution either diverges and produces physically unrealistic results (oscillatory, misdirected or unreasonably large flows/pressures etc.) or that certain physical properties can become unrealistic (e.g. obtaining negative densities). Conventionally, the Expert user would have to make educated "guesses" as to when and where these issues will occur (i.e. where and when simulation stability and solution accuracy will be problematic) and how to use pre-configured control parameter changes to "calm" the rapidly changing physical conditions in the domain and safeguard the simulation stability and ensure that the solution accuracy is preserved. Such issues mean that the control parameters are typically set to quite conservative values, which can seriously affect the overall simulation time. Once the simulation becomes stable again, the conservative control parameters are usually re-justified/relaxed. The first scenario used to test the robustness and effectiveness of the Experiment Engine was designed to reflect the sort of conditions that would be expected in a real life fire scenario with sufficient complexity to be of use to fire safety engineers and fire fighters. The EE controlled simulation was then compared with the simulation performance using pre-configured time step size changes set by the Expert.

The EE primarily modifies the time step size, during the controlled simulation, and this is likely to be more suitable for complex and unstable cases (i.e. those where the physical conditions change rapidly and frequently, thus requiring changes to the time step size). However, as a generic control technique, it is also important that the EE can efficiently control stable cases without imposing excessive extra computational costs. In order to test the efficiency of the EE, a fairly stable corridor cable fire experiment is chosen as the second example. Since the cable fire scenario is inherently quite stable, the default control parameter settings, in *SMARTFIR*E, should be sufficient to solve the problem. Hence the performance of the EE controlled simulation can be compared with the non-controlled simulation that uses the default time step size. In addition, it is also relatively easy for the Expert to determine an optimal or almost optimal set of control parameter settings for this case by conducting a simple coarse mesh trial. So the EE controlled simulation can also be compared with the simulation using the pre-configured "optimal" control parameter settings (whereby the time step size is changed at pre-configured times). Finally, the solution produced by the EE controlled simulation is compared with the results yielded by the non-controlled simulation and experimental data in order to examine the effect of the control changes on the physical results.

Because of the nature of control techniques deployed by the Experiment Engine, it is believed that as the simulation complexity increases, so the validity, effectiveness and performance of the EE system will improve. A further test was formed, mainly based on the first test, but with considerably increased complexity. This final case has been configured with many transitional effects and extreme conditions throughout the simulated period. The EE controlled simulation was again compared with non-controlled simulation for assessment of fault recovery and simulation time improvement. A so called "gold standard" run was also carried out so that high quality results could be used to assess the accuracy of the results produced by both the EE controlled and non-controlled simulations.

8.2 Apartment fire case

8.2.1 Model setup and boundary conditions

8.2.1.1 The Geometry

The test case geometry is typical of what might be found in a section of a residential apartment building. It consists of a main living room, a kitchen, a dining area and several en-suite bedrooms. There is also a stairway connecting the apartment floor to floors above and below and to provide an escape route in the event of fire (when lifts cannot – generally – be used). The fire is assumed to start in the living room.

The geometry was first specified in the Scenario Designer (a CAD like floor-plan entry tool) before being loaded as a 3D model into the *SMARTFIRE* Case Specification Environment where the detailed physics and transitional specifications were configured.

The apartment is represented as a 13.4 m x 12.6 m x 2.6 m tall rectangular volume. Figure 8-1 shows a 3D wireframe view of the building taken from the Case Specification Environment (CSE) of *SMARTFIRE.*

8.2.1.2 Vents

The scenario has the following openings and vents:

- There are two openings to the outside of the building. One is an opening through the ceiling at the head of the main stair and is approximately 2.4 m x 3.4 m in size. The second vent is a small vent located in the bottom right hand corner of the kitchen wall and represents an air brick or grille as would typically be found in a kitchen. This provides pressure relief during the early stages of the simulation.
- There are two windows in the living room, where the fire starts, also connected to the outside of the building. These are initially closed and are assumed to be broken (and fully removed) when the adjacent air temperature reaches 773 Kelvin (i.e. 500 degrees Celsius).

- The rooms are inter-connected by doors. These doors are all assumed to be open during the entire simulation except for the main apartment door which connects the apartment entrance hallway to the rest of the building. The main door is initially closed except for a small leakage area which represents the small air gap around the door. This leakage allows realistic escape of smoke to the main hallway. The main door is opened at a time of 630 seconds when the fire fighters are deemed to have reached the area just outside of the apartment.

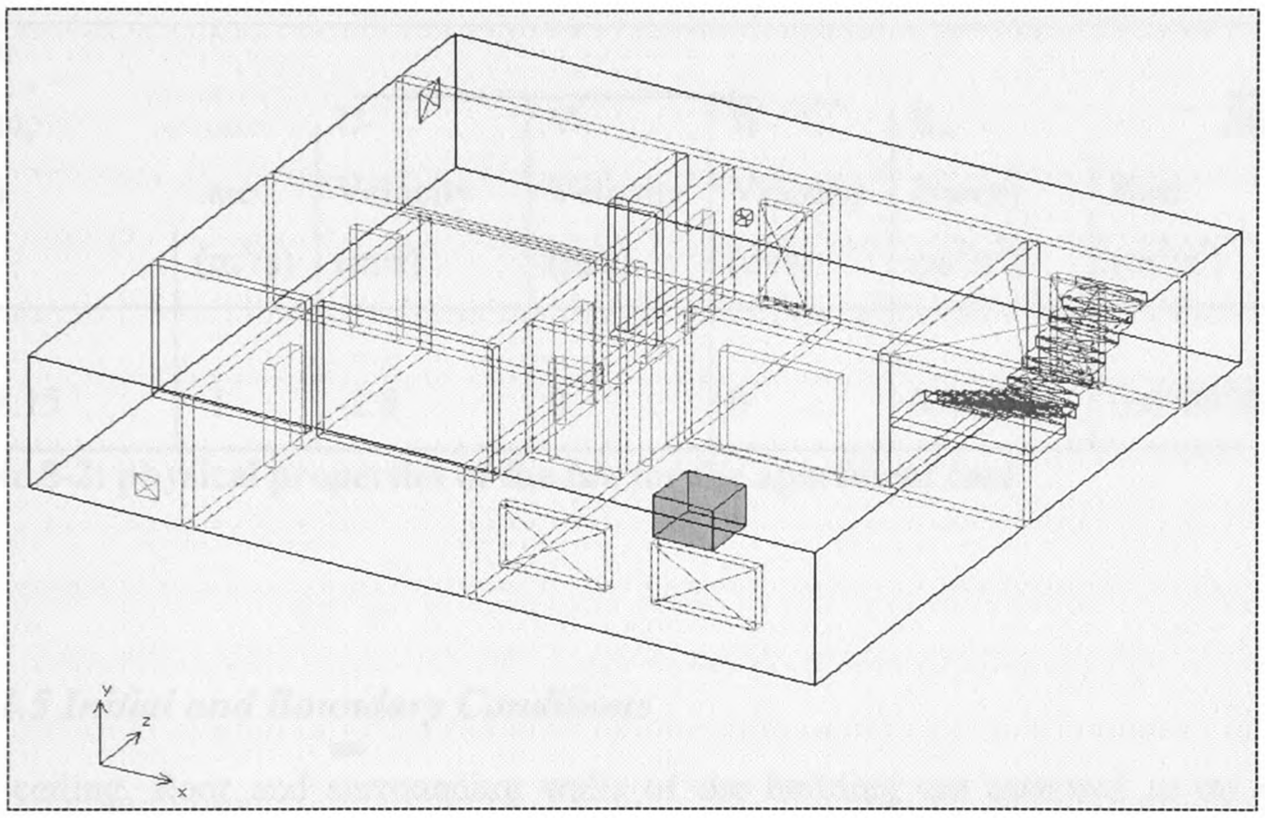


Figure 8-1: 3D view of the apartment geometry

8.2.1.3 Material properties

All obstructions dividing the buildings to several rooms are assumed to be composed of default wall material. The input data given in Table 8-1 is taken from the material database provided by *SMARTFIRE.*

Default Wall Material				
Conductivity (W/m K)	Specific Heat (J kg /K)	Density (kg/m^3)	Lamina viscosity (Pa s)	Thermal expansion coefficient (K^{-1})
0.69	840	1600	1E+10	1E-10

Table 8-1: obstruction properties for the apartment case

8.2.1.4 The fan

An extracting inlet (representing a fan duct) is located at the end of the main corridor. The physical property of the fan is shown in Table 8-2. The fan is initially inactive but is activated when a smoke detector detects a certain concentration of smoke. The extracting fan is intended to provide a smoke management solution which will help to keep the corridor clear of smoke and prevent smoke from spilling into the stairs – which would hinder people using the stairs to exit the building.

The fan						
Temperature (K)	Flow rate (m^3/s)	U-Velocity (m/s)	V-Velocity (m/s)	W-Velocity (m/s)	Kinetic Energy (m^2/s^2)	Dissipation Rate (m^2/s^3)
288.15	-1	-2.8	0	0	0.015431	0.0106504

Table 8-2: physical properties of the fan for the apartment case

8.2.1.5 Initial and Boundary Conditions

The ceiling, floor and surrounding walls of the building are assumed to be non-conducting material (i.e. heat is not conducted through the material and the timescale of the simulation is such that the heat loss to the walls will not be significant). Table 8-3 shows the summary of the initial and boundary conditions.

Wall thickness (m)	Ambient Temperature (k)	Initial Temperature (k)	External pressure (Pa)	Material inside domain
0.2	288.15	288.15	101325	Standard air

Table 8-3: initial and boundary conditions for the apartment case

8.2.1.6 Fire specification

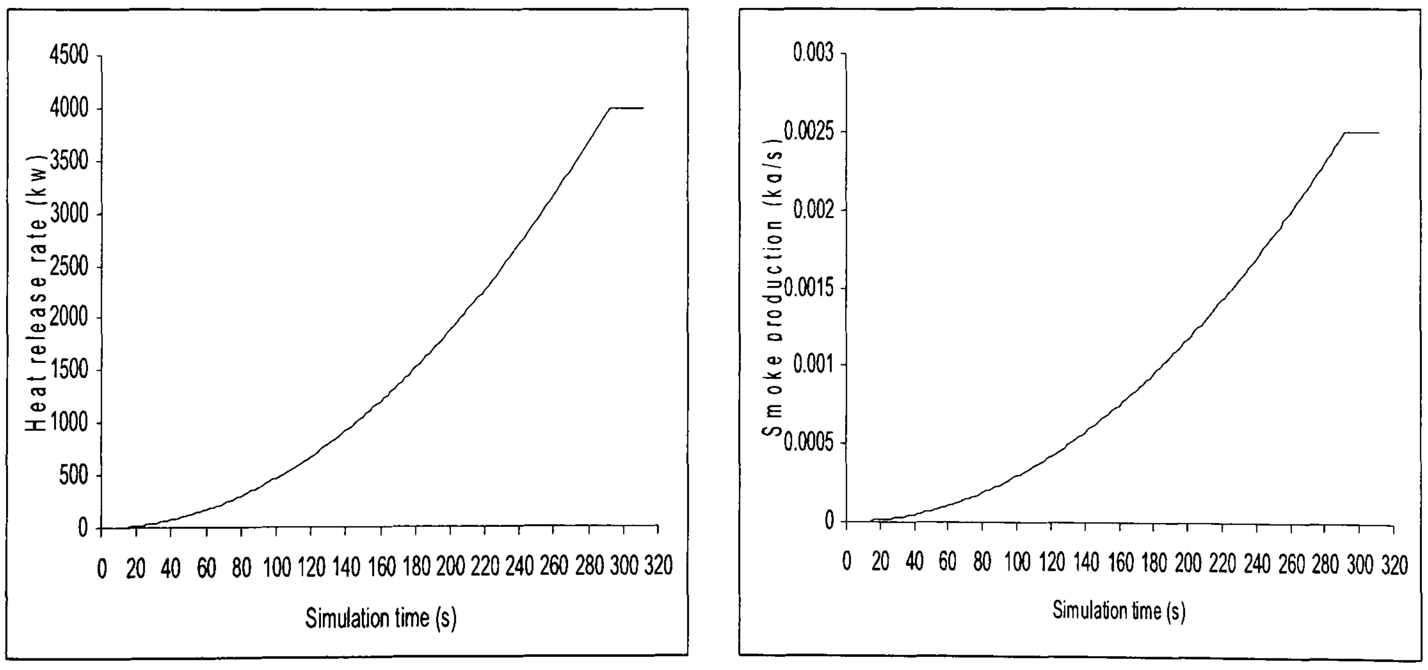
The fire starts at a time of t=0 s of the simulation and grows according to a fast t-squared curve (with c = 0.0469 kW/s^2) to a constant peak value of approximately 4 MW at a time of t=292 s as shown in Figure 8-2(a). The fire source was represented as a rectangular object of size 1.0 m x 0.8 m and a height of 1.0 m with the specified heat release rate located on the floor in the centre of the living room. The heat release rate remains at a constant value after the peak output time. Smoke was calculated from the heat release rate, the heat of combustion of the fuel and the yield of smoke production as follows,

Y_{smoke}= *Mass smoke/Mass fuel*

Fuel burnt rate = Peak heat / Heat of combustion

Therefore, *Smoke rate= Peak heat/ Heat of Combustion* * Y_{smoke}

The fuel type is assumed as wood, and the resulting smoke production curve is shown in Figure 8-2(b).



(a) (b)

Figure 8-2: the heat release rate curve (a) and the smoke production curve (b) of the apartment complex case

8.2.1.7 Physical models used

Flow is enabled. Combustion is disabled for simplicity, since the extra solved and calculated variables would extend the simulation time but would not particularly change the outcome of the results or the effectiveness of the Experiment Engine. The radiation and smoke models are also both active.

8.2.2 The mesh

Some of the meshing considerations were made even before starting to set up geometry, for example, the wall thickness was set to 0.2 m and care was taken with object positioning to avoid the presence of very thin cells. This is seen as a normal requirement for the user to perform since an automated alignment system cannot know which object positions are critical and which can be moved without influencing the ultimate solution. These preparatory steps make the construction of a "high" quality mesh much easier without having to use an overly large cell budget. Initially the mesh was generated automatically by the Case Specification Environment within *SMARTFIRE*. The meshing parameters, that were used, are those which are suitable for a single floor of building. The mesh was then improved and thoroughly tested by the Experiment Engine. This meant, for example, that the aspect ratio rules were enforced and the mesh was tested during the toughest simulation period which was identified as when the fire has the maximum rate of heat rise. These checks help to ensure that the mesh is adequate and that there would be no problems caused by having an inappropriate meshing solution. This was also necessary since an overly fine mesh would take considerably longer to get to a solution. The overall cell budget was 89460 cells (i.e. 63 * 20 * 71 cells). The resulting mesh was visually checked and approved by the Expert user. Figure 8-3, 8-4, 8-5 shows the mesh in x-, y-, and z-directions respectively.

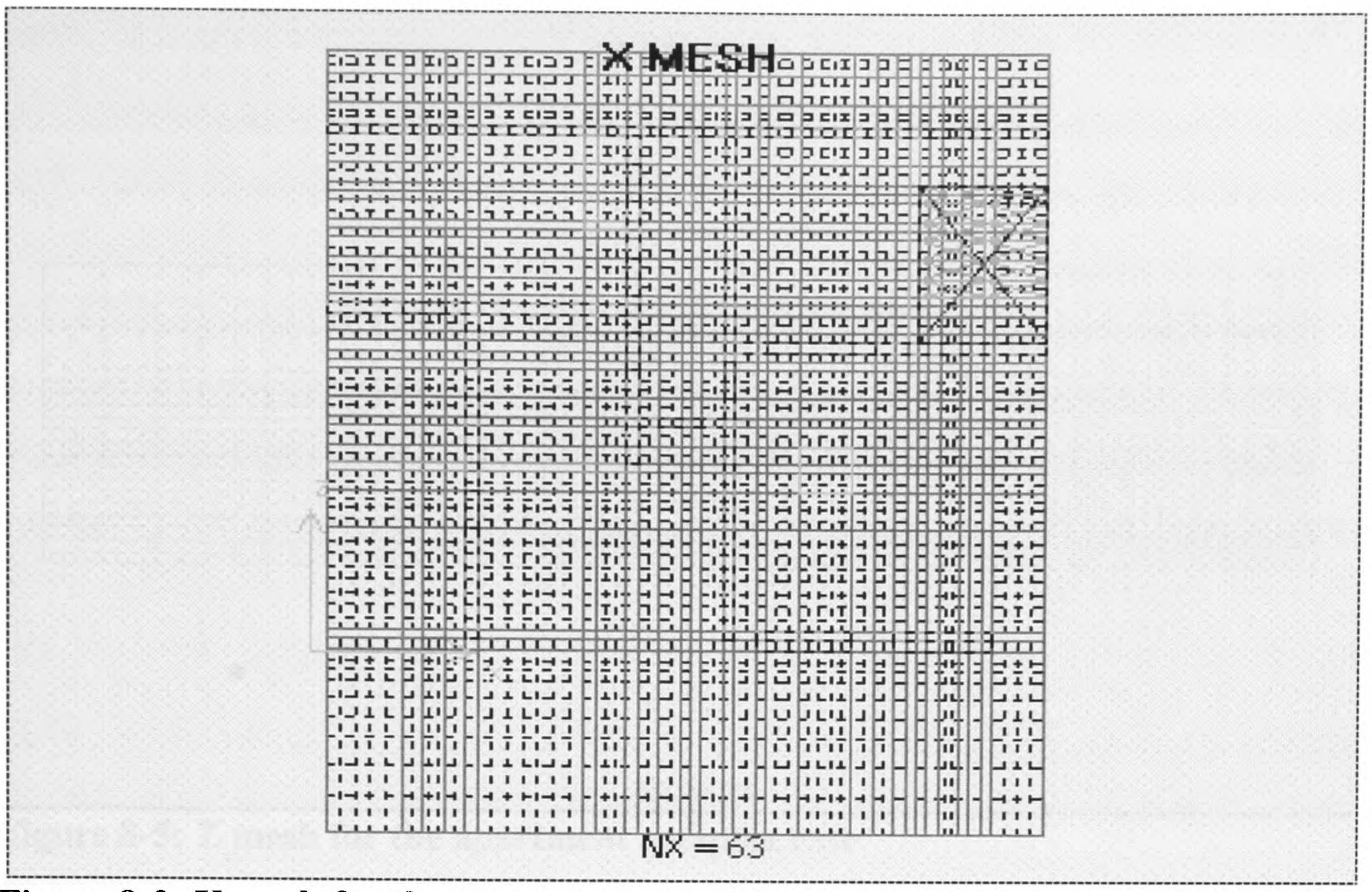


Figure 8-3: X mesh for the apartment complex case

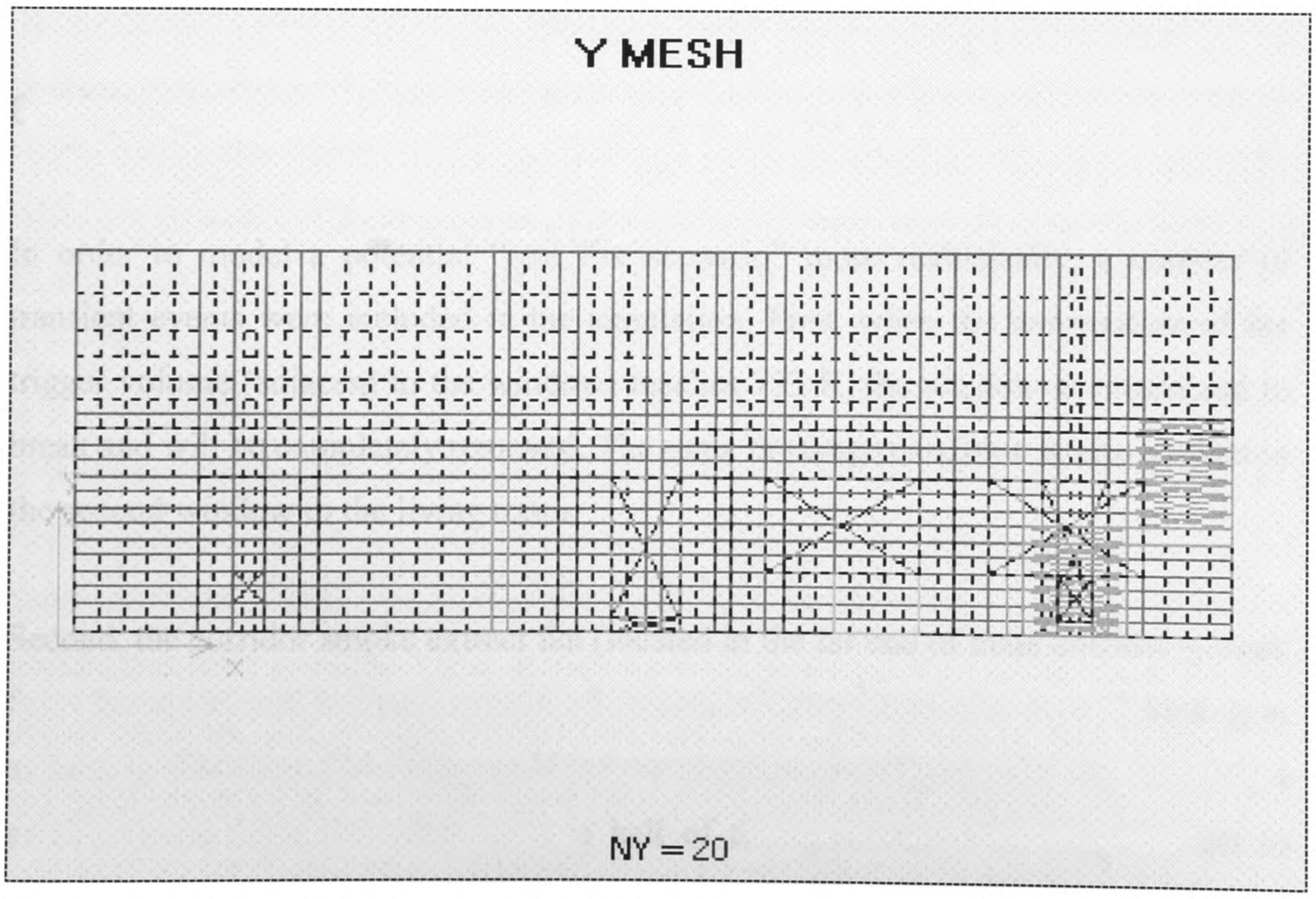


Figure 8-4: Y mesh for the apartment complex case

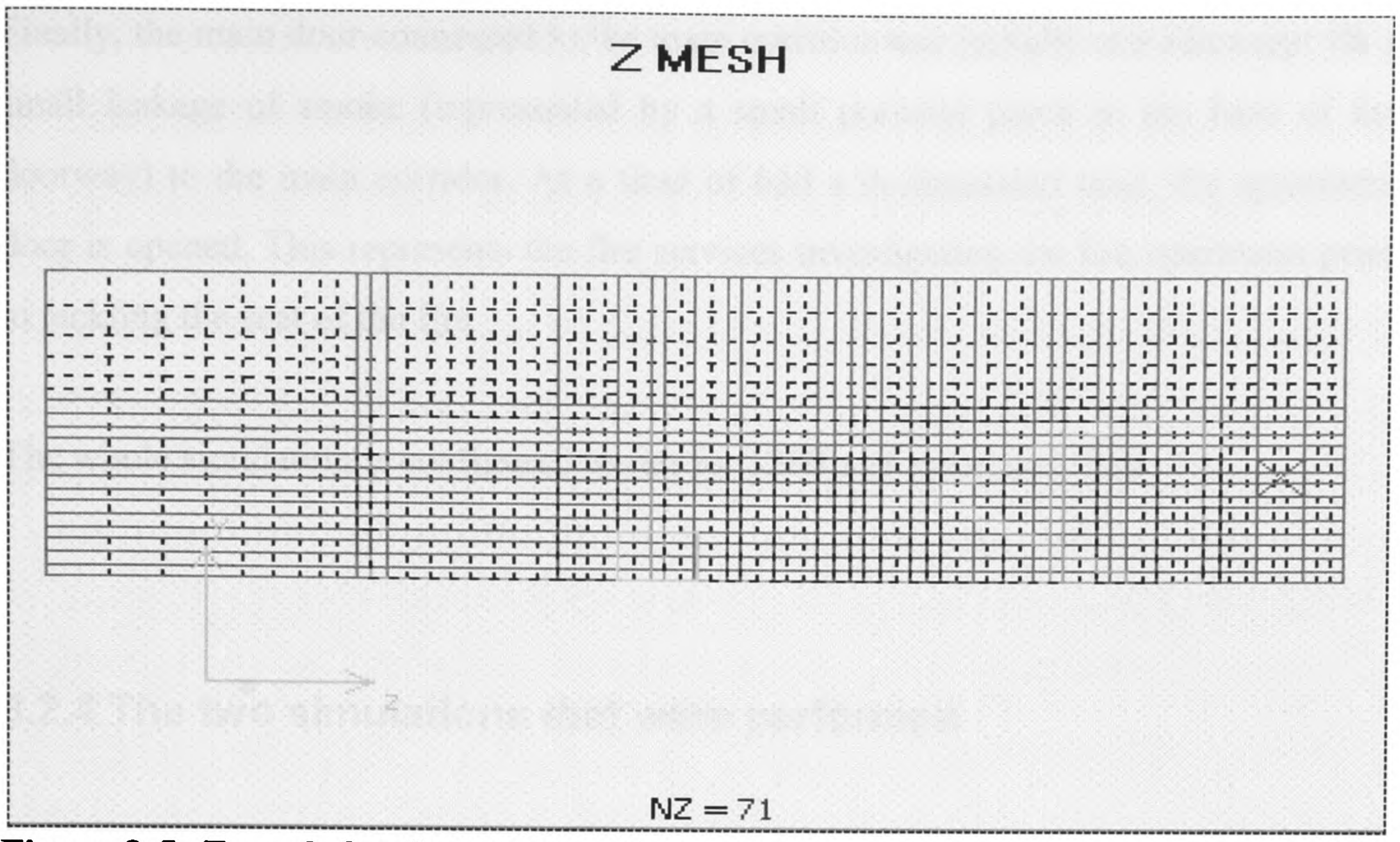


Figure 8-5: Z mesh for the apartment complex case

8.2.3 Transient events

In order to model a potential "real fire scenario" more realistically, a number of transient events were included in the simulation. First, when the temperature of the trigger volume (adjacent to the window) reaches 773 K, the window is considered to break and will be completely removed. The same breakage condition is also applied to the second window in the living room.

Second, the corridor smoke extract fan (located in the far end of main corridor – away from the stairs) will have a extraction flow rate of 2.8 m^3/s (as shown in Table8-2) at activation. The fan will be triggered by the activation of a trigger point (representing a smoke alarm) located in the entrance hall of the apartment. The trigger is set to activate when a prescribed mass fraction of smoke is detected at the trigger location.

Third, the door that leads to the stair is opened at a time of 600s. This represents the arrival of the fire fighting services at the fire floor.

Finally, the main door connected to the main corridor was initially closed except for a small leakage of smoke (represented by a small porosity patch in the base of the doorway) to the main corridor. At a time of 630 s in simulated time, the apartment door is opened. This represents the fire services investigating the fire apartment prior to tackling the seat of the fire.

The whole simulation is configured to run for 1200 s of simulated time.

8.2.4 The two simulations that were performed

In order to examine the validity, efficiency and effectiveness of the Experiment Engine, two simulations were performed. It was found that the complexity of this scenario meant that the default control parameter settings were not able to progress the simulation through all of the transient events. It was found that a number of the events caused the solution to become unstable and for the solution to diverge, leading to physically unrealistic solutions. This led to the decision to omit the simulation using the *SMARTFIRE* default control parameter settings. Instead, the first simulation used a set of pre-defined time step sizes as prescribed by an Expert user (whereby the time step size is pre-configured to change at certain times). The second simulation was completely controlled by the Experiment Engine during the entire simulation, without any manual intervention after the essential problem set-up completed by the Expert in the Case Specification Environment. These two simulations used exactly the same problem set-up and the same mesh (as described previously). The two simulations also used the same global error tolerance of 1e-4 (except for the less stable variables, e.g. MOMENTUM, in which the error tolerance was reduced to anything below 0.01), which both simulations were able to achieve.

Figure 8-6 below shows that the simulation proceeds as expected with the hot gas and smoke from the fire spreading throughout the apartment. The left window in the fire room has been broken at around 135s of the simulated time, due to the high temperature in the room, and drawing hot gas and smoke out of the window. The extract fan sucks some smoke out of the main corridor but prevents the smoke from spilling into the stairs.

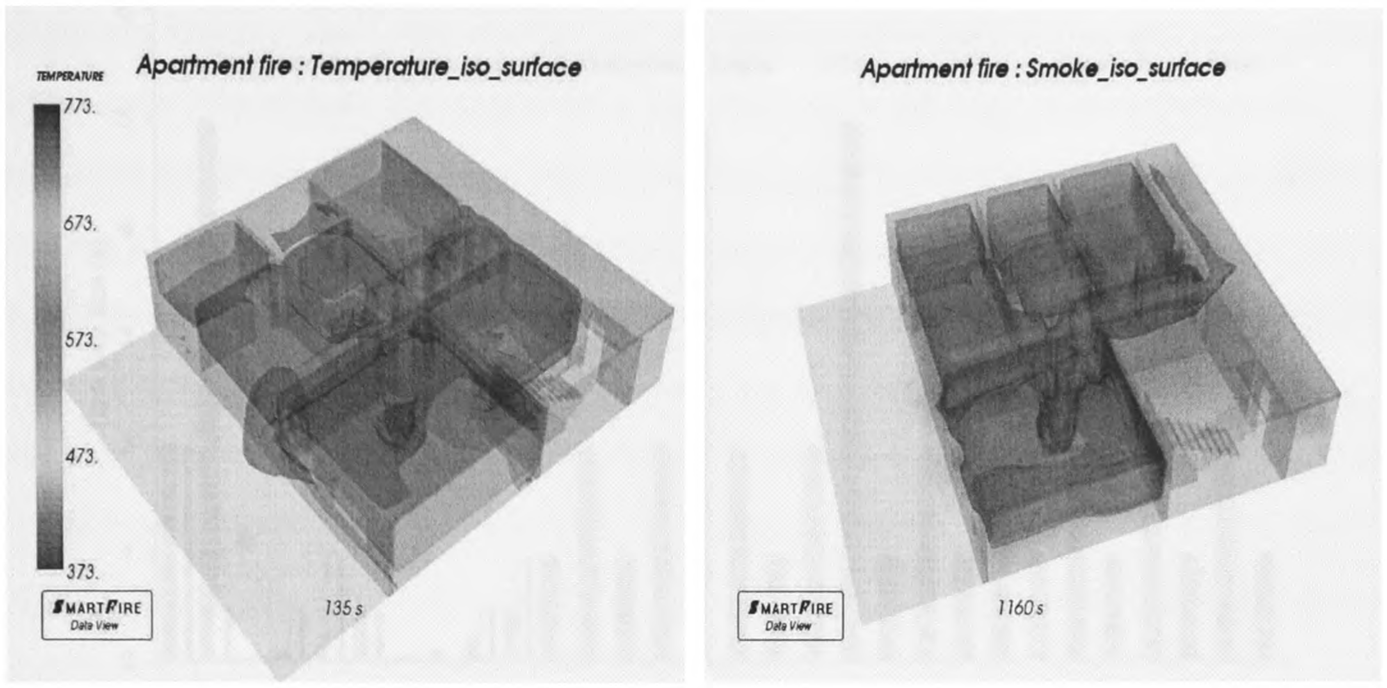


Figure 8-6: Temperature iso surface at 135s and smoke iso-surface at 1160s for apartment fire case based on the Experiment Engine controlled simulation.

8.2.4.1 Changes to the time step size in response to the transient events

The timeline of transient events in the simulation is shown in Table 8-4. The time step size changes made by both the Expert user and the Experiment Engine, during the simulation, are compared and shown in Figure 8-7.

Real simulation time (s)	Transient Events
58	The fan is activated by the smoke alarm
133.2	first window breaks
157.33	second window breaks
600	stair door opens
630	apartment door opens

Table 8-4: the time line of the transient events based on the results of the Experiment Engine controlled simulation

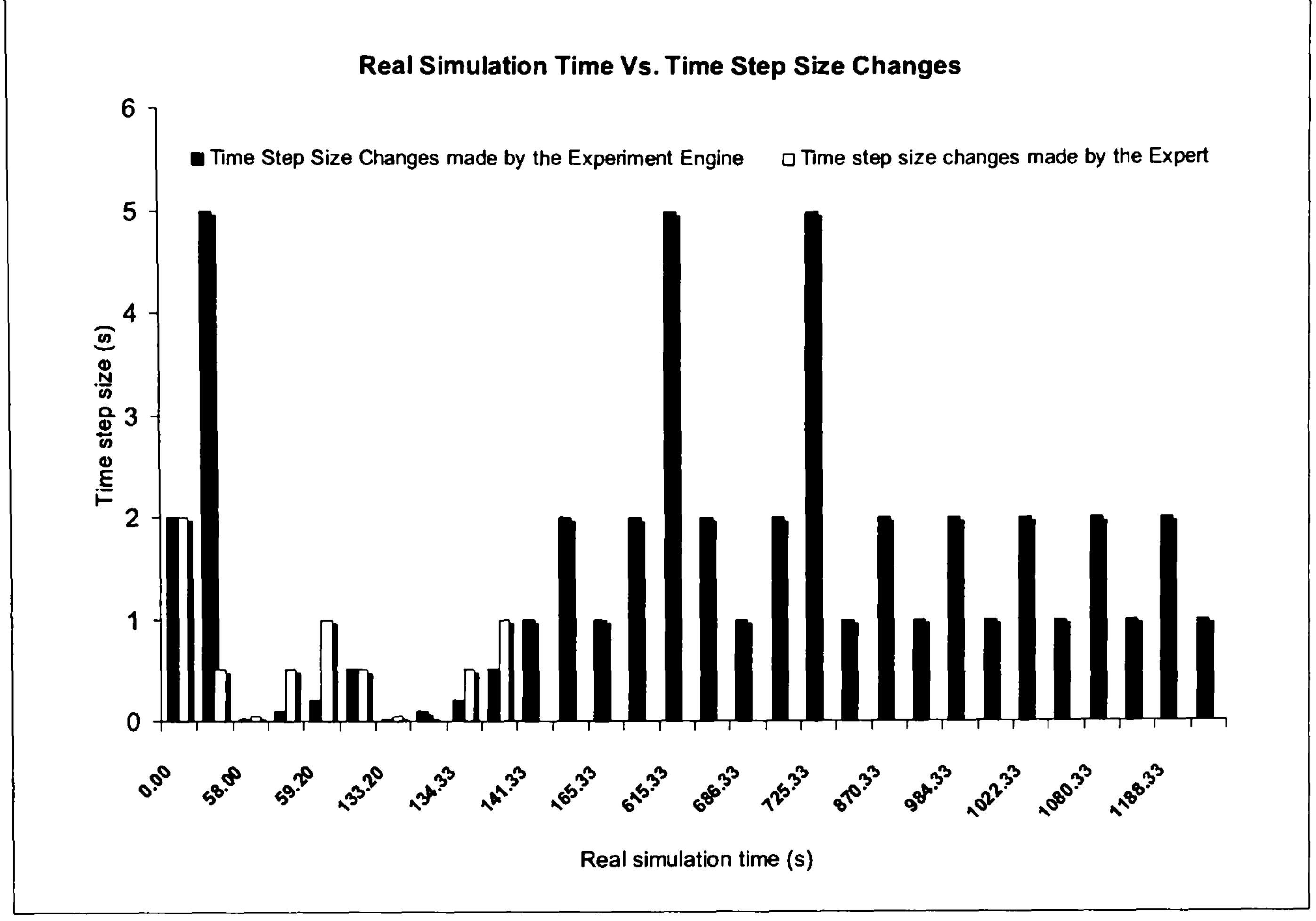


Figure 8-7: Comparison of time step sizes used during the simulation by the Expert (hollow) and the Experiment Engine (solid)

Figure 8-7 shows that the simulation controlled by the Expert user and the Experiment Engine are similar in terms of when the time step size changes are made (i.e. in response to the transitional events detailed in Table 8-4 above). However there are two significant differences between the two simulations. First, the Expert user had to learn the timeline of the transient events, and to learn what to use for the corresponding time step sizes, through an initial trial run. In this case, the Expert user had to do two restart runs on the trial, since the trial failed twice due to the Expert using an "incorrect" time step size when the fan activated and when the first window breaking event took place. Needless to say, the Expert user spent considerable time and effort on the trial. Nevertheless, the Expert user was able to learn from the failures and was eventually able to run the simulation successfully. While the trial and error search is an integral part of the Experiment Engine, so the Experiment Engine was able to successfully control the simulation without any difficulty. Secondly, the Expert user's decision to make the time step size changes was in direct response to the transient events rather than being performance oriented. It was also observed that the Expert user's response was based on the result of qualitative analysis rather than quantitative analysis. Conversely, the Experiment Engine knows nothing about the transient events (except for their impact on the solution quality) and merely kept trying to ensure that the simulation stability and solution convergence were maintained. However, the simulation stability and solution convergence are the indicators of physical changes in the domain, so the Experiment Engine was able to respond to the transient events in the same way that the Expert user did (please refer to Table 8-4 for the actual timeline of the transient events). Furthermore, when the simulation stability and solution convergence were acceptable, the Experiment Engine was able to look for possible performance optimizations. This is the reason why the Experiment Engine was observed to make more time step size changes than the Expert user did, as shown in Figure 8-7. This is a natural response for the Experiment Engine, when the solution seems to be progressing well.

It should be noted that the performance based approach, operated by the Experiment Engine, is generic to many different simulation scenarios, since it does not have to know what is actually happening physically in the simulation to still be able to take the simulation under control. It was observed that, because the performance based approach does not rely on the knowledge of detailed physical changes in the domain directly, the EE sometimes undertook performance-enhancement experiments when it was not strictly sensible to do so (for example, in Figure 8-7, between t=870 s to t=1200 s, the Experiment Engine still kept trying to look for performance gains by making time step size changes). With hindsight, it is known that there were no significant physical changes in the domain during the time, so any performance gains are short lived and this has resulted in time step size control changes that oscillate up and down quite frequently. This is likely to actually degrade the overall performance. Although the Experiment Engine has built-in mechanisms to restrict the excessive costs but in the situation like this, the Expert user definitely has the upper hand, since the Expert will keep the most appropriate time step size when there are known to be no significant physical changes during a particular time period. The above observations can be the starting point for further improvement of the Experiment Engine. These will be discussed in chapter 9.

8.2.4.2 Speed comparison

In the previous section, it was pleasing to observe that the Experiment Engine controlled simulation finished successfully without experiencing any convergence difficulty or failure. Furthermore the control changes made by both the EE and the Expert user, to the time step size, showed a similar pattern in response to the physical changes in the domain and, in particular, to the transitional events during the simulation. The following section, examines both simulations more closely in order to compare the performance in terms of simulation speed.

The speed comparison is based on the total number of iterations needed by both of the simulations. The actual CPU run time was also recorded but it is considered to be less reliable because the two simulations were run on different machines, therefore there could be issues relating to the performance and load on the two PCs even though, ostensibly, they had the same CPU speed and memory capacity. For the Expert user controlled simulation, the "duration" is simply defined as a total number of iterations required to complete the simulation. However, for the Experiment Engine controlled simulation, the two speed measurements used are:

- Real simulation length – total number of iterations performed during the simulation (both "normal" and search-induced)

- Effective simulation length – the number of iterations performed during the simulation excluding search iterations, which were unable to produce any benefit to the simulation and are therefore considered to be wasted iterations.

Figure 8-8 presents the convergence history for the two simulations and the effective iterations performed by the Experiment Engine is also shown to examine the efficiency of the trial and error searches used to improve the performance. The corresponding improvement factors are shown in Table 8-5.

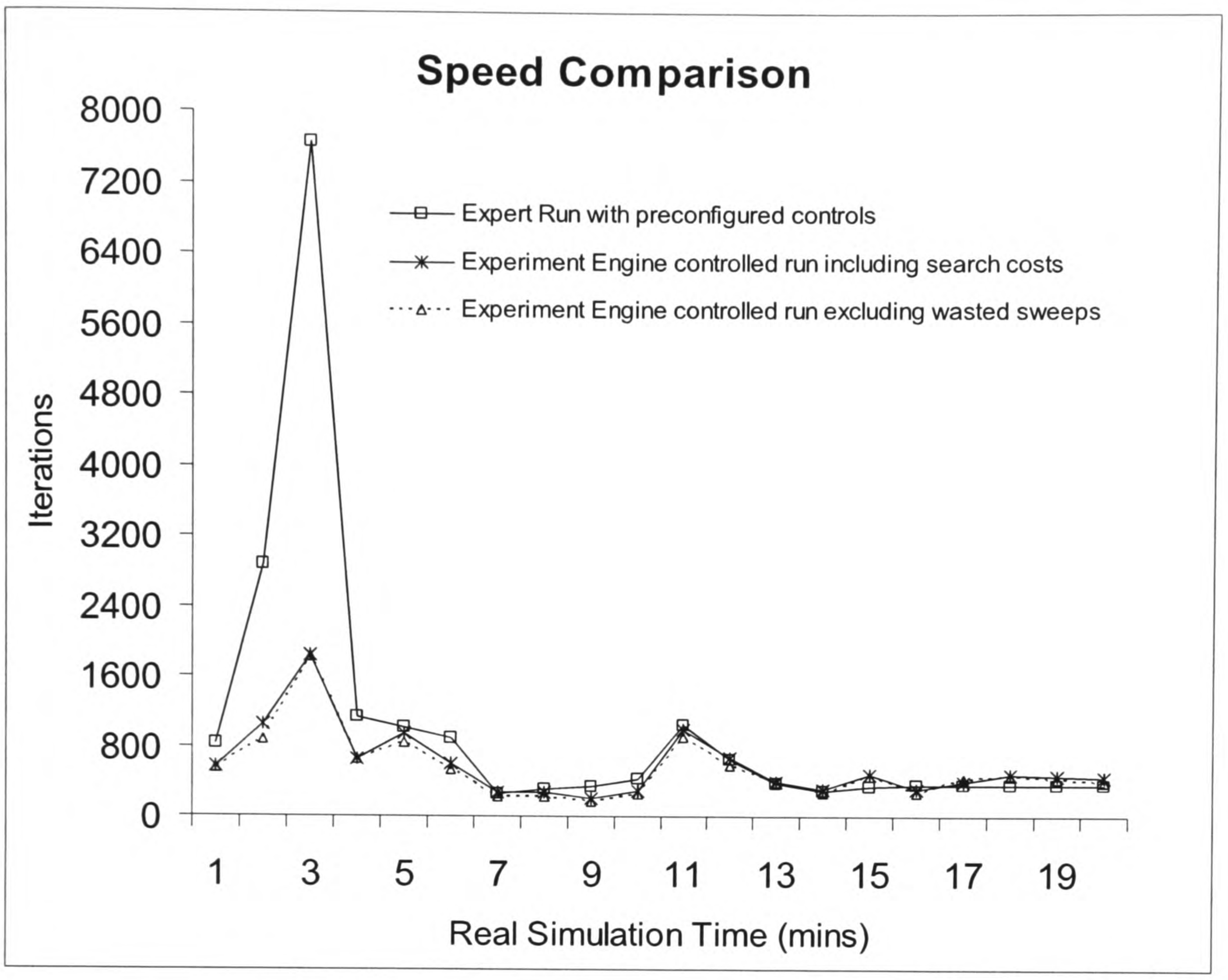


Figure 8-8: Convergence speed history of the simulations (apartment building case)

Type of control	Total simulation iterations	CPU Run time(s)	Speed up on iterations	Saving on CPU Run time
Expert pre-configured run	20274	195037s (54h 10m 37s)	/	/
Experiment Engine Controlled run including search costs	11693	139538s (38h 45m 38s)	42%	28%
Experiment Engine Controlled run excluding wasted iterations	10801	/	46%	/

Table 8-5: the performance gains made by the Experiment Engine

The convergence history clearly shows that the simulation controlled by the Experiment Engine has produced significant performance gains compared to the simulation using pre-configured controls set by the Expert user. This shows that the

Expert user control decisions were rather intuitive and based more on experience than on performance, thus the Experiment Engine controlled simulation is able to out perform the Expert user pre-configured simulation. This is especially true in the first quarter simulation period, when the most destructive events to the solution took place (i.e. when the fan was activated and when the two windows were broken). The time step size chosen by the Expert user was able to deal with these events, but it was sub-optimal due to a lack of quantitative analysis. However, during the last quarter of the simulated time, when the simulation has calmed down, Figure 8-7 shows that the EE was still trying performance oriented changes. As a result, the Experiment Engine controlled run in this period used slightly more iterations than the Expert user pre-configured simulation in the same period. Despite this deficiency of the EE, it was still able to achieve a greater than 40 % speed up in this case as shown in Table 8-5. The EE was proved to be efficient in controlling this case, since few iterations were actually wasted in searches for speed-up of the simulation. In fact, over 92% (10801/11693, from the figures in Table 8-5) of iterations are doing useful work during the EE controlled simulation.

It should be noted that, as the 28% CPU time (wall clock) saving is quite different to the 42% saving on when considering iterations (as seen from Table 8-5). This is most likely because first of all, as previously stated, the two simulations were run on different machines, therefore there could be issues relating to the performance and load on the two PCs even though, ostensibly, they had the same CPU speed and memory capacity. Secondly, that an iteration does not necessarily have a fixed amount of time or compute resources since it also has sub-iterations which can end early if convergence is reached for a particular solved variable. Thirdly, when the EE is performing experiments or recovering from a simulation fault, it relies on automatically saved restart files by the EE, so that after the evaluation of the changes made, the simulation can proceed with the best possible control parameters by using one of the saved restart file. Therefore, the EE controlled simulation tends to use more CPU time than the normal run of the same number of iterations, because of the extra "saving" and "loading" actions that the EE simulation has to perform.

8.2.4.3 Fault recovery and accuracy assurance

The EE proved to be very effective and efficient in recovering from divergence and restoring simulation stabilities. As mentioned earlier, in this case, the initial trial runs, performed by the Expert user, demonstrated that there were several periods where the solution struggled to converge and, indeed went on to crash the solution. In the EE controlled run, the quality of convergence was restored by reducing the time step size and when there were serious instability problems, the EE was able to determine the required time step size very quickly, hence simulation stability was easily restored. In the test case, all the time steps in the Experiment Engine controlled run were converged to the error tolerance level (i.e. PRESSURE was converged to range of 1e-5, Velocity and other calculated variables converged to range of 1e-3 in the case).

8.3 The Corridor Cable Fire Experiment

The corridor cable fire experiment used to test the Experiment Engine is described as follows.

8.3.1 Description of the experiment

The geometry for the experiment is a U-shaped corridor with sub corridors of length 19.1 m, 8.5 m, and 17 m (See Figure 8-9). The corridor was also divided into five sections by four high soffits, of depth 0.4m, at locations A, B, C and D. At the inner corner between the first two parts of the corridor there is an opening (2m x 0.4m) starting at 2.0m high which is connected to a 4 m x 3 m room. The height of the corridor and the room is 2.4 m. The corridor was constructed in a modular system with non-combustible promatect boards with a conductivity of 0.189 (at $20^{O}C$) and a thickness of 0.01m.

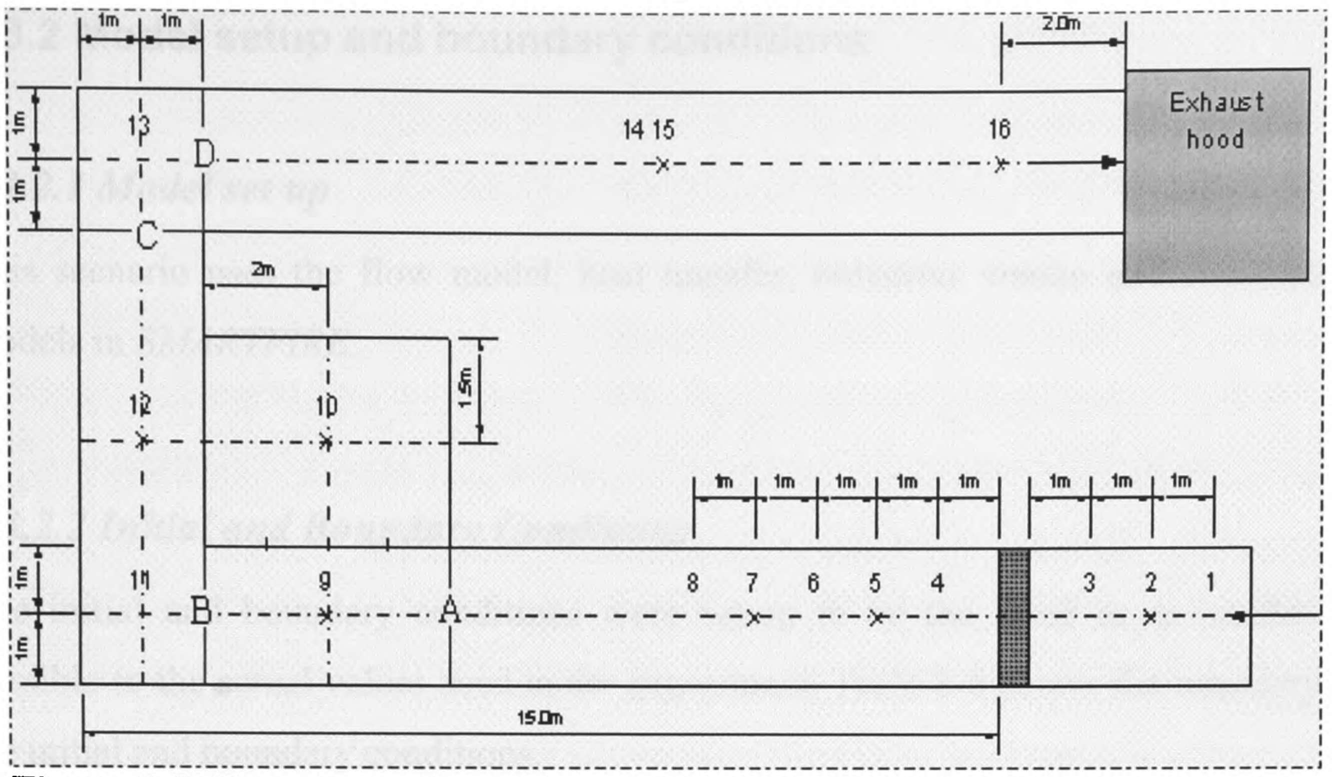
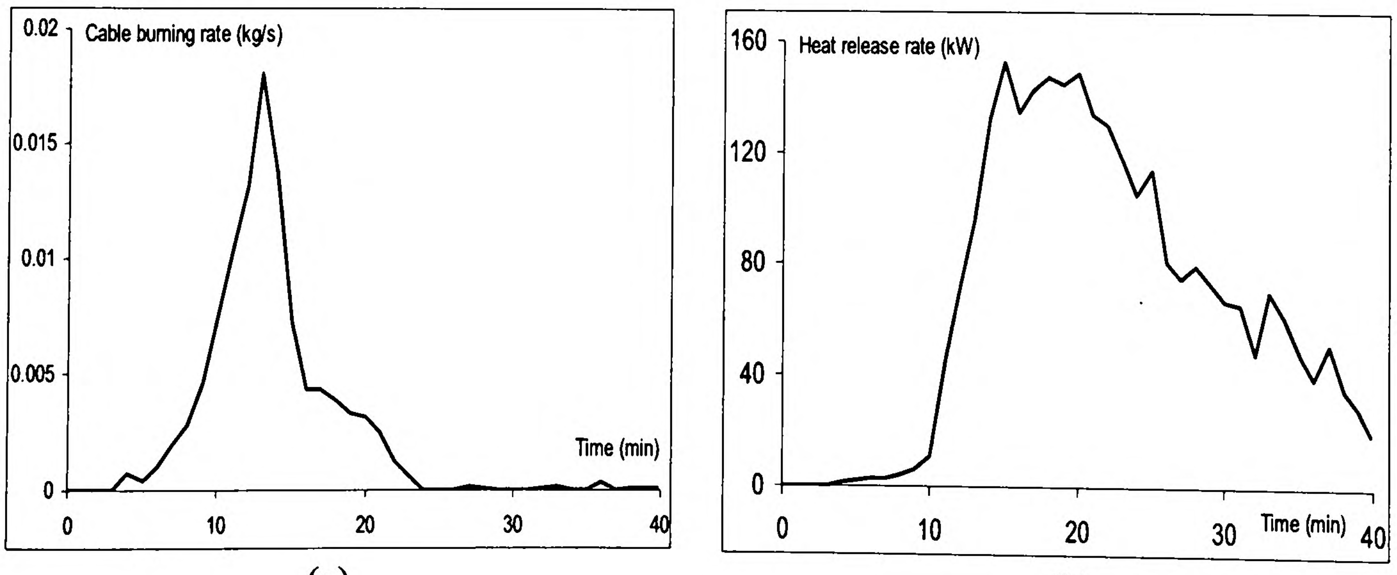


Figure 8-9: Top views of experimental geometry and measurement locations

Temperatures were measured at locations 1 to 16 as shown in Figure 8-9. Thermocouples were placed at heights of 2.2m, 2.0m, 1.5m, 1.0m and 0.5m, and load cells were used to measure the mass loss of the cables. The fire effluent was collected via a hood to a calorimeter. The test selected here is one with a NHMH [Fagrell-98] cable tray fire. Aside from the main fuel, i.e. NHMH cables, there is a propane burner generating 30 kW of heat during the whole experiment time. The measured cable loss rate and total heat release rate are shown in Figure 8-10.



(a) (b)

Figure 8-10: Measured cable burning rate (a) and total heat release rate (b)

8.3.2 Model setup and boundary conditions

8.3.2.1 Model set up

This scenario uses the flow model, heat transfer, radiation, smoke and combustion models in *SMARTFIRE.*

8.3.2.2 Initial and Boundary Conditions

The initial and boundary conditions were set-up to be the same or as similar as possible to the actual values used in the experiment. Table 8-6 shows the summary of the initial and boundary conditions.

Wall thickness (m)	Ambient Temperature (K)	Initial Temperature (K)	External pressure (Pa)	Material inside domain
0.01	293	293	101325	Standard air

Table 8-6: initial and boundary conditions (corridor cable fire case)

8.3.3 The mesh

The mesh for the scenario was manually constructed by the Expert user, to make sure that the simulation will not be affected by any possible meshing issues. The overall cell budget of the resulting mesh was 129920 cells (i.e. 56 * 20 * 116 cells) and Figures 8-11, 8-12 and 8-13 show the top, front and side view of the mesh respectively.

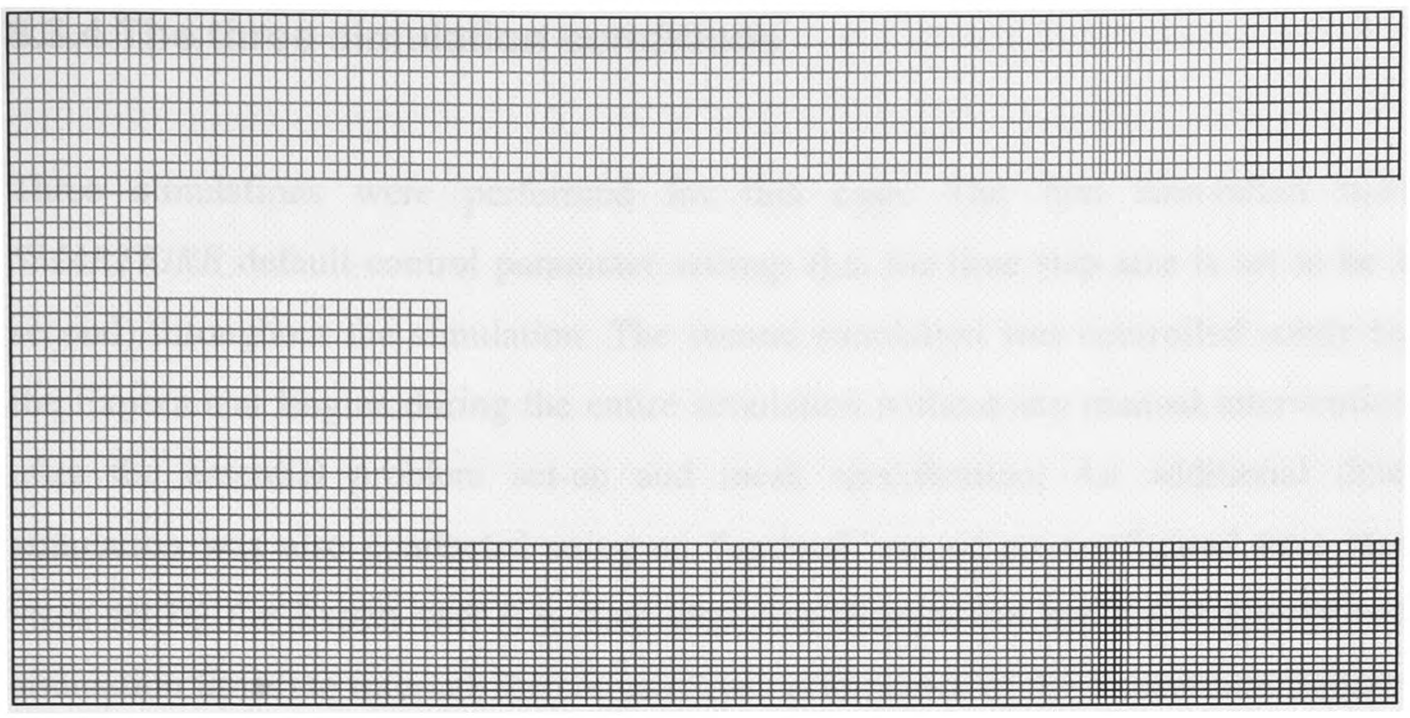


Figure 8-11: Top view of the mesh for the corridor cable fire case

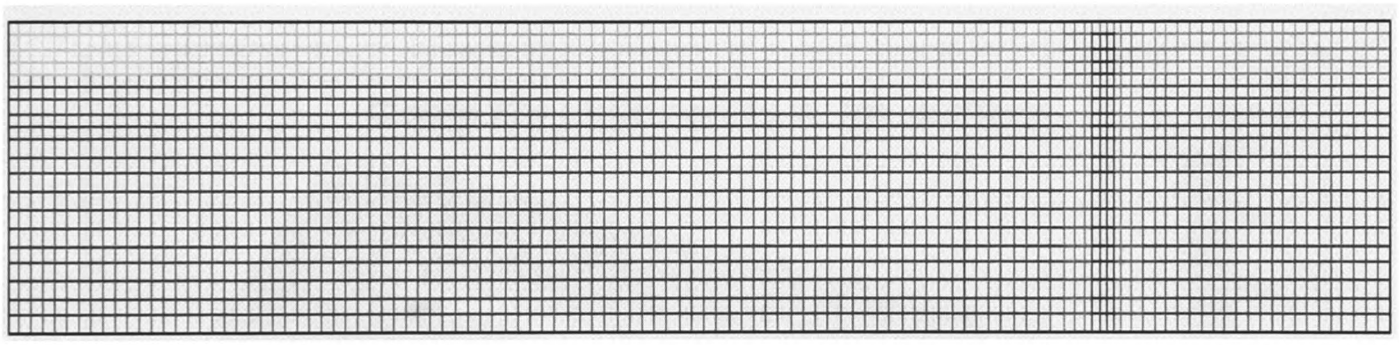


Figure 8-12: front view of the mesh for the corridor cable fire case

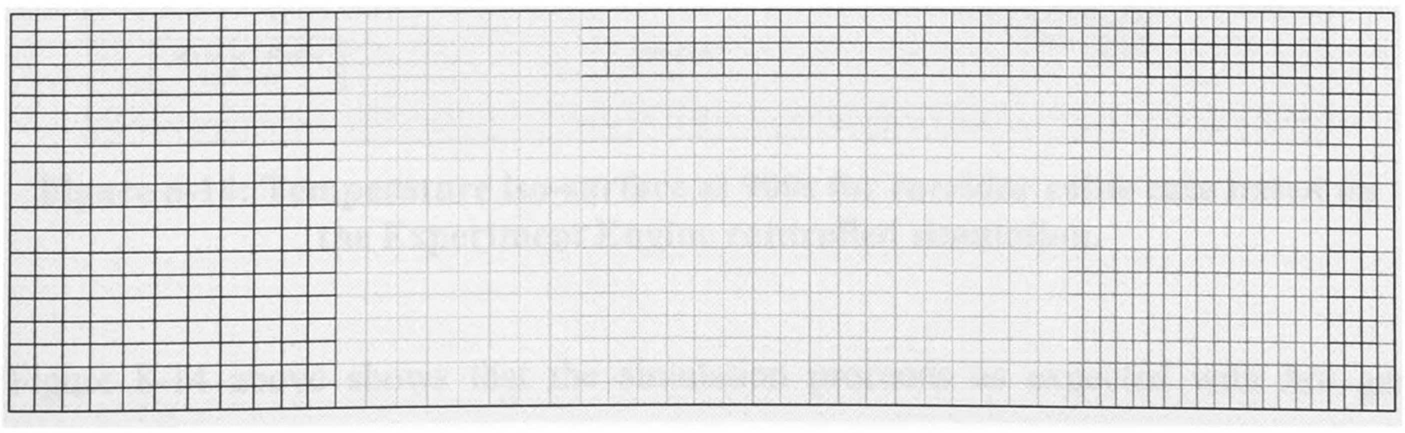


Figure 8-13: side view of the mesh for the corridor cable fire case

8.3.4 The three simulation conditions

Three simulations were performed for this case. The first simulation used *SMARTFIRE* default control parameter settings (i.e. the time step size is set to be 1 second) throughout the simulation. The second simulation was controlled solely by the Experiment Engine during the entire simulation without any manual intervention after the essential problem set-up and mesh specification. An additional third simulation was also conducted using an "optimal" set of pre-configured time step sizes set by the Expert user (i.e. time step size changes are applied at pre-defined simulation times) to measure the comparative performance of the Experiment Engine. These three simulations used exactly the same problem set-up and the mesh described previously. The three simulations also used the same global error tolerance of 1e-4 (except for the less stable variables, e.g. MOMENTUM, in which the error tolerance was relaxed to anything below 0.01), which all of the simulations were able to achieve.

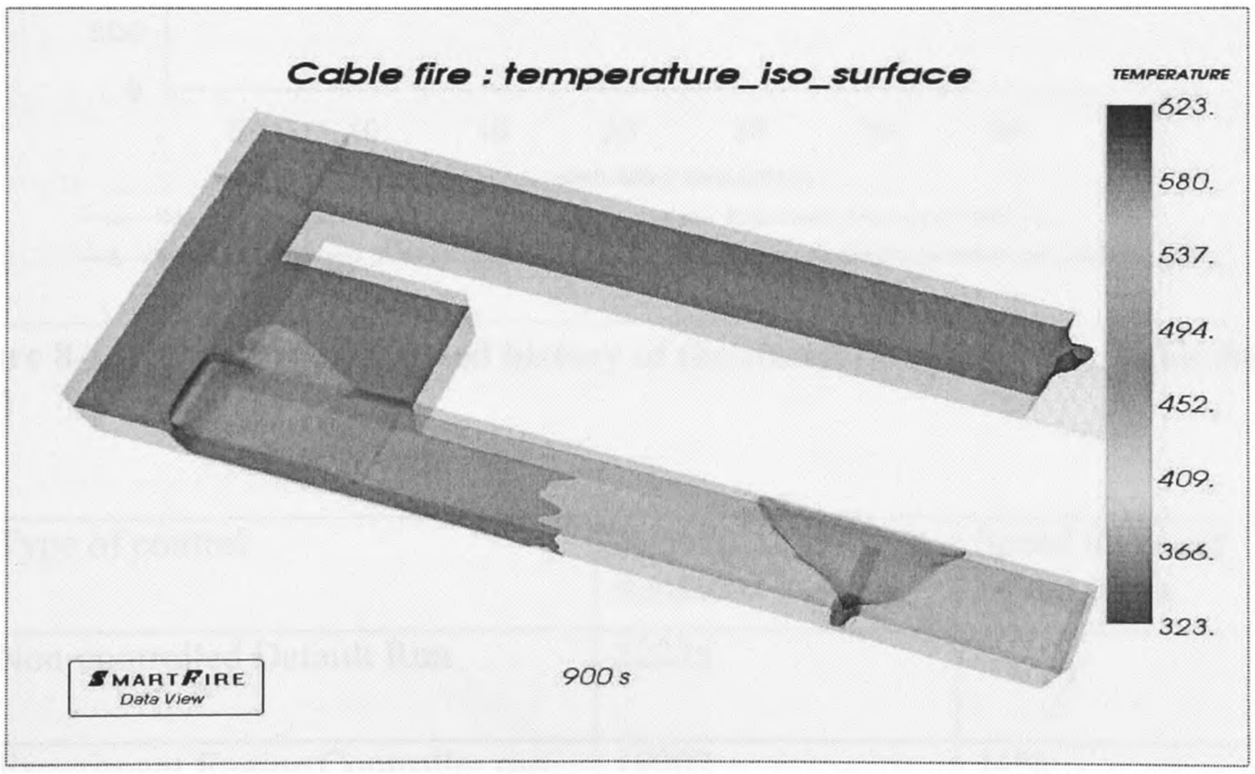


Figure 8-14: Temperature iso-surface at 900s for corridor cable case based on the Experiment Engine controlled simulation.

Figure 8-14 above shows that the simulation proceeds as expected with hot gas initially flows along the first section of the corridor (where the fire starts), then takes a gentle turn into the open room, before proceeding to the remaining sections of the corridor, and is drawn towards the only opening at the end of the corridor.

8.3.4.1 Speed comparison

Again the speed comparison is based on the total number of iterations used by the simulations. Figure 8-15 presents the convergence history for the three simulations and an additional curve representing the iterations, performed by the Experiment Engine controlled simulation, without the search costs to show the theoretical improvement to the simulation speed. The corresponding improvement factors are shown in Table 8-7.

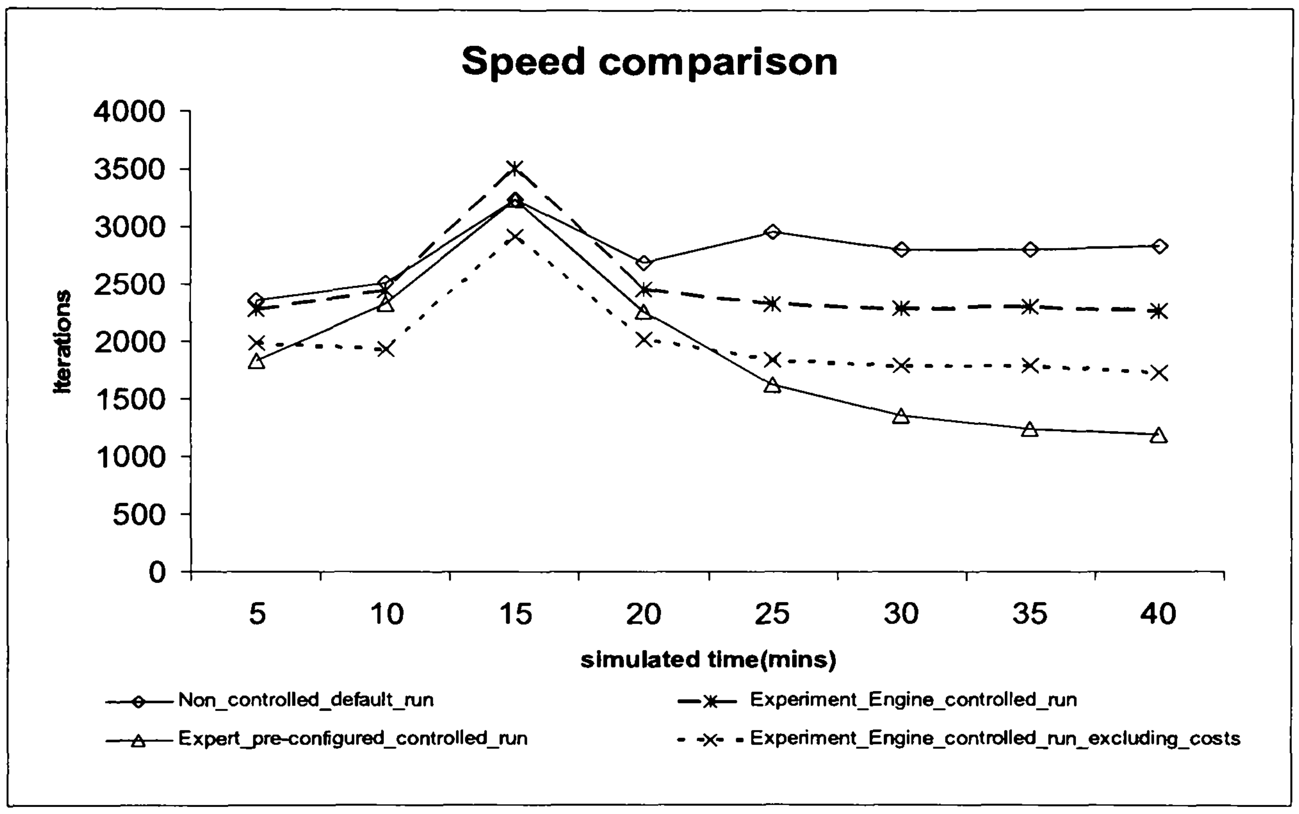


Figure 8-15: Convergence speed history of the simulations (corridor cable fire case)

Type of control	Simulation iterations	Speed up based on iterations
Non-controlled Default Run	22235	/
Experiment Engine Controller run including search costs	19902	11%
Experiment Engine Controlled run Excluding the search costs	16044	28%
Optimised controlled run set-up by the Expert user	15081	32%

Table 8-7: Performance improvement comparison (corridor cable fire case).

The convergence history graph shows that the best performance (in terms of improved simulation speed) was the one with pre-configured control set by the Expert user. This result was no real surprise, because for such a stable case (i.e. the peak heat output was only 220kW and there were no transient events), it was relative easy for the Expert user to work out an almost optimal set of control parameters based on a fairly simple trial. In addition, there were no excessive costs for testing various time step sizes (i.e. only one coarse mesh trial was needed). However, it was observed that the theoretically savings made by the EE, shown in table 8-7, closely match the Expert user performance (i.e. 28 % for EE controlled against 32% for the Expert user). The Experiment Engine control was able to emulate the Expert user's ability to control the simulation, and in this case performed reasonably well. A theoretical saving of 28% is not very large. The real savings are quite small because of the relatively high search costs. This is especially true for such a stable case (i.e. for stable cases, the potential for wasted searches is large because the stable physical conditions in the domain normally do not require any time step size management). Even with all search costs considered, the Experiment Engine controlled simulation still managed to save 11% comparing with the non-controlled default run. This again proved that the control technique employed by the Experiment Engine are reasonably efficient (please refer to the notes on minimizing the costs of the production run section in chapter 7).

8.3.4.2 Accuracy assessment

The accuracy of the Experiment Engine controlled simulation was assessed by comparing the results with those produced by the non-controlled simulation and the experimental data as shown in Figure 8-16. The figure shows the temperatures in the near fire region at position 5 from t=0 seconds to t=40 minutes. It clearly shows that both the Experiment Engine and the non-controlled simulations produce physically sound results, which are in good agreement with the experimental data. Furthermore, the results produced by Experiment Engine controlled simulation are virtually identical to the non-controlled simulation when using the same mesh and error tolerance. This indicates that the control technique deployed by the Experiment Engine has negligible impact on the final results.

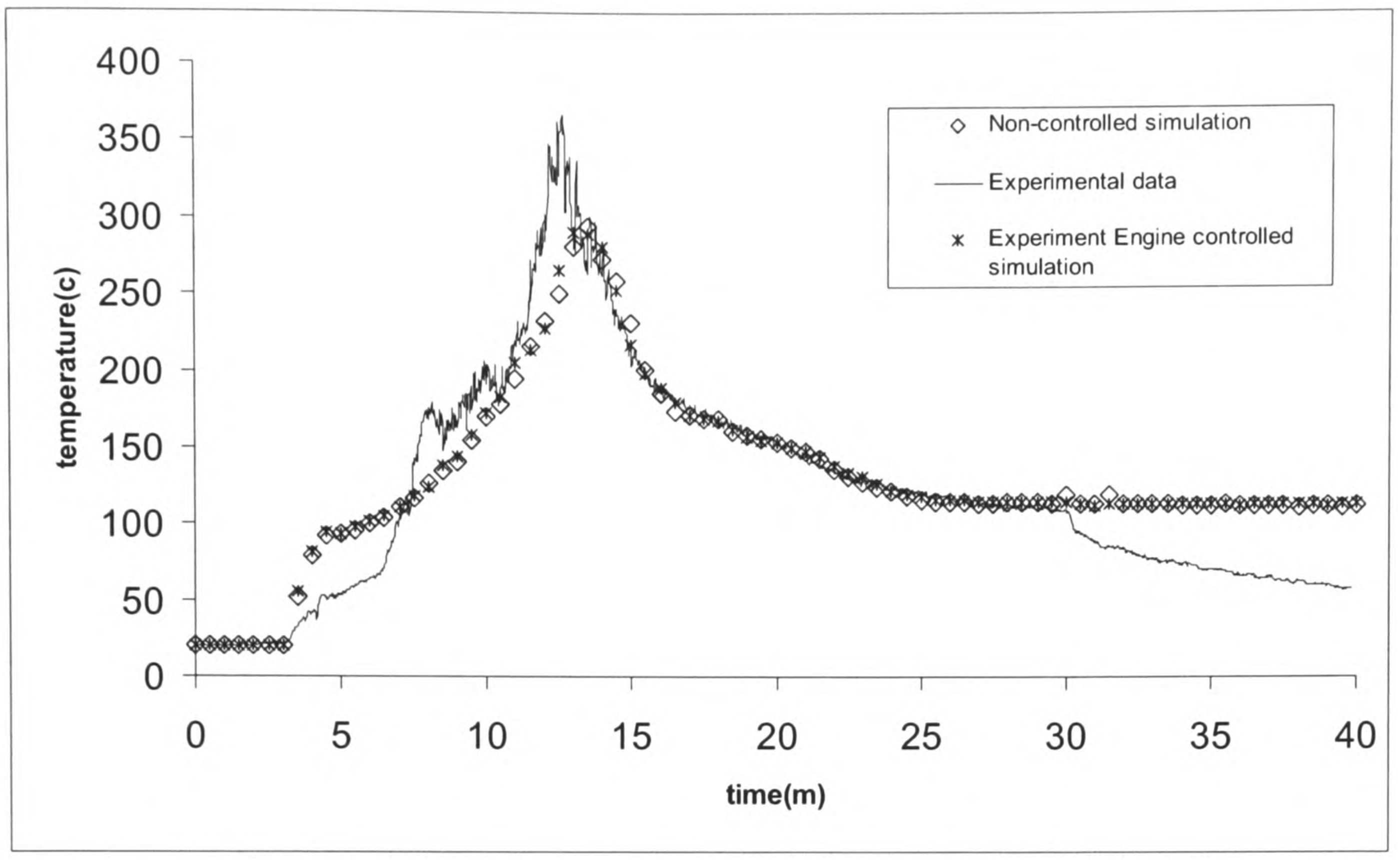


Figure 8-16: Comparison of temperature at position 5 (2.2m) for the corridor cable case

8.4 Three-storey apartment building fire case

8.4.1 Model setup and boundary conditions

8.4.1.1 The Geometry

As mentioned previously, the third test case geometry extends the scenario from the first test case, to have two more additional storeys. The entire building is represented with a 15.6 m x 11 m x 18.6 m rectangular volume. Figure 8-17 shows a 3D wireframe view of the building taken from the Case Specification Environment (CSE) of *SMARTFIRE*.

As shown in Figure 8-18, each floor consists of a main living room, a kitchen, a dining area and several en-suite bedrooms, which are similar to that of the first test case. There is also a stairway connecting the apartment floor to floors above and below and to provide an escape route in the event of a fire. The fire in each floor starts in the living room at subsequently designated time (estimated to represent the spread of fire between floors over the building façade), except for the ground floor where there is also an additional fire source started at 120 seconds of the simulated time in the kitchen area, adjacent to the main living room.

The geometry was again first specified in the Scenario Designer before being loaded as a 3D model in the *SMARTFIRE* Case Specification Environment.

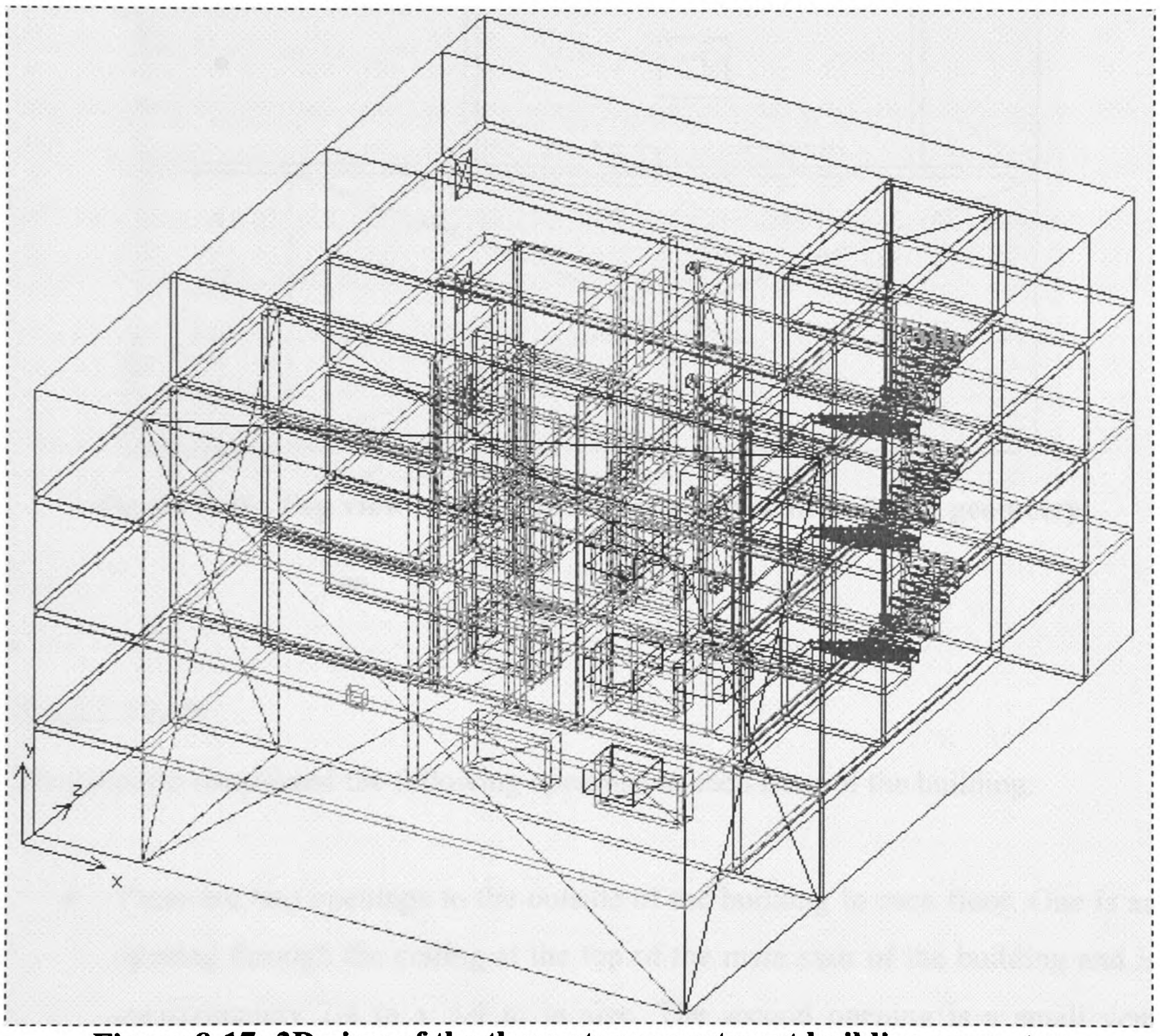


Figure 8-17: 3D view of the three-story apartment building geometry

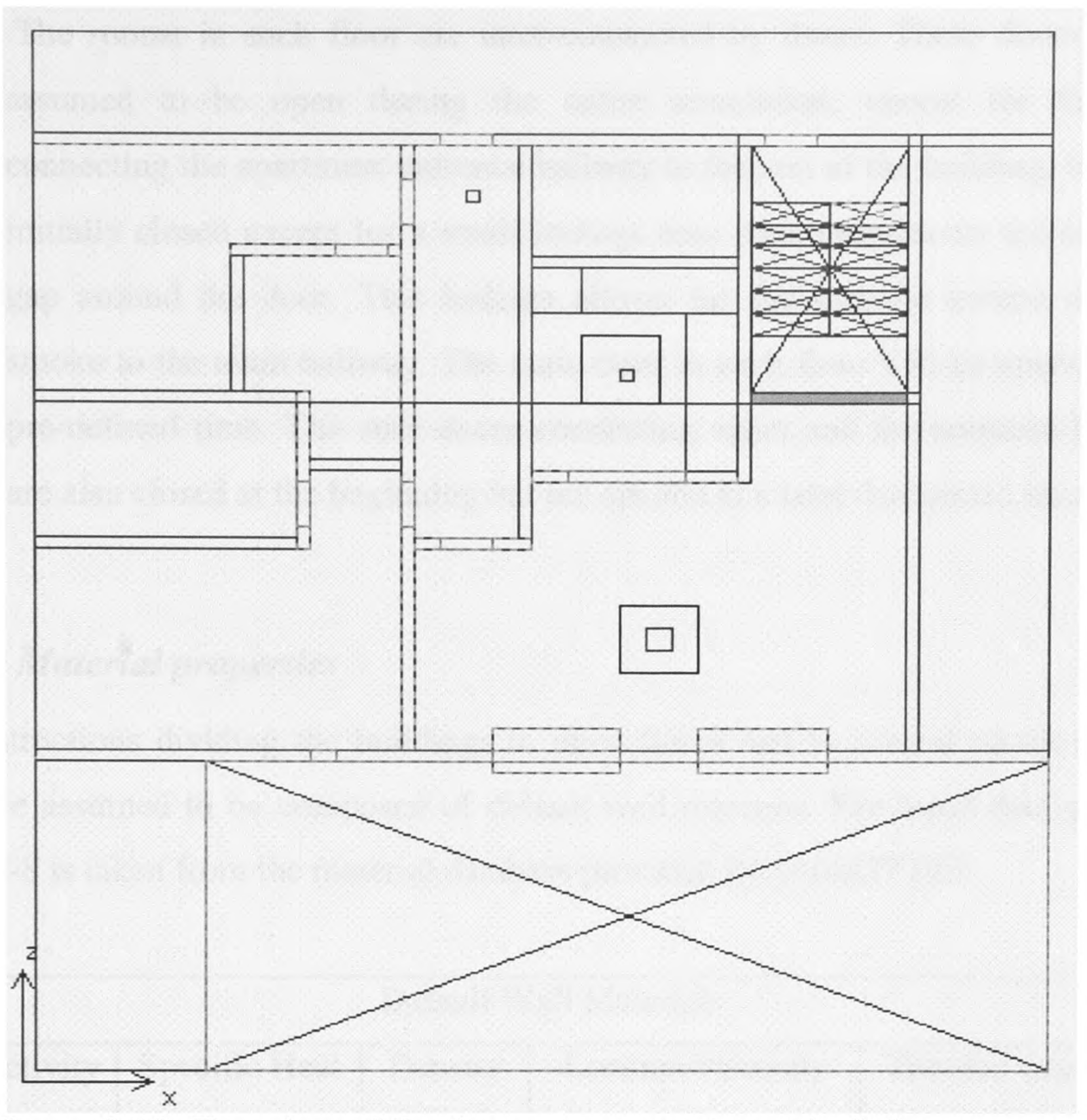


Figure 8-18: Top view of the three-story apartment building geometry

8.4.1.2 Vents

The scenario considered the following openings in each floor of the building:

- There are two openings to the outside of the building in each floor. One is an opening through the ceiling at the top of the main stair of the building and is approximately 2.4 m x 3.4 m in size. The second opening is a small vent located in the bottom right hand corner of the kitchen wall and represents an air brick or grille as would typically be found in a kitchen.

- There are two windows in each living room, where the fire starts, which connect to the outside of the building. These are initially closed and are assumed to be broken and fully removed at specified times.

- The rooms in each floor are inter-connected by doors. These doors are all assumed to be open during the entire simulation, except for the door connecting the apartment entrance hallway to the rest of the building, which is initially closed except for a small leakage area which represents the small air gap around the door. This leakage allows for the realistic escape of some smoke to the main hallway. The main door in each floor will be opened up at pre-defined time. The stair doors connecting stairs and the entrance hallway are also closed at the beginning but are opened at a later designated time.

8.4.1.3 Material properties

All obstructions dividing the buildings to three floors and to several rooms in each floor are assumed to be composed of default wall material. The input data given in Table 8-8 is taken from the material database provided by *SMARTFIRE.*

Default Wall Material				
Conductivity (W/m K)	Specific Heat (J kg /K)	Density (kg/m^3)	Laminar viscosity (Pa s)	Thermal expansion coefficient (K^{-1})
0.69	840	1600	1E+10	1E-10

Table 8-8: obstruction properties for the three-storey apartment building case

8.4.1.4 The fan

An extract fan is located at the end of the main corridor in each floor. This is modelled as an (extracting) inlet in the simulation. The physical properties of the fan are shown in Table 8-9. The fan is initially inactive but is activated at a designated time. The extract fan is intended to provide a smoke management solution which will help to keep the corridor clear of smoke and prevent smoke from spilling into the stairs – which would hinder people using the stairs to exit the building.

The fan						
Temperature (K)	Flow rate (m^3/s)	U-Velocity (m/s)	V-Velocity (m/s)	W-Velocity (m/s)	Kinetic Energy (m^2/s^2)	Dissipation Rate (m^2/s^3)
288.15	-4.8	-8	0	0	0.128	0.203532

Table 8-9: physical properties of the fan at each floor for three-storey apartment building case

8.4.1.5 Initial and Boundary Conditions

The ceiling, floor and surrounding walls of the building are assumed to be non-conducting material (i.e. heat is not conducted through the material). Table 8-10 shows the summary of the initial and boundary conditions.

Wall thickness (m)	Ambient Temperature (k)	Initial Temperature (k)	External pressure (Pa)	Material inside domain
0.2	288.15	288.15	101325	Standard air

Table 8-10: initial and boundary conditions of the three-storey apartment building case

8.4.1.6 Fire specification

Each of the four fire sources in the building was represented as a rectangular object (small red coloured objects shown in Figure 8-17) of 1.2m x 0.6 m and a height of 1.2 m with the same fuel generation rate as shown in Figure 8-19. Each of these fires will last for 420 seconds and generate an equivalent peak heat output of approximately 2MW. The living room fire at the ground floor is active at the start of the simulation. Other fire sources are ignited at subsequent pre-configured times as the simulation progresses.

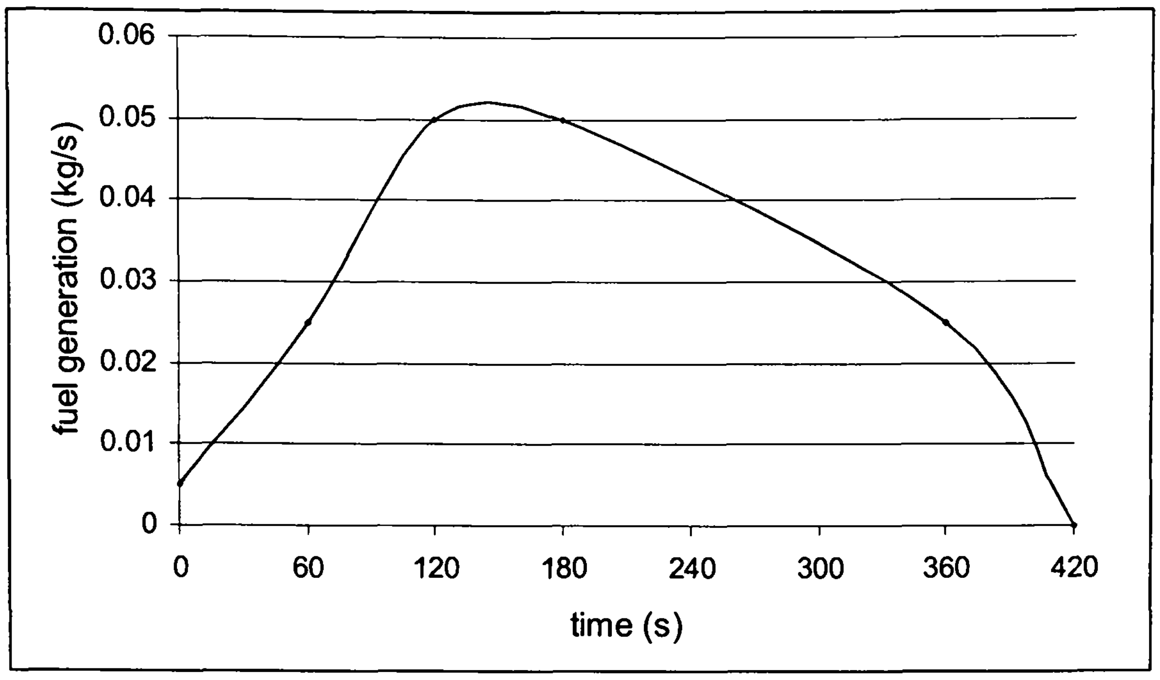


Figure 8-19: the fuel generation rate curve of the three-storey apartment building case

8.4.1.7 The timeline of the geometric changes and transient events

The simulation has been configured with numerous transitional events and with moderately extreme conditions throughout the simulation. The following table shows the designated times for various events that occur during the simulation.

Real time (s)	Geometric changes and transient Events
0	The first fire starts in the living room of the ground floor
60	The first fan, located at the end of the main corridor in the ground floor, is activated.
120	The second fire starts in the kitchen in the ground floor
150	The ground floor stair door is opened
170	The first window, located in the living room of the ground floor apartment, breaks
200	The second window, located in the living room of the ground floor apartment, breaks
230	The third window, located in the living room of the first floor apartment, breaks
260	The fourth window, located in the living room of the first floor apartment, breaks
350	The fifth window, located in the living room of the second floor apartment, breaks
380	The sixth window, located in the living room of the second floor apartment, breaks
400	The third fire starts in the living room of the first floor apartment

500	The second fan, located at the end of the main corridor in the first floor, is activated.
620	The first floor stair door is opened
680	The fourth fire starts in the living room of the second floor apartment
700	The main ground floor apartment door is opened
900	The main first floor apartment door is opened
1000	The third fan, located at the end of the main corridor in the second floor, is activated
1100	The main second floor apartment door is opened
1120	The second floor stair door is opened

Table 8-11: the time line of the pre-defined geometric changes and transient events for the three-storey apartment building case

8.4.1.8 Physical models used

Flow model is enabled. Combustion model is used. The radiation and smoke models are active. The simulations are set to run for 1200 seconds.

8.4.2 The mesh

Some of the meshing considerations were again made even before starting to set-up the geometry in order to ensure that a high quality mesh can be obtained with a reasonable cell budget. The same mesh was used in all the simulations on this case. This mesh was set up by the Expert, in order to ensure that the simulations will not be affected by any possible meshing issues. Figure 8-20, 8-21, 8-22 shows the mesh in x-, y-, and z- directions respectively.

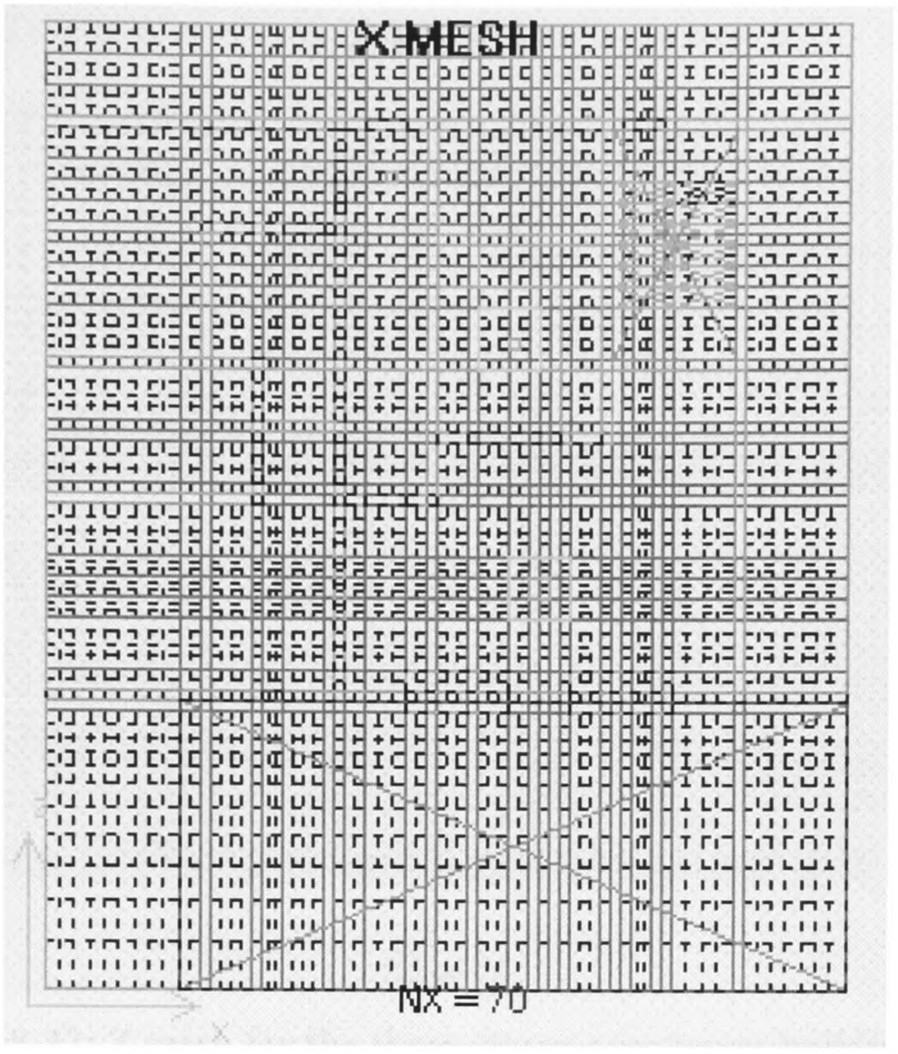


Figure 8-20: X mesh for the three-storey apartment building case

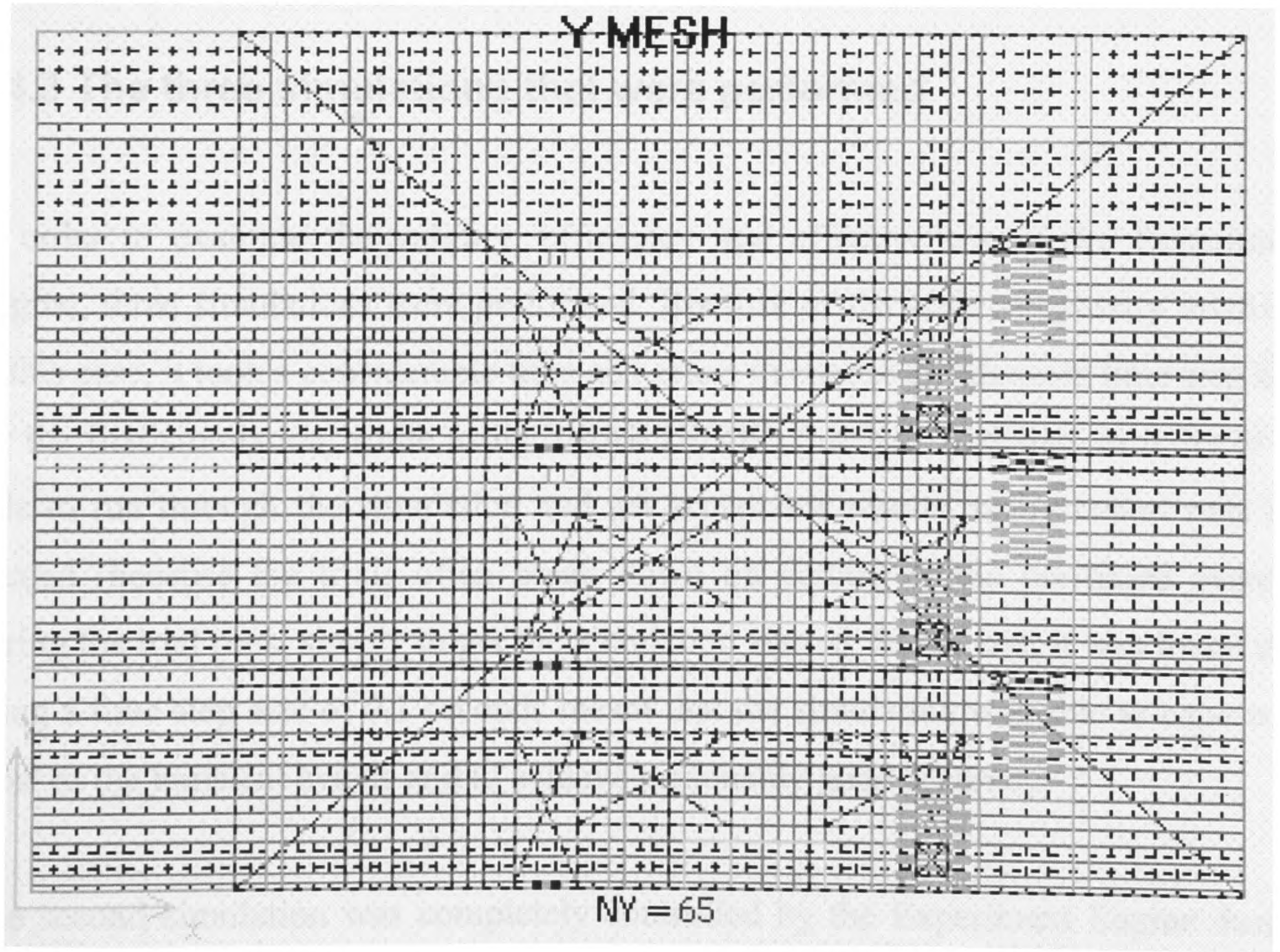


Figure 8-21: Y mesh for the three-storey apartment building case

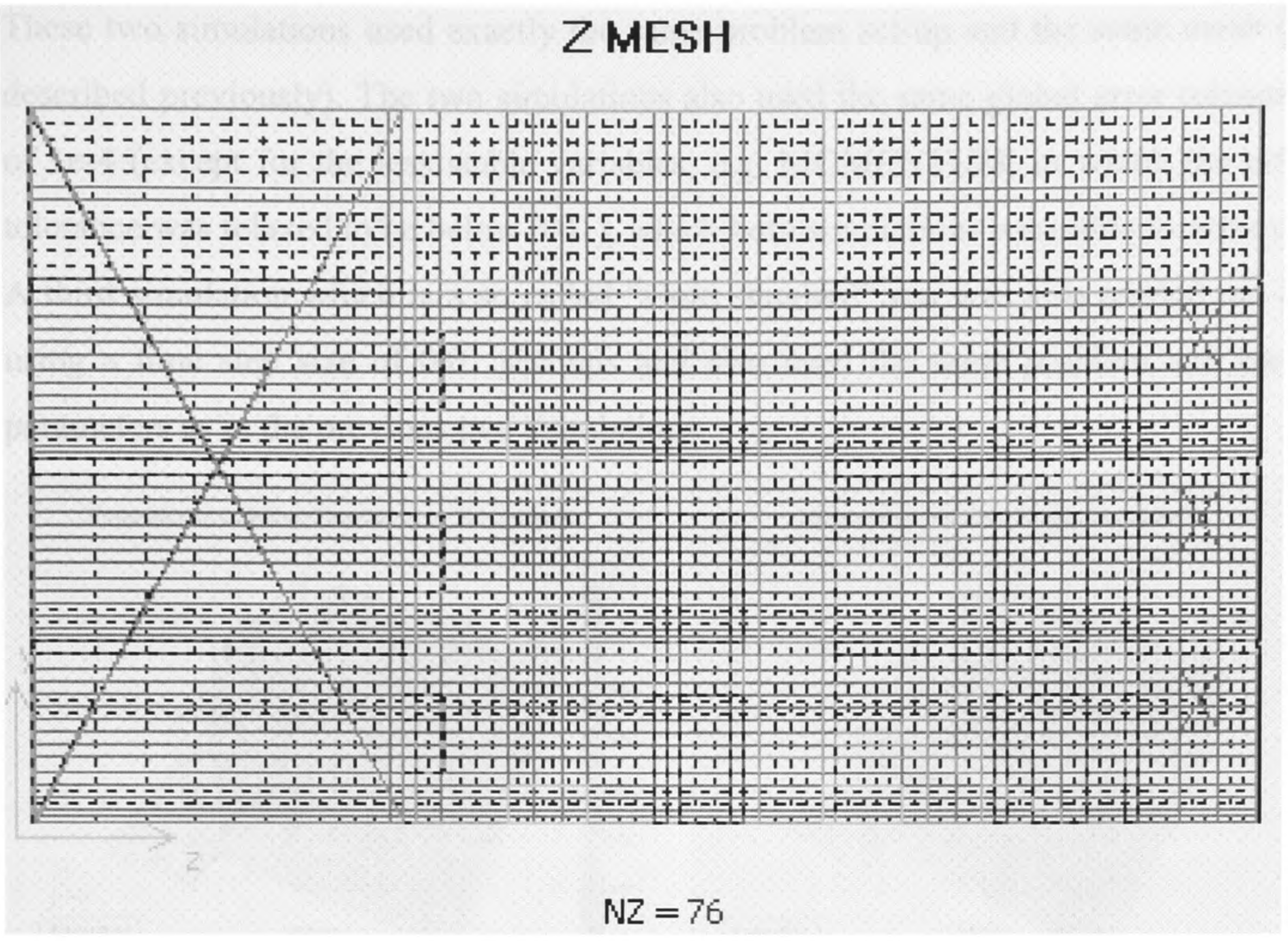


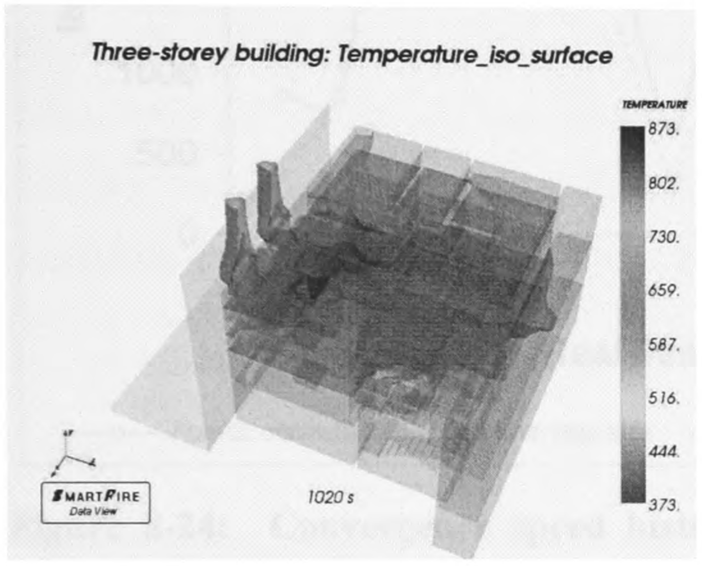
Figure 8-22: Z mesh for the three-storey apartment building case

8.4.3 The three simulations that were performed

In order to examine the validity, efficiency and effectiveness of the Experiment Engine, three simulations were performed. Because of the high complexity involved in this case, it took a considerable time and effort to choose an adequate time step size for the first simulation, while using SMARTFIRE's "default" settings. In order to be able to run through the simulation and get acceptable results, many restart runs are needed (because the trials often break down on certain of the pre-timed events), during the trial runs, simply to get the simulation started. In the end, it was found that using a time step size of 0.1 seconds meant that the simulation was able to cope with most of the transient events and to achieve reasonably good accuracy.

The second simulation was completely controlled by the Experiment Engine during the entire simulation period, without any manual intervention after the essential problem set-up completed by the Expert user in the Case Specification Environment.

These two simulations used exactly the same problem set-up and the same mesh (as described previously). The two simulations also used the same global error tolerance of 1e-4 (except for the less stable variables, e.g. MOMENTUM, in which the error tolerance was relaxed to be below 0.01), which both simulations were able to achieve. A third simulation which is a so called "Gold standard" run was also carried out by using a time step size of 0.01 seconds and also used the same problem and mesh parameters as in the previous two simulations.



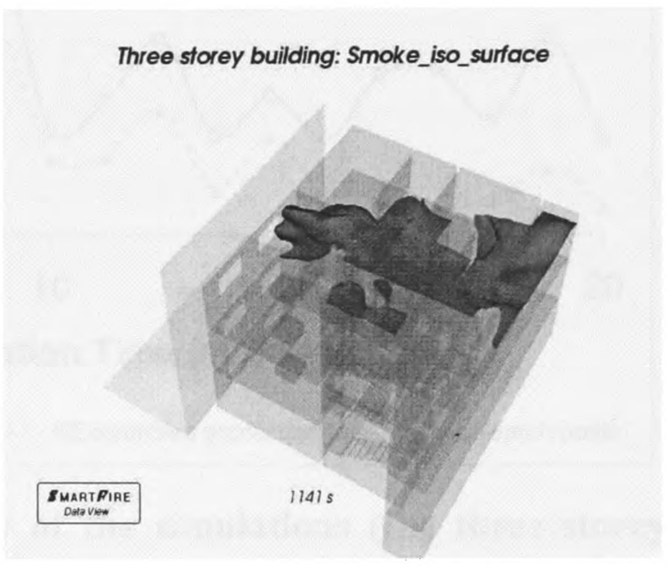


Figure 8-23: Temperature iso surface at 1020s and Smoke iso surface at 1141s for three-storey apartment building fire case based on the Experiment Engine controlled simulation.

Figure 8-23 above shows that the simulation proceeds as expected with the hot gas and smoke from the fire room spreads throughout the building. The broken windows, which represent failure due to the high temperature in the fire rooms, allow hot gas and smoke to escape from the building and to – potentially – cause fire spread up the building façade. The extract fans are able to take some smoke out of the main corridors on each floor and are able to prevent smoke from spilling into the communal stairs.

8.4.3.1 Speed comparison

The speed comparison is again based on the total number of iterations needed by both the EE-controlled and non-controlled simulations. Figure 8-24 presents the convergence history for the two simulations.

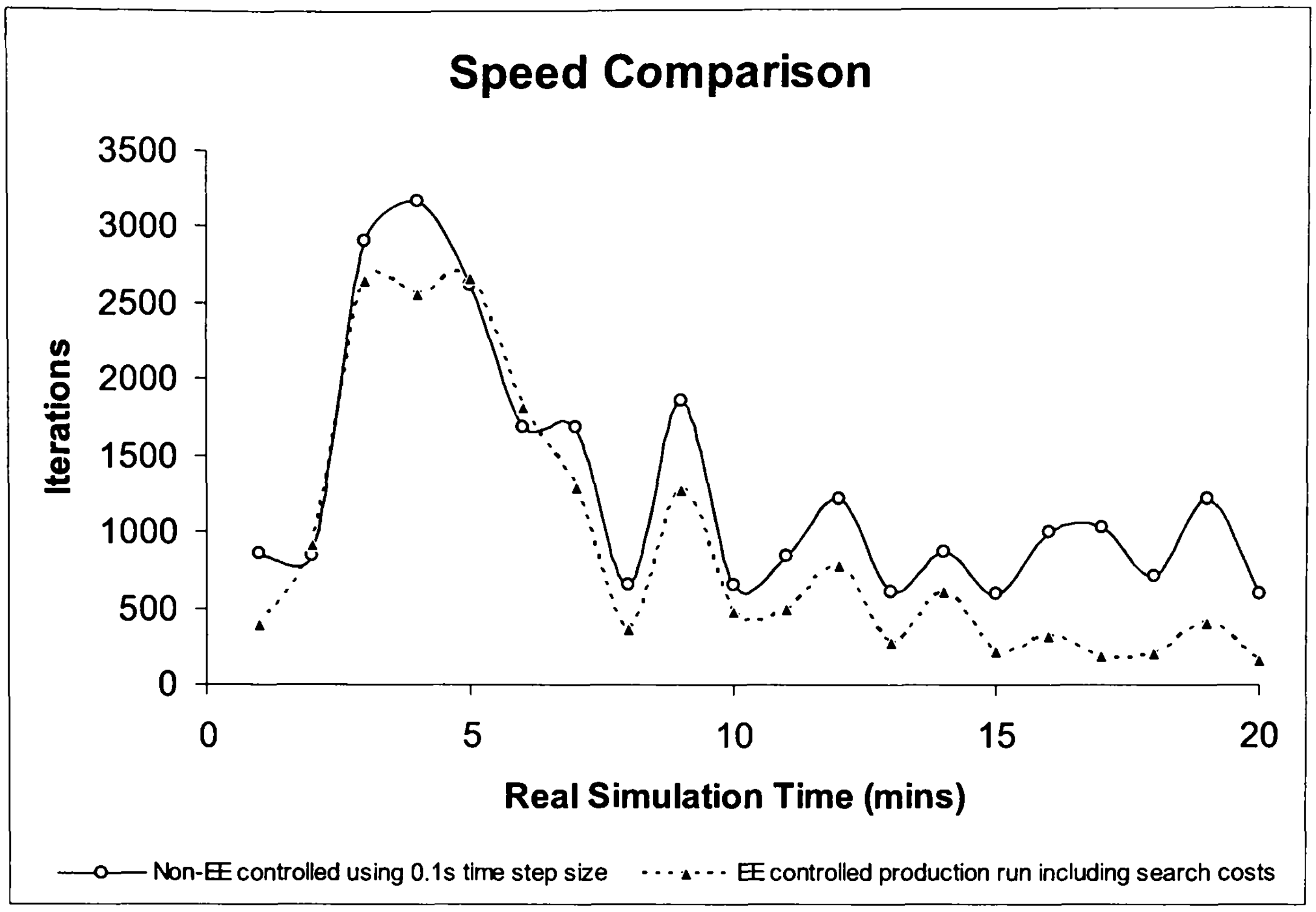


Figure 8-24: Convergence speed history of the simulations (the three storey apartment building case)

As shown in the convergence speed history graph above, both simulations needed many more iterations to converge in the first quarter of the simulation than during the remainder of the simulation. This agrees with the timeline of events, listed in table 8-11 which shows that the majority of the events occurred in the early period of the simulation. However, the path that the EE-controlled simulation takes to convergence is clearly more efficient than that of the non-controlled run. The total number of iterations used to produce the solution for non-controlled simulation was 25628, while the EE controlled simulation only used 18077 iterations (which includes all of the extra searching costs). The EE controlled simulation managed to save some 30% (1-18077/25628) of the total outer-iterations needed to produce the simulation results compared with the non-controlled simulation. Compared to the performance of Experiment Engine (please refer to Figure 8-8 and Figure 8-15) for the first two test cases, this indicates that the nature of control techniques deployed by the Experiment Engine gives higher efficiency as the simulation complexity increases. Hence the validity and effectiveness of the EE system is likely to improve with scenario complexity.

8.4.3.2 Fault recovery

It should be noted that the first simulation, which used 0.1 second time steps, did fail once at 500 seconds of simulated time. This was when the second fan, located at the end of the main corridor in the first floor, was activated. Fortunately, because the event was a pre-defined event, it was possible to save a "bookmark" just before any potential problematic event. With advice from the Expert user, it was possible to manually change the time step size to a smaller one and to restart the simulation from the saved bookmark. The simulation then was able to proceed past the problematic event. After this period, the instability in the simulation had been overcome and the time step size was restored to 0.1 seconds as used before the failure. Ultimately, the simulation produced the results shown in Figure 8-25. Conversely, the Experiment Engine controlled simulation was fully able to complete the simulation without experiencing any problems.

One interesting point about the simulations was that they were conducted over the Christmas holiday period. It was expected that both the EE controlled and non-controlled simulations would be completed before the end of the holiday. However, it was discovered that the non-controlled simulation had actually crashed after only 500 seconds of simulated time. This meant that several more days of processing, beyond the allotted period, were actually required to complete the simulation. This shows that, as the size and complexity of simulations increase, so the cost of simulation failures can be very expensive and can have huge impacts on project deadlines.

8.4.3.3 Accuracy assessment

As previously mentioned, a "gold standard" run was also conducted for the three-storey apartment building case to allow assessment of the accuracy of both the EE-controlled and non-controlled simulations. A region with high temperatures and large flows (near the centre of the living rooms door) in the fire room was selected to demonstrate that the changes made by the Expert user – for the non-controlled simulation – and the techniques deployed by the Experiment Engine, have not significantly changed the simulation results. Figure 8-25 shows the vertical temperature profile in the near fire region in the living rooms. Although these results are not identical, they clearly show that the results produced by both the EE controlled and the non-controlled simulations are in good agreement with those of the "gold standard" run. This indicates that the control techniques deployed by the Experiment Engine have had negligible impact on the quality of the final results.

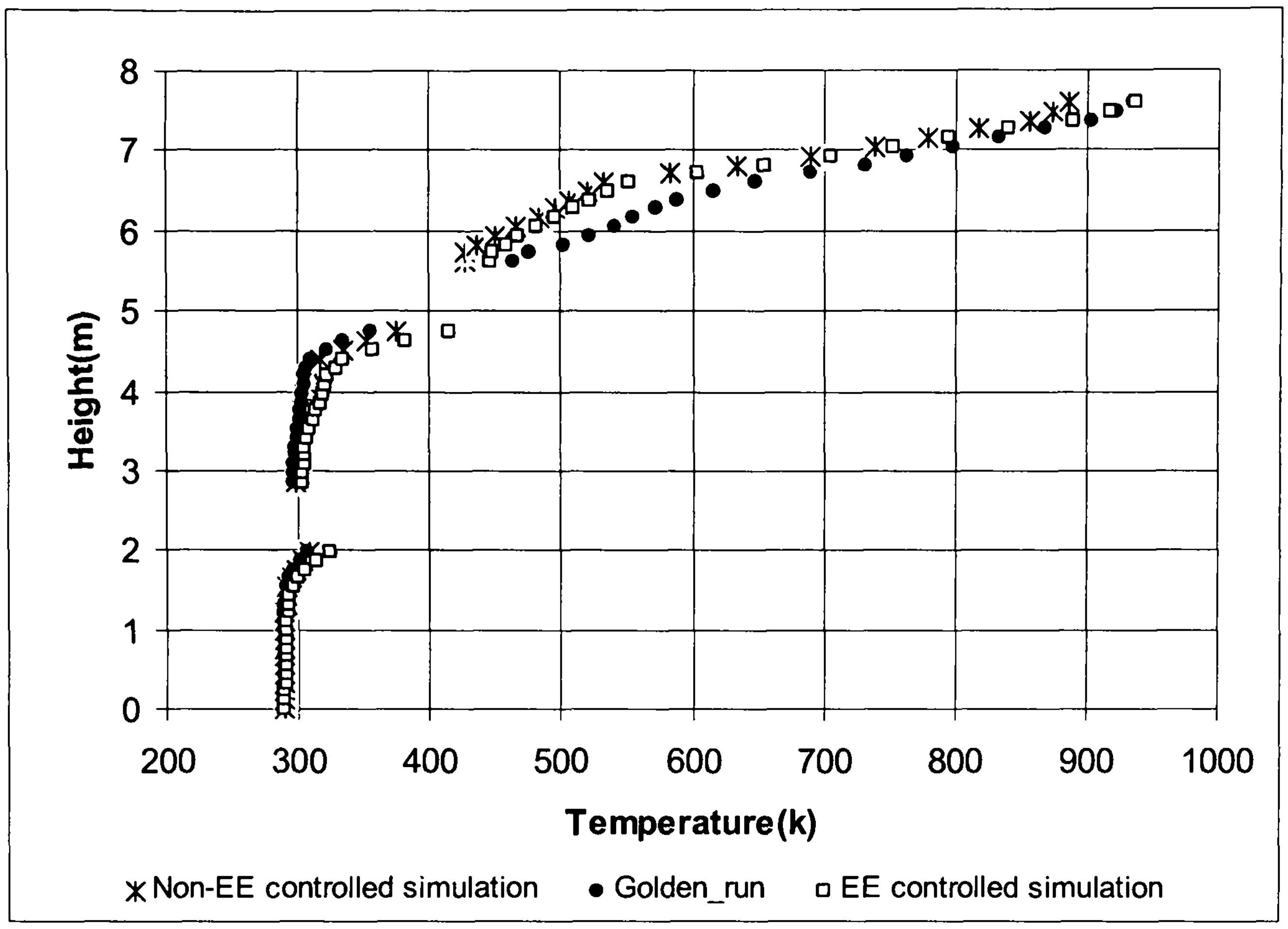


Figure 8-25: vertical temperature profile at centre of doorway in the fire room at 960s for the three-storey apartment building case

[Chapter Summary]

This chapter examines the behaviour of the prototype Experiment Engine for controlling a number of simulation scenarios that represent some typical fire modelling challenges that a FFM user would encounter.

In order to determine the effectiveness of the EE at controlling these simulations, the simulations have been compared with "default" non-controlled and Expert user controlled simulations. In some cases it has not always been possible to compare with the "default" non-controlled and Expert user controlled simulations because the "default" and Expert configurations were not able to complete the entire simulation. In order to obtain useful comparison information, the Expert user was given further opportunities to modify the configuration – in order to obtain a solution. This did, however, demonstrate that the EE controlled simulation provided a high degree of robustness for handling unstable simulation scenarios.

Comparison of the results has demonstrated that the Experiment Engine controlled simulation can produce physically "correct" results with almost no user intervention. Analysis of the changes that have been applied by the EE has demonstrated that the EE provided a reasonable emulation of an Expert user's control decisions in terms of changing time step size in response to the physical changes in the domain. Furthermore the EE demonstrated that it was very efficient at recovering from simulation failures. Generally, the EE was able to ensure that the simulation was always stable and gave accurate results. Analysis of the processing time and iterative steps indicated that the prototype Experiment Engine was also capable of providing significant performance improvements over the "default" simulation, especially for the unstable case, where the EE was able to reduce the total number of iterations used, by over 40%.

It was noted that the selected test cases demonstrated that the EE system performed more effectively as the problem size and complexity increased. It is argued that, because of the nature of control techniques deployed by the EE, further increase in simulation complexity are likely to lead to further improvements to the effectiveness of the EE system – when compared to either a default simulation or an Expert user controlled simulation.

9 Conclusions and Further work

[Chapter Overview]

This thesis describes a systematic study to investigate techniques to assist with the reliable specification and successful simulation of Fire Field Modelling Scenarios. As a result, the research has identified and investigated a methodology for making use of a CFD Expert's knowledge to support various stages of the simulation process including the key stages of creating a mesh and performing the simulation. The research has also indicated an approach to the control of a FFM CFD simulation which is analogous to the way that a FFM CFD Expert would approach the modelling of a previously unseen scenario. The investigations have led to the identification of a set of requirements and appropriate knowledge which have been instantiated as the, so called, Experiment Engine. This prototype component (which has been built and tested within the *SMARTFIRE* FFM environment) is capable, both of emulating an Expert users' ability to produce a high quality and appropriate mesh for arbitrary scenarios, and is also able to automatically adjust a key control factor of the solution process.

The two major topics considered in this thesis are:

- The method for the automatic specification of an appropriate and suitably high quality of structured mesh for an arbitrary simulation scenario, and
- The method for automated control of the simulation solution process

This chapter summarises the research that has been undertaken, the knowledge and approaches that have been identified and their applicability. The chapter then concludes with answers to the research questions that were posed in the introduction. This chapter finally goes on to identify the shortcomings and limitations of the current research, which will need further investigation.

9.1 Contribution and summary of work

The contribution to knowledge and work done in this research are discussed here as answers to the research questions posed in the introduction.

9.1.1 Automated reliable CFD set up

The main research question of the research was:

- To what extent is it possible to emulate an Expert users' ability to analyse a series of trials starting from a simplified, coarse mesh simulation run (or even to learn from a complete simulation failure) to help make a better set-up and meshing solution.

This question has been investigated by looking at the key factors for successful simulation of a FFM scenario. After identification of the stages and approaches that a CFD FFM Expert would use to ensure a successful outcome from a simulation, it was possible to propose an automated knowledge based approach to provide the same responses – without the need for interaction by the Expert. This approach has been tested using the development of a new component within the *SMARTFIRE* FFM environment, called the Experiment Engine (EE). The EE has subsequently been tested on significant FFM scenarios and it demonstrates that it is possible to emulate an Expert user and to successfully control and support the set-up of a FFM simulation. This thesis explains in detail how this task was accomplished (with knowledge elicitation, investigation of appropriate techniques and rapid prototyping) and then demonstrates the effectiveness of the EE system on a number of simulations. The following discussion provides an overview of the mode of operation of the EE system.

First, a default test case is formed as a result of case recognition and from applying simplification strategies (i.e. Linearising/simplifying any complex curve for the Heat Release Rate, simulation period selections etc.). The default test case is then passed to the CFD Engine (solver) for simulation. At the same time, the Case Specification Environment (CSE) and the CFD Engine establish communication channels using hand-shaking messages initiated by (CSE) Experiment Engine component. The experiment control parameters (i.e. the predefined solution error tolerance, convergence tolerances etc.) are wrapped as messages according to a defined message protocol and then passed to the CFD Engine through the communication connection. From this point, the EE has taken full control of running the set-up routines in the CSE and the CFD Engine.

While the CFD Engine is running, the solution status and results are constantly monitored by a monitoring process that checks the convergence status at the end of each iterative sweep. In the event of detecting a critical condition (e.g. convergence, having reaching the configured number of iterations or encountering some fatal error), the current solution status and the results summary are extracted and wrapped as messages that are sent back to the Test Controller (TC), an EE component in the CSE that controls the tests. This TC then decides if any action needs be taken because of the message or if it is acceptable to just let the CFD Engine continue running.

The TC compares the solution status with the experiment control rules. If the reported error is bigger than the predefined error tolerance, then the TC will activate the local mesh refinement sub-process. The result of this action will lead to a refinement of the current mesh according to the error distribution that was found on the previous unrefined mesh.

In the event of mesh refinement taking place, the TC will construct a new test which consists of the newly refined mesh and any required changes to the solution control parameters (N.B. only the time step size is controlled at the time of writing although there is no reason that more complex control strategies could not be implemented).

This revision of the solution control depends on the quality assessment of the current solver status information and other critical variable values (e.g. pressure, velocity values, residuals, etc.)

If the TC decides that it is not necessary to refine the mesh, then it is possible to retrieve the previously stored restart files to make other necessary modifications (i.e. new time step size). So the results of the previous time step are used as a starting point for the re-run. In this way, it is not necessary to re-run the test from the very beginning, and in this way, the EE keeps the cost, of the experiments, to a minimum.

When the CFD Engine starts to solve a new time step, a bookmark is initially saved. This will be used if the experiment needs to be restarted from this point – due to subsequent poor solution performance or solution failure.

At the end of this iterative trial, the TC will check to determine that no more changes are deemed to be necessary, and the trial will be assumed to be successful at the prescribed end-simulation time. The production-run set-up is formed according to the original problem specification and the set-up, meshing and control parameters of the last successful test case from the EE process.

It has been demonstrated that the EE process gives an improved successful rate of simulating typical and moderately complex fire scenarios. The EE also gives better solution reliability, prevention of (and recovery from) "bad" set-up and meshing, greater robustness and greater ease-of use for the software.

9.1.2 Effective and efficient simulation solution control

The second research question, answered by this thesis, is:

- To what extent would it be possible to emulate an Expert's ability to control the production run assuming that appropriate set-up and meshing has been obtained in the experiment stage.

It was quickly determined that even a near perfect set-up for a FFM scenario cannot hope to guarantee a successful simulation because there are so many factors which can effect the simulation outcome during the production processing. Furthermore, the simplifications that have been used to perform experimental tests are not necessarily fully representative of all stages of the subsequent simulation. This research then attempted to determine what sort of problems could occur during a simulation, how these could be detected, what coping strategies a CFD Expert could apply and how these strategies could be automated. Investigations into these questions used rapid prototyping within the Experiment Engine to attempt to emulate the Expert user's control and to test the effectiveness of the prototype control system. A number of significant simulation studies have demonstrated that the enhanced EE is both fully robust and provides a fully automatic control technique to take control of the whole of the fire simulation process.

The Experiment Engine consists of a sophisticated control system that constantly monitors the solution state and takes any necessary actions to make sure that the solution converges to the preferred error tolerance (whenever this is possible) and also provides recovery from any detected simulation instability or failure. The EE is also looking for any possible simulation speed improvement so as to produce a good quality solution in as short a time as possible.

This research has been helped by the fact that the EE has already checked the appropriateness of the simulation set-up and production-run meshing solution, so the control system (embedded in the EE) is able to utilize a simpler, more direct and better evaluation function (with a sophisticated convergence forecast function) to determine the simulation state. This makes the control decisions more effective and keeps the cost of the test searches (for simulation speed-up) to a minimum, thus making the EE more efficient. As a result, control parameters changes made by the control system can respond effectively and quickly to the physical condition changes (i.e. heat release rate change, transient events etc.) in the domain, in the same way that an Expert use would. It has been demonstrated that the EE controlled simulation can save up to 40% of simulation sweeps for complex fire scenarios, when compared with non-controlled simulation. The control technique deployed by the Experiment Engine has negligible negative impact on the final physical results, thus the Experiment Engine controlled simulations are able to produce physically sound results, which are generally identical to the non-controlled simulations when using the same set-up, meshing solution and global error tolerance.

It should also be noted that, as the simulation scenario becomes more complex with more numerous transitional changes and with more extremity of physical conditions, so the EE based control technique is likely to give even better solution control performance in what are likely to be very lengthy simulations.

9.1.3 Case classification, case recognition and block-wise mesh justification

The first subsidiary research question answered by this thesis is:

- Can an arbitrary "real" fire scenario be characterised into a distinct category and is it possible to build a corresponding library of meshing parameters for each category?

In order to provide the best possible meshing solution for a FFM simulation, it became apparent that FFM CFD Experts would apply scenario specific knowledge to the meshing. This led to an investigation of the different classes of geometry that would require specific forms of meshing. It was decided to focus on a structured meshing system because there are still a very large number of structured mesh FFM systems in use. Expert FFM users were questioned about strategies that would be used to classify and mesh various types of geometry. This investigation led to the specification of additional meshing libraries for different types of identified geometries. This was identified as a critical need because the original meshing library, in the target system, was only based on meshing expertise for a simple room fire scenario. The first task that was investigated was how to classify the real world of fire simulations into typical categories according to the geometry type and fire source characteristics. After this, a case recognition algorithm was developed to automate the selection of the most appropriate set of meshing parameters from the corresponding library. One problem that was identified, with the previously generated mesh from the structured meshing system, was that of the problems caused by poor cell aspect ratio. It was demonstrated that a newly developed block-wise mesh justification approach was able to recover from cell aspect ratio problems more effectively and efficiently.

9.1.4 Block-wise adaptive mesh refinement

The investigation of Block-wise adaptive mesh refinement methods, in this research, was in response to the following subsidiary research question:

- What are the indicators to signal that a mesh needs to be refined, and to what degree, and how should it be refined?

The method of determining where refinement is needed is to use the local continuity error e_i computed by the solver as an enrichment indicator. It is assumed that large errors come from regions where the local error estimate e_i is large. These errors will be even more significant when the cell volume is large. Consequently, regions with large errors are where the mesh should be refined. Using the combination of the interrelated global error tolerance and the local error tolerance, gives a global mesh quality that is assured and, at the same time, only regions that are likely to cause problems are actually refined. This makes the refinement process more efficient and effective.

Currently, the refinement method has been investigated for a structured meshing algorithm – which is typical of many of the structured meshing systems which are still widely used in FFM. Research has indicated that the new refinement method should use blocks as units rather than refining individual cells. This "block-wise" refinement takes advantage of the block structure that is a key feature of the target meshing system. The refinement strategy, that has been adopted, makes all of the cells in the target block, smaller in size by adding only the minimum number of cells required to improve the cell quality. This has shown that the target block is refined with a high degree of confidence that this refinement will minimize the global errors.

In summary, the adaptive mesh refinement procedure consists of the following stages: (i) construction of an initial base mesh, (ii) local refinement of the base mesh in regions where the existing mesh resolution is inadequate, and (iii) correction and justification of the refined mesh when it becomes overly distorted (i.e. having aspect ratio or cell neighbour length problems). This methodology makes use of earlier works on the case classification, case recognition and block-wise mesh justification, and exploits the block structure of the structured meshing system.

9.1.5 Changing time step size according to the physical conditions in the domain

The following subsidiary research questions concerned the quality of the simulation solution:

- In what circumstances would Expert users tend to change times steps size, to what degree should the time step size be changed, and what possible impact would the Expert expect these changes to have on the solution?
- How can the convergence of every time step size and simulation stability be assured throughout the whole simulation process?
- Is it possible to prevent and recover from all types of solution fault?
- Can the system provide improvements in terms of simulation speed when compared to non-controlled simulations?

These subsidiary research questions were investigated using knowledge acquisition (from Expert users) during the rapid prototyping of a new solution control system embedded into the new Experiment Engine procedures.

After consulting with Expert users, it was found that there is at least one parameter that is well understood. This is the time step size. There is a reasonable agreement, among the Expert users, about the effects of time step size changes on the simulation stability. Generally, there is a limit on the magnitude of sudden solution changes that a numerical code can handle. Excessive and/or sudden changes of simulation conditions (i.e. large heat release rate changes or major geometry changes caused by transient events) are the biggest stability problems facing the CFD simulation. On one hand, the time step size is the measure of how much the simulation can progress in a single time step of computation. On the other hand, it is also a means of controlling/limiting the magnitude of the changes that are applied during each time step in the simulation. This gives a clue how to ensure the stability of the CFD simulation. When the magnitude of sudden changes approach the limitations of what a numerical code can handle, then a smaller time step size should be applied to allow the excessive changes to be applied in smaller portions, where each proportional change is within the limitations that the code can deal with. In this was the simulation will be able to progress past critical/sizeable changes that would otherwise cause instability or break the simulation. The same logic applies to a larger time step size. A larger time step size may make the simulation advance faster since, for each time step, a larger amount of time has passed. However this may also make the simulation less stable depending on how large the changes are during the new time steps. It is possible to summarize the Expert users' knowledge as follows:

- Increasing the time step size can lead to divergence but may also speed up the simulation.
- Decreasing the time step size tends to stabilise the simulation but not necessarily slowing down the simulation.

Moreover, Expert users generally believe that there is an optimal time step size range for each particular stage of the simulation, depending on the physical conditions in the domain at that time. The time step size should adjust to the varying physical conditions in the domain. The time step size changes, made by the Experiment Engine, are based on an assessment of the simulation stability and solution convergence behaviour according the residual graphs (which are the indicators of the physical changes in the domain). The Experiment Engine is then able to respond to the transient events in a similar way as an Expert user would. A significant computation example has demonstrated a 40% saving on the number of iterations needed when using the Experiment Engine compared with a non-controlled simulation that had the same set-up, mesh and global error tolerance.

9.1.6 Component-based software architecture

The remaining research questions that are answered by this thesis are:

- Is it possible to instantiate the Knowledge acquired through this research into an automated software system (a Knowledge Based System integrated into the software architecture of a CFD code) such that the captured knowledge offers complete and automated user support for using the software through all stages from set-up, through meshing and also the run time control. The ultimate purpose of this system is to enhance robustness, stability and convergence of the CFD simulation, to provide performance gains and consequently enhance the rate of making successful fire modelling simulations?
- Given the complexities of the FFM software, how should the Experiment Engine concept fit into the existing FFM architecture?
- Is the design flexible and extensible, in order to accommodate future developments and research directions?
- Is there a role for the Experiment Engine to provide a framework for the operation of other CFD supporting techniques (e.g. ICS and Group Solvers) that also aim to facilitate and/or optimize fire modelling simulations?

The approach used to instantiate the Experiment Engine concept, within the target FFM environment, was to leave the existing Case Specification Environment (CSE) and the CFD solver largely unaltered. The Experiment Engine was demonstrated as a new component which resides in the Case Specification Environment and acts as a communication channel between the CSE and the CFD Engine. The new EE framework then provides links between the original Case Specification Environment (CSE) and the CFD Engine components. These components can now communicate via data passed through the well defined message interface. In this way, it is possible to maintain the original Case Specification process and CFD Solver process completely independently. The two components interact with each other only when the EE is operating. This form of componentization has enabled rapid prototyping and implementation/research of new research ideas and requirements as and when they have been conceived. In the Experiment Engine enabled mode of the simulation, the

EE takes full control of the running the software. The EE decides when to deploy the ICS and/or Group Solver techniques in order to further optimise the solution process. In this way, the EE is not a substitute for ICS and/or Group Solvers, but rather a form of management tool to make sure that the ICS and Group Solvers techniques work under optimal conditions – with support of full monitoring and fault tolerance.

9.2 Further Work

The tests and validation of the Experiment Engine demonstrate that it is quite robust and effective. However, there are still a number of ways to improve the overall efficiency, reliability, and robustness of the EE and its mode of operation.

9.2.1 The need for solution data re-mapping

One major disadvantage of the mesh refinement procedure is that many (if not all) of the grid points will be redistributed after the refinement. This makes the re-use of the solution values, from the pre-refined mesh onto the refined mesh, extremely difficult. Currently, if the mesh is changed, then the Experiment Engine has to form a new test case with a newly refined mesh that is passed to the CFD Engine for computation from the very beginning of the simulation. This is certainly not ideal and is not very efficient. In this current research it was decided to limit the considerations to a typical structured mesh architecture since these are still widely used for FFM. The target system used for testing these concepts had a number of limitations and software maintenance issues, that made it difficult to alter the existing solution procedure to allow for the re-use of older results. It is now proposed that the development of a re-mapping procedure, to re-use results from the previous computation, would make the Experiment Engine process much quicker and more cost efficient since it would be able to use refinement more optimally as a preferred tool to handle many of the production run solution problems.

9.2.2 The need for Intelligent control scheduling

Some efforts were made to restrict the computational expense of the search costs of the Experiment Engine controlled production run. The control scheduling does not fully analyse the physical condition at every time step, but rather it is performed with a predefined frequency (i.e. it is run at every 10 time steps), which has resulted in a marked underachievement for the first computation example presented in the previous chapter (please refer to the chapter 8 for details). In the last quarter of the simulation, the time step size is repetitively changed up and down and these repetitive oscillatory changes do not provide real benefits for the simulation speed. In fact, the overall performance is slightly slower than the corresponding non-controlled run during the same simulation period. It is believed that substantial benefits can be obtained by introducing a more sophisticated scheduling of the search for speed improvement. An intelligent control scheduling should use a variable and flexible interval of search to make the search more effective and thus more cost efficient. The intent would be to eliminate potentially unsuccessful searches and rather to apply the search as soon as it is deemed necessary. The possible solution to this issue would be a procedure to recognise the physical state of the simulation explicitly (i.e. recognise the stage that a simulation has reached, e.g. flash over, decay phase etc.). Also the Experiment Engine should keep a search history that remembers all of the search failures and decisions and would not repeat the same search mistakes.

9.2.3 Extension to control multiple control parameters

Currently, only the meshing is adapted and the control parameter changes are limited to adjustment of the time step size, in the Experiment Engine controlled simulation. It is possible that other aspects of the scenario could also be controlled. For example, the changes to the time step size during the simulation have proved to be very effective at solving the simulation stability problem. This control also provides speed benefits to complex scenarios. However this control strategy may not be the most suitable one for already stable simulations. Although the changes to the control parameters should be limited to one at a time, a major study on how the various control parameters affect the convergence behaviour of the simulation could lead to the development of problem-specific control actions, in which the control parameter changes are specific to only a particular type of scenario (i.e. in a stable simulation, changes of the linear relaxations may provide a better speed improvement when compared to changes of the time step size). In addition, an understanding of the detailed physical condition should play a key part in the assessment since the author feels that analysis of the residual graph alone, is not adequate for this task.

[Chapter Summary]

This chapter has summarized how the research, investigations, knowledge elicitation and testing have answered the research question posed at the start of this research. In the process of investigating this research question, a number of subsidiary research questions became apparent and these were also investigated and answered. This thesis represents a successful outcome to the research question, since the conclusions are that it is possible to learn-from and emulate an Expert users' ability to analyse a series of trials starting from a simplified, coarse mesh simulation run; and to be able to make an adequate set-up or better mesh solution even from a complete solution failure.

Further investigations have demonstrated that it is also possible to emulate an Expert users' ability to control a simulation solution and to provide significant benefits in terms of performance, overall reliability and accuracy of results, by instantiating all of the monitoring and control techniques, along with Expert user's knowledge and improvement strategies into the new control module called the "Experiment Engine". The prototype EE has been implemented with communication channels that enable run-time communication between the Case Specification (set-up) process and the CFD solution process and, through this mechanism, the EE controller is able to take full and automatic control of the whole simulation process and is consequently able to produce robust simulation performance and assured results to a prescribed quality with almost no user interventions. The EE was also typically able to perform these automated simulations in a shorter simulation time than a default configuration. Furthermore the EE demonstrated that this approach to simulation management could surpass individual Expert users when operating on previously unseen scenarios that contained many transitional events.

This chapter finally gives a number of pointers to further work that the author feels would be beneficial to this area of research.

References

[Ainsworth-00]

Ainsworth M., Oden J.T., "*A Posteriori Error Estimation in Finite Element Analysis*", Wiley-Interscience, John Wiley & Sons, September, 2000.

[Arney-87]

Arney, D.C., Biswas, R., and Flaherty J.E., "*A posteriori error estimation of adaptive finite difference schemes for hyperbolic systems*", In Transactions of the 5th Army Conference on Applied Mathematics and Computing, pp 437-458, 1987.

[Arney-90]

Arney D., Flaherty J., "*An adaptive mesh-moving and local refinement method for time-dependent partial differential equations*", ACM Transactions on Mathematical Software 16(1): pp 48-71, 1990.

[Babrauskas-75]

Babrauskas V., COMPF: "*A Program for Calculating Post-flashover Fire Temperatures (UCB FRG 75-2)*", Fire Research Group, University of California, Berkeley, 1975.

[Babuska-83]

Babuska I.,Chandra J. and Flaherty J.E., editors, "*Adaptive Computational Methods for Partial Differential Equations*", Philadelphia, SIAM. 1983.

[Babuska-01]

Babuska I., Strouboulis T., "*The Reliability of the FE Method*", Oxford Press extra, 2001.

[Baker-89]

Baker T. J., "*Automatic Mesh Generation for Complex Three-Dimensional Regions Using a Constrained Delaunay Triangulation*", Engineering with Computers, Vol. 5, pp.161-175, 1989.

[Bangerth-03]
Bangerth W., Rannacher R., "*Adaptive Finite Element Methods for Differential Equations*", Lectures in Mathematics VIII Vol. 207. Birkhauser, 2003.

[Bathe-96]
Bathe K. J., "*Finite Element Procedures*", Prentice-Hall Inc, New Jersey, 1996.

[Benson-91]
Benson R. and Mcrae, D.S., "*A Solution Adaptive Mesh Algorithm for Dynamic/Static Refinement of Two and Three Dimensional Grids*", Third International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Barcelona, Spain, June 1991.

[Berger-84]
Berger M. and Oliger J., "*Adaptive mesh refinement for hyperbolic partial differentialequations*", J. Comput. Phys. 53, pp 484-512, 1984.

[Brackbill-82]
Brackbill J.U. and Saltzman J.S., "*Adaptive Zoning for Singular Problems in Two-Dimensions*", Journal of Computational Physics 46, pp342-368, 1982.

[CFAST-93]
CFAST, "*The Consolidated Model of Fire Growth and Smoke Transport*", NIST Technical Note 1299, 1993.

[CFAST-00]
A Technical Reference for CFAST, "*An Engineering Tool for Estimating Fire Growth and Smoke Transport*", NIST Technical Note 1431, 2000.

[CFAST-04]
"*CFAST Computer Code Application Guidance for Documented Safety Analysis Final Report*", U.S. Department of Energy Office of Environment, Safety and Health July, 2004.

[CFX-97]

"*CFX-4.1/4.2 flow solver use guide*", AEA Technology, October, 1997.

[CFX-sol]

"*Computational Fluid Dynamics Solution*", ANSYS, http://www.ansys.com/assets/brochures/cfd-solution-10.pdf.

[CFX-web]

http://www.ansys.com/cfx

[Chyu-95]

Chyu W. J., Rimlinger M. J., and Shih, T., "*A procedure for automating CFD simulations of an inlet-bleed problem*", In NASA. Lewis Research Center, Surface Modeling, Grid Generation, and Related Issues in Computational Fluid Dynamic (CFD) Solutions pp 731-749, 1995.

[Cook-82]

Cook W.A., Oaks W.R., "*Mapping methods for generating three-dimensional meshing*", Computers in Mechanical Engineering, pp 67-72, August, 1982.

[Cox-86]

Cox G. and Kumar S. "*Field Modelling of Fire in Forced Ventilated Enclosures*", Combustion Science and Technology, 52, 7, 1986.

[Cox-95]

Cox, G., Compartment fire modelling. Chapter 6 of "*Combustion Fundamentals of Fire*", edited by Cox, G., Academic Press, 1995.

[CUBIT]

CUBIT Mesh Generation Toolkit, web site: http://endo.sandia.gov/SEACAS/CUBIT/Cubit.html

[Delaunay-34]

Delaunay B. N., "*Sur la Sphere*" Vide. Izvestia Akademia Nauk SSSR, VII Seria, Otdelenie Matematicheskii i Estestvennyka Nauk vol. 7 pp 793-800, 1934.

[Dorr-84]

Dorr M.R. *"The approximation theory for the p-version of the finite element method"*, I. SIAM Journal on Numerical Analysis 21(1984), 1180-1207.

[Eagles-98]

Eagles N., "*Evaluation of the STAR-CD commercial CFD code for use in the Thermal Systems and Fluid Systems Groups* ", Rolls-Royce plc, Technical Design Report No. DNS 51528, 1998.

[Ewer98]

Ewer J., Galea E., Knight B., Patel M., Janes D. and Petridis M., "*Fire Field Modelling using the SMARTFIRE Automated Solution Control Environment*", University of Greenwich, CMS Press, 1998.

[Ewer99a]

Ewer J., Galea E. R., Patel M., Taylor, S., Knight B., Petridis M., "*SMARTFIRE: An Intelligent CFD Based Fire Model*", Journal of Fire Protection Engineering, 10 (1), pp 13-27, 1999.

[Ewer-99b]

Ewer J., Galea E., Patel M. and Knight B., "*The Development and Application of Group Solvers in the SMARTFIRE Fire Field Model*", Proceedings of Interflam 99, Edinburgh, UK, 1999, Vol. 2, pp 939-950.

[Ewer-00]

Ewer J, "*An Investigation into the Feasibility, Problems and Benefits of Re-engineering a Legacy Procedural CFD Code into an Even Driven, Object Oriented System that allows Dynamic User Interaction*", Ph.D. Thesis, The University of Greenwich, CMS Press, July 2000.

[Fagrell-98]

Fagrell, O. and Robinson, J.E., "*Low cost building wire in a large scale fire*", Plastic in telecommunication VIII Proceedings, London, Sept. 1998.

[Flahety-00]

Flaherty J.E. *"Adaptive Finite Element Techniques",* Chapter 8, Finite Element Analysis lecture notes. 2000.

[FLOW3D-91]

"*FLOW3D Release 2.3.3 Reference Guide*", CFD Dept AEA Harwell UK, Feb 1991.

[FLUENT-6-2]

"*FLUENT 6.2 Documentation",* Fluent, http://202.41.85.84/doc/fluent6.2/help/index.htm

[FLUENT-web]

"*CFD for the Mixing industry*", Fluent, http://www.fluent.com/solutions/brochures/mixing.pdf.

[Friday-01]

Friday, P. A., Mowrer, F. W., "*Comparison of FDS Model Predictions with FM/SNL Fire Test Data*", (3848 K) NIST GCR 01-810; 104 p. April 2001.

[Galea-89]

Galea E.R., "*On the field modelling approach to the simulation of enclosure fires*", Journal of Fire Protection Engineering, Vol 1 (1), pp11-22. 1989.

[Galea-93]

Galea E. R. and Ierotheou C. "*A parallel implementation of a general purpose fluid flow code and its application to Fire Fired Modelling*", Proceed. Parallel Computing 1991. Editors: Joubert, Evans and Liddel. Elsevier press 1993.

[Galea-97]
Galea E. R., "*The use of mathematical modelling in fire safety engineering*", paper No. 97/IM/25, CMS Press, University of Greenwich.

[George-91]
George P.L., Hecht F. and Saltel E., "*Automatic Mesh Generator with Specified Boundary*", Computer Methods in Applied Mechanics and Engineering, North-Holland, vo.l 92, pp.269-288, 1991.

[Giarratano-04]
Giarratano J. and Riley G., "*Expert Systems -- Principles and Programming*", 4th Edition, Thomson/PWS Publishing Company, 2004.

[Grandison -03]
Grandison A., "*Improving the regulatory acceptance and numerical performance of CFD based Fire-Modelling software*", Ph.D. Thesis, The University of Greenwich, CMS Press, 2003.

[GridPro]
"*GridPro TIL Manual*" – Program Development Company, White Plains, NY

[Hurst-04]
Hurst N., Ewer J., Grandison A., and Galea E., "*Group solvers: A means of reducing run-times and memory overheads for CFD based fire simulation software*", Proceedings of Interflam 2004, Edinburgh, UK, Vol.1, pp 659-664, 2004.

[Jacquotte-99]
Jacquotte O. P., "*Grid Optimization Efforts for Quality Improvement and Adaptation*", Handbook of Grid Generation, CRC Press, Boca Raton, pp. 33-1 to 33-33, 1999.

[Janes-02]
Janes, D. S., "*Intelligent control system for CFD modelling software*", Ph.D. Thesis, The University of Greenwich, CMS Press, 2002.

[Karlsson-00]

Karlsson B. and Quintiere J. G., "*Enclosure fire dynamics*", CRC Press, 2000.

[Kevin-02]

Kevin B. McGrattan, Howard R. Baum, Ronald G. Rehm, Anthony Hamins, Glenn P. Forney, Jason E. Floyd, Simo Hostikka, Kuldeep Prasad, "*Fire Dynamics Simulator-Technical Reference Guide*", National Institute of Standards and Technology, Gaithersburg, MD., NISTIR 6783, 2002.

[Kolodner -93]

Kolodner J. "*Case-based reasoning*", San Mateo, CA, Morgan Kaufmann, 1993.

[Kumar-91]

Kumar S., Gupta A.K. and Cox G., "*Effects of Thermal Radiation on the Fluid Dynamics of Compartment Fires*", Fire Safety Science, Proceedings of third International Symposium on Fire Safety Science., pp345-354, 1991.

[Kumar-01]

Kumar S., Cox G, "*Some guidance on 'correct' use of CFD models for fire applications with examples*", Interflam'2001 Vol. 1, pp823-834, 2001.

[Lawson-77]

Lawson C. L., "*Software for C1 Surface Interpolation*", Mathematical Software III, pp.161-194, 1977.

[Lewis-97]

Lewis M.J., Moss M.B. and Rubini P.A., "*CFD Modelling of Combustion and Heat Transfer in Compartment Fires*", Fire Safety Science, Proceedings of 5th International Symposium on Fire Safety Science, Ed: Hasemi Y., pp463-474, 1997.

[Liu-03]

Liu Y., Moser A., Gubler D. and Schaelin A., "*Influence of time step length and sub-iteration number on the convergence behavior and numerical accuracy for transient CFD*", CFD 2003, 11th Annual Conf. of the CFD Society of Canada, Vancouver, British Columbia, Canada, May 2003.

[Lo-91]

Lo S. H., "*Volume Discretization into Tetrahedra - II. 3D Triangulation by Advancing Front Approach*", Computers and Structures, vol. 39, no 5, pp.501-511, 1991.

[Lohner-96]

Lohner R., "*Progress in Grid Generation via the Advancing Front Technique*", Engineering with Computers, vol. 12, pp.186-210, 1996.

[Lu-01]

Lu Y., Gadh R. and Tautges T. J., "*Feature based hex meshing methodology: feature recognition and volume decomposition*", Computer-Aided Design, Volume 33, Issue 3, pp 221-232, 1 March 200.

[McRae-01]

McRae S.D., "*Adaptive mesh algorithms - A review of progress and future research needs*" AIAA Computational Fluid Dynamics Conference, 15th, Anaheim, CA, June 11-14, 2001.

[Morvan-05]

Morvan H. P. "*Automating CFD for non-experts*", Journal of Hydroinformatics 7, pp17-29, 2005.

[Nielsen-00]

Nielsen C., "*An Analysis of Pre-Flashover Fire Experiments with Field Modelling Comparisons*", Fire Engineering Research Report, School of Engineering, University of Canterbury, 2000.

[Owen-98]
Owen, S. J., "*A Survey of Unstructured Mesh Generation Technology*", 7th International Meshing Roundtable, Dearborn, Michigan, Oct. 26-28, 1998.

[Pantakar-80]
Pantakar S. V., "*Numerical Heat Transfer and Fluid Flow*", Intertext Books, McGraw Hill, New York, 1980.

[Petridis-95]
Petridis M, PhD Thesis: *Integrating an Intelligent Knowledge Based System into CFD Software*, University of Greenwich, CMS School, UK, 1995.

[Pirzadeh-93]
Pirzadeh S.,"*Unstructured Viscous Grid Generation by Advancing-Layers Method*", AIAA-93-3453-CP, AIAA, pp 420-434, 1993.

[Prahl-75]
Prahl J., and Emmons H. W., "*Fire Induced Flows through an Opening*", Combustion and Flames, 25, 369, 1975.

[Quitiere-84]
Quitiere J., "*A perspective on Compartment Fire Growth*", Combustion Science and Technology, Vol 39, pp. 11-54, 1984.

[Quintiere-89]
Quintiere J., "*Fundamentals of Enclosure Fire 'zone' Models*", Journal of Fire Protection Engineering, 1:99-119, 1989.

[Rajagopalan-03]
Rajagopalan K., Eiseman P., "*Automatic Nested Refinement - A Technique for the Generation of High Quality Multi-block Structured Grids for Multi-Scale Problems Using Gridpro*", 12th International Meshing Roundtable September 14-17 2003.

[Roache-98]

Roache, P. J., *"Verification and Validation in Computational Science and Engineering"*, Hermosa Publishers, New Mexico, 1998.

[Rugg-92]

Rugg G. *et al.*, "*A comparison of sorting techniques in knowledge elicitation*", Knowledge Acquistion **4** (1992) (3), pp. 279–291.

[Sarah]

http://www.cham.co.uk/phoenics/d_polis/d_enc/enc_s.htm#sarah

[SFPE-02]

"*Fire Plumes, Flame Height, and Air Entrainment*", in the SFPE Handbook of Fire Protection Engineering, 3rd Ed., National Fire Protection Association, 2002.

[Shah-01]

Shah J., Anderson D., Kim Y. and Joshi S., "*A Discourse on Geometric Feature Recognition From CAD Models*", Journal of Computing and information Science in Engineering, Vol.1, pp 41-51, 2001.

[Shephard-91]

Shephard M. S., Georges M. K., (1991) "*Three-Dimensional Mesh Generation by Finite Octree Technique*", International Journal for Numerical Methods in Engineering, vol. 32, pp 709-749, 1991.

[Shephard-04]

Shephard M. S., Seol E., Wan J. and Bauer A. C., "*Component-based Adaptive Mesh Control Procedures*", chapter 1, Research report 2004, Scientific Computation Research Centre, Rensselaer Polytechnic Institute.

[Sibson-78]

Sibson R., "*Locally equiangular triangulations*", The Computer Journal 21(3), pp 243-245, 1978.

[Slack-03]
Slack M. D., Porte S. D. and Engleman M. S., "*Designing Automated Computational Fluid Dynamics Modelling Tools for Hydrocyclone Design*", Hydrocyclone, 2003.

[Smith-85]
Smith G. D., "*Numerical solution of partial differential equations: finite difference methods*" 3rd Edition, Oxford: Clarendon, 1985.

[Spalding -81]
Spalding D.B., "*A General Purpose Computer Program For Multi-Dimensional One- and Two-Phase Flow*", Mathematics and Computers in Simulations, North Holland (IMACS), Vol. XXIII, 1981, 267.

[Spalding-04]
Spalding B., April 2004, "*CONWIZ: The Convergence-Promoting Wizard*", PHOENICS User Conference, Melbourne, 2004.

[STAR-CD-web]
"*The CFD Expert System for combustion system design in IC engines*", CD adapco Group, http://www.cd-adapco.com/products/brochuredownload.htm.

[Steckler-82]
Steckler, K. D., Quintiere, J. G., and Rinkinen, W. J., "*Flow Induced by Fire in A Compartment*", NBSIR-822520, National Bureau of Standards, Gaithersburg, MD, 1982.

[Stuben-97]
Stuben K., "*Europort-D: paramllel computing for European industry*", IEEE Concurrency, 5(4):7-10, 1997.

[Tautges-97]
Tautges T.J., Liu S., Lu Y., Kraftcheck J., Gadh R., "*Feature Recognition Applications in Mesh Generation*", AMD-Vol. 220 Trends in Unstructured Mesh Generation, ASME, pp117-121,1997.

[Taylor-97a]

Taylor S., Petridis M., Knight B., Ewer J., Galea E.R. and Patel M., "*SMARTFIRE: An Integrated CFD code and expert system for fire field modelling*", Fire Safety Science, Proceedings of the 5th Int. Symp., Ed: Hasemi Y., pp 1285-1296, 1997.

[Taylor-97b]

Taylor S., "*An investigation into Automation of Fire Field Modelling Techniques*", PhD Thesis,University of Greenwich, UK, 1997.

[Thompson-85]

Thompson J.F, Warsi Z.U.A. and Mastin C.W., "*Numerical grid generation: foundations and applications*", Elsevier North-Holland, Inc. New York, NY, USA 1985.

[Tristano-03]

Tristano J. R., Chen Z., Hancq D. A. and Kwok W., *"Fully automatic adaptive mesh refinement integrated into the solution process"*, Proceedings, 12th International Meshing Roundtable, Sandia National Laboratories, pp.307-314, Sept. 2003.

[Tstt]

http://www.tstt-scidac.org

[Vavasis]

Vavasis S. A., QMG web site: http://simon.cs.cornell.edu/Info/People/vavasis/qmg-home.html

[Verfurth-96]

Verfurth R., *"A Review of Posterior Error Estimation and Adaptive Mesh Refinement Techniques"*, Teubner-Wiley, Stuttgart, 1996 .

[Versteeg-95]

Versteeg H.K., Malalasekera W., "*An introduction to computational fluid dynamics, The finite volume method*",Longman, Scientific & Technical, Essex, 1995.

[Walton-02]

Walton, W. D., "*Zone computer Fire Models for Enclosures*", Chapter 7; Section 3; NFPA HFPE-02; SFPE Handbook of Fire Protection Engineering. 3rd Edition, DiNenno, P. J.; Drysdale, D.; Beyler, C. L.; Walton, W. D., Editor(s), 3/189-193 p., 2002.

[Watson-81]

Watson D. F.,"*Computing the Delaunay Tesselation with Application to Voronoi Polytopes*", The Computer Journal, vol. 24(2) pp.167-172, 1981.

[Watson-97]

Watson I. D., "*Applying case-based reasoning: techniques for enterprise systems*", San Francisco, Calif.: Morgan Kaufmann, 1997.

[Weatherill-94]

Weatherill N. P., Hassan O., "*Efficient Three-dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints*", International Journal for Numerical Methods in Engineering, vol 37, pp.2005-2039, 1994.

[White-95]

White D. R., "*Automated hexahedral mesh generation by virtual decomposition*", Proceedings of the 4th International Meshing Roundtable, Sandia National Laboratories, pp165-176, 1995.

[William-02]

William D. Walton "*Computer Fire Models for Enclosures*", The SPE Handbook of Fire Protection Engineering Third Edition, Chapter 7, Section, 3, pp. 3/189-193, 2002.

[Wyman-01]

Wyman N. "*State of the Art in Grid Generation* ",CFD Review,www.cfdreview.com

[Xue-01]

Xue, H., Ho, J. C. and Cheng, Y. M., "*Comparison of Different Combustion Models in Enclosure Fire Simulation*", Fire Safety Journal, Vol. 36(1), pp.37-54, 2001.

[Yerry-84]

Yerry M. A., Shephard M. S., "*Three-Dimensional Mesh Generation by Modified Octree Technique*", International Journal for Numerical Methods in Engineering, Vol. 20, pp1965-1990, 1984.

[Yoshimura-99]

Yoshimura S., Wada Y., Yagawa G., "*Automatic mesh generation of quadrilateral elements using intelligent local approach*", Computer Methods in Applied Mechanics and Engineering. 179 pp125-138, 1999.

[Zhang-96]

Zhang H., Moallemi M.K., and Prasad V., "*A numerical algorithm using multizone grid generation for multiphase transport processes with moving and free boundaries*", Numerical Heat Transfer, B29:399--421, 1996.

[Zhang-04]

Zhang J. , Ewer J., JIA F., GrandisonA., Galea E., "*SMARTFIRE verification and validation report*", August 2004.

Appendix A - coefficients and source terms

The coefficients and source terms for the essential equations of FFM are listed in Table A-1.

Where,

$$\mu_e = \mu + \mu_t, \qquad \mu_t = C_\mu \rho k^2 / \varepsilon, \qquad \sigma_\varepsilon = k^{2/3} / (C_2 - C_1) C_\mu^{1/2},$$

$$\Gamma_e = \mu_t / \sigma_t + \mu / \sigma_l, \quad \Gamma_c = \mu_t / \sigma_c + \mu / \sigma_l, \quad \Gamma_k = \mu_e / \sigma_k, \quad \Gamma_\varepsilon = \mu_t / \sigma_\varepsilon$$

$$\Gamma_h = \mu_e / \sigma_h \ , \ G_B = \beta g \frac{\mu_t}{\sigma_t} \frac{\partial T}{\partial y}$$

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

C_m	C_D	C_1	C_2	σ_c	σ_k	σ_h	σ_e
0.09	1.0	1.44	1.92	0.9	1.0	0.9	1.3

Equation	Variable ϕ	Diffusivity Γ_ϕ	Source term S_ϕ
Continuity	1	0	0
u-momentum	u	μ_{eff}	$-\frac{\partial \rho}{\partial x}+\frac{\partial}{\partial x}\left(\mu_{eff}\frac{\partial U}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu_{eff}\frac{\partial V}{\partial x}\right)+\frac{\partial}{\partial z}\left(\mu_{eff}\frac{\partial W}{\partial x}\right)-\frac{2}{3}\frac{\partial}{\partial x}(\rho k)$
v-momentum	v	μ_{eff}	$-\frac{\partial \rho}{\partial y}+\frac{\partial}{\partial x}\left(\mu_{eff}\frac{\partial U}{\partial y}\right)+\frac{\partial}{\partial y}\left(\mu_{eff}\frac{\partial V}{\partial y}\right)+\frac{\partial}{\partial z}\left(\mu_{eff}\frac{\partial W}{\partial y}\right)-\frac{2}{3}\frac{\partial}{\partial y}(\rho k)+g(\rho-\rho_0)$
w-momentum	w	μ_{eff}	$-\frac{\partial \rho}{\partial z}+\frac{\partial}{\partial x}\left(\mu_{eff}\frac{\partial u}{\partial z}\right)+\frac{\partial}{\partial y}\left(\mu_{eff}\frac{\partial V}{\partial z}\right)+\frac{\partial}{\partial z}\left(\mu_{eff}\frac{\partial W}{\partial z}\right)-\frac{2}{3}\frac{\partial}{\partial z}(\rho k)$
Enthalpy	h	Γ_e	S_h
Concentration	C	Γ_c	S_c
Kinetic energy	K	Γ_k	$G_k - C_p\rho\varepsilon + G_B$
Dissipation	ε	Γ_ε	$C_1\frac{\varepsilon}{k}(G_k+G_B)(1+C_3R_f)+C_2\rho\frac{\varepsilon^2}{k}$

Table A-1: The coefficients and source terms of essential equations in FFM

Appendix B - USER GUIDE FOR THE *SMARTFIRE* EXPERIMENT ENGINE

PURPOSE OF SECTION

This section contains the User Guide for the ***SMARTFIRE*** Experiment Engine Solution Control system. This new component links the existing ***SMARTFIRE*** Case Specification Environment (CSE) and CFD Engine seamlessly and is able to generate run-time tests with an improved configuration, according to analysis of the error yielded from previous configurations, until a satisfactory configuration is found or all possible improvement options are exhausted. The Experiment Engine also continually monitors the solution process after applying appropriate set-up and meshing during the experimental phase of the EE controlled solution process. The EE is able to make run time adjustments on the solution control parameters (i.e. time step size) – as and when necessary – in order to increase the success rate and quality of the *SMARTFIRE* simulations.

The purpose behind this section is to describe the Experiment Engine mode of operation and the meaning and purpose of the Experiment Engine Solution Control parameters and options, are also described. The section starts with a general description of the Experiment Engine in terms of its mode of operation. A description of the meanings of the Experiment Engine Solution Control parameters and their available options follows. It is very important for the User to understand the purpose, meaning and effect of the Experiment Engine Solution Control parameters and the capabilities of the tool, in order to successfully run the simulation scenario while using the Experiment Engine.

GENERAL DESCRIPTION AND MODE OF OPERATION

The *SMARTFIRE* Experiment Engine Solution Control system has been developed to emulate an experts' ability to analyse a series of trials starting from a simplified, coarse mesh simulation or even to learn something from a complete failure to make a better set-up and/or meshing solution for a particular case. The EE is also intended to emulate an experts' ability to control simulation solutions and provide similar benefits in terms of performance, overall solution reliability and results accuracy.

All of the user interactions with the Experiment Engine (including starting the EE, examining the output of the solution status/state – in term of validity and reasonableness from the Experiment Engine, and the settings of the EE Solution Control parameters) are accessible from the main user interface of the ***SMARTFIRE*** Case Specification Environment (CSE). The CSE generally acts as a pre-processor for set-up and specification of a case and for meshing of scenario.

CONDUCT EXPERIMENTS MENU

The mode of operation to the Experiment Engine is quite straight forward. There are only two options used to run the Experiment Engine from the CSE. One is called "*Conduct Experiments*", which makes full use of the Experiment Engine to control the ***SMARTFIRE*** solution process. The other one is "*Run Production*" option, which is disabled by default, as it is an advanced option intended only for expert users (this option will be described later). As shown in Figure 1, the "normal" mode of the Experiment Engine operation can be activated by selecting [RUN] => [Conduct Experiments] command from the main menu of the CSE.

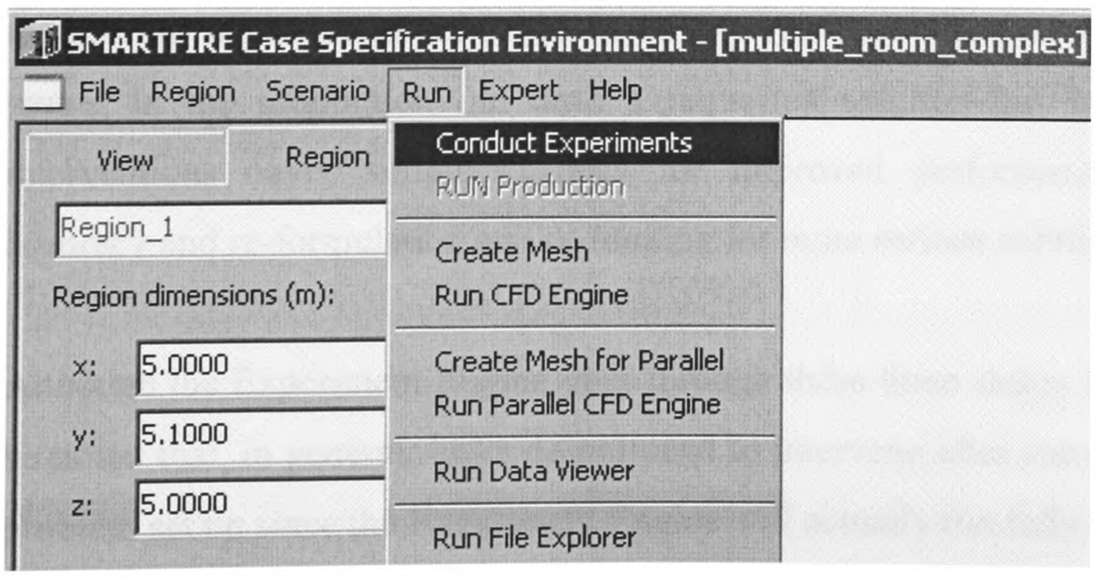


Figure 1: Conduct Experiments Menu.

If this mode of operation is selected, then the whole ***SMARTFIRE*** simulation process will step through three distinct stages.

The first stage is the mesh testing stage. In this stage, the main focus is to make sure that the mesh quality is adequate/appropriate for achieving the required solution with the desired accuracy.

The second stage is to perform time step size testing. Once the Experiment Engine controlled simulation has entered this stage, the flows and other solution characteristics of the scenario are assumed to be reasonably well developed, so that the Experiment Engine can try a range of reasonable time step sizes to find out which time step size gives the best solution convergence behaviour and is suitable for the preferred solution accuracy. In practice the EE has to reconfigure some of the scenario conditions and options in order to perform these tests – however these changes will not be taken forward into the production run since they are merely for testing for appropriate time stepping. The EE then selects the "best" time step size along with the suitable mesh, established in the first stage, to be chosen for configuring a production run configuration.

In the third and final stage of the Experiment Engine controlled simulation, the original problem definition that was set-up by the users is restored (thus undoing any simplification measures – if any were used) during the experimental phase and mesh settings and the selection of the time step size. These selections are used to start the "real" production simulation run. The Experiment Engine will also continually monitoring the solution state and make any run-time adjustments if anything goes wrong in the production run until a requested solution has been achieved. The interventions cover simple changes for improved performance and/or solution accuracy and re-formulation and re-running for more serious solution failures.

Although the Experiment Engine goes through these three stages internally, it should be noted that, in general, users do not need to intervene after completing the original problem set up since the Experiment Engine will actually run fully automatically.

Experiment Engine Solution Controls

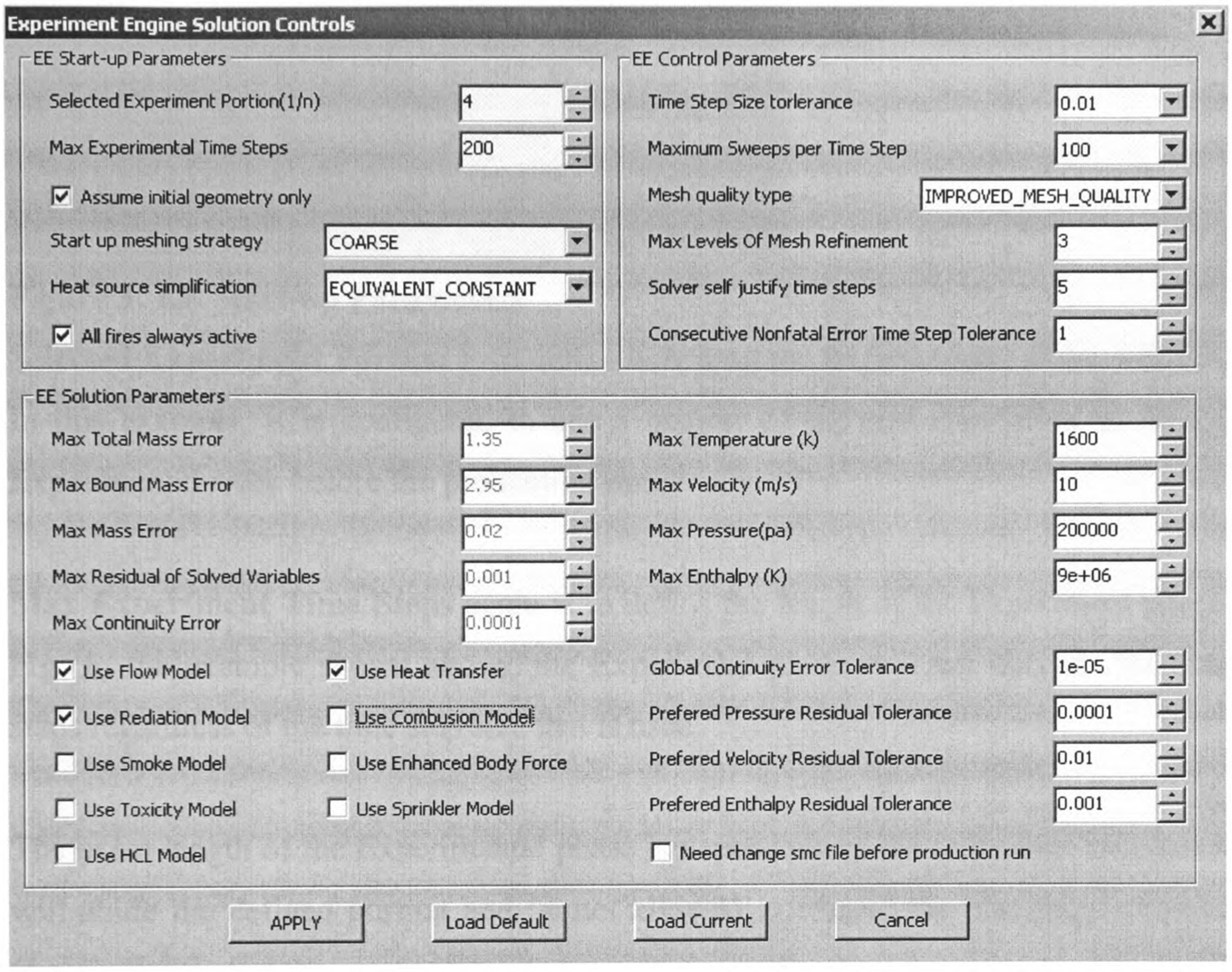
The Experiment Engine is trying to emulate an experts' ability to assess the set-up parameters, the quality of meshing and the solution state. In order to achieve this, first and foremost, the Experiment Engine needs to have some rules to decide whether the mesh and other scenario settings are suitable and, during the run time, whether the solution state is healthy etc. Secondly, the Experiment Engine needs to know what actions can and ought to be taken, and to what degree, if things go wrong.

The "default" Experiment Engine (EE) operation is pre-configured by a set of parameters and instructions stored in the "smf_exp.ini" file in the ***SMARTFIRE*** ini folder.

For a typical fire scenario, the values of these parameters are usually sufficient, thus modifications to the values of the parameters are not absolutely necessary. However, for Expert users who require more optimal control of the Experiment Engine operation for a particular fire scenario, the Experiment Engine will also automatically save a copy of these parameters in a file called smf_exp_this.ini in current case folder. The values of the parameters stored in the current folder are specific to the case that is being simulated. In this way, Expert users can experiment with their own optimal value settings for these parameters, for the case at hand, without affect the "global" default EE parameter settings stored in the ini folder.

The values of these parameters can be changed in two ways. First, users can change the parameter values by manually editing the "smf_exp_this.ini" file directly (please note that normally users should not need to modify the "smf_exp.ini" file directly) and then load it into the ***SMARTFIRE*** by clicking [*Load Current*] button in the Experiment control window in the ***SMARTFIRE*** CSE (see Figure 2). Second (actually the preferred method) is to change the values by using the Experiment control window interface. When users click the [*Apply*] button to confirm the changes, the "smf_exp_this.ini" file is automatically updated and saved (See Figure 2).

The following snapshot shows the Experiment Control window in ***SMARTFIRE*** CSE. Selecting [Scenario] => [Experiment Control....] command on the main CSE to access to this window.


Experiment Engine Solution Controls

EE Start-up Parameters

Selected Experiment Portion(1/n): 4
Max Experimental Time Steps: 200
☑ Assume initial geometry only
Start up meshing strategy: COARSE
Heat source simplification: EQUIVALENT_CONSTANT
☑ All fires always active

EE Control Parameters

Time Step Size torlerance: 0.01
Maximum Sweeps per Time Step: 100
Mesh quality type: IMPROVED_MESH_QUALITY
Max Levels Of Mesh Refinement: 3
Solver self justify time steps: 5
Consecutive Nonfatal Error Time Step Tolerance: 1

EE Solution Parameters

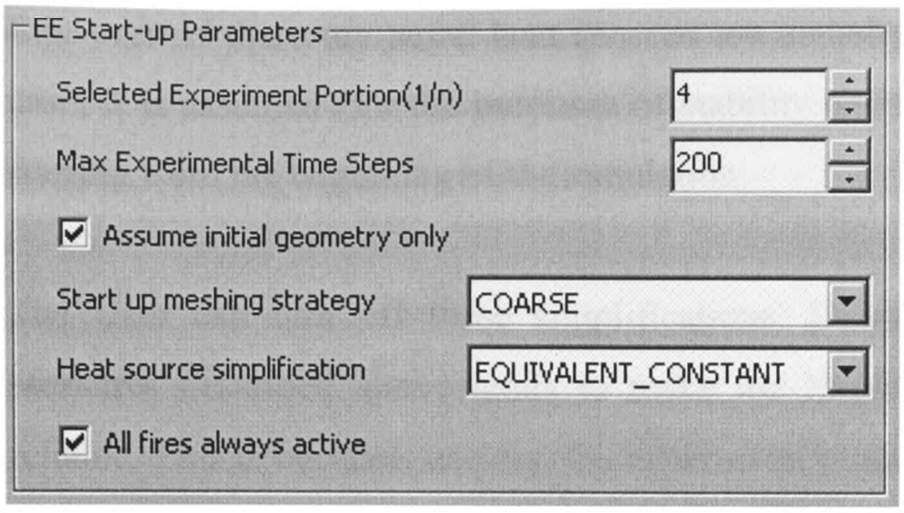
Max Total Mass Error: 1.35
Max Bound Mass Error: 2.95
Max Mass Error: 0.02
Max Residual of Solved Variables: 0.001
Max Continuity Error: 0.0001
Max Temperature (k): 1600
Max Velocity (m/s): 10
Max Pressure(pa): 200000
Max Enthalpy (K): 9e+06
Global Continuity Error Tolerance: 1e-05
Prefered Pressure Residual Tolerance: 0.0001
Prefered Velocity Residual Tolerance: 0.01
Prefered Enthalpy Residual Tolerance: 0.001

☑ Use Flow Model
☑ Use Heat Transfer
☑ Use Rediation Model
☐ Use Combusion Model
☐ Use Smoke Model
☐ Use Enhanced Body Force
☐ Use Toxicity Model
☐ Use Sprinkler Model
☐ Use HCL Model
☐ Need change smc file before production run

Figure 2: Experiment Engine Solution Controls window

As shown in the figure above, there are three sub-sections of parameters, namely EE Start-up, EE Control and EE Solution parameters.

EE Start-up Parameters control how the first Experiment test case is formulated. And this section of parameters will only affect the Experimental phase (involving mesh testing and time step size testing stages) of the Experiment Engine operation.


EE Start-up Parameters

Selected Experiment Portion(1/n): 4

Max Experimental Time Steps: 200

[✓] Assume initial geometry only

Start up meshing strategy: COARSE

Heat source simplification: EQUIVALENT_CONSTANT

[✓] All fires always active

Figure 3: EE Start-up Parameters

Selected Experiment Portion is to define how long the Experimental phase will run. In this example, it is configured to use a quarter of the real simulation to do the Experimental work before the production run.

Max Experiment Time Steps again is to define the length of the Experiment phase. E.g. In the example shown above, the Experiment phase will not exceed 200 time steps regardless of the time step size that is used.

The exact length of the Experimental phase is dependant on the individual case but it will abide the defined portion and be not allowed to exceed the maximum allowed experiment time steps.

In a complex real life fire scenario, sometimes, the geometric objects in the domain of interests will change during the course of simulation, i.e. opening doors, breaking windows etc. Turning on **Assume initial geometry only** option will simplify the scenario by assuming that these geometry changes will not happen, hence making the scenario less complex in the Experimental phase.

Start-up meshing strategy is used to define what sort of initial base mesh will be used for the default test case. E.g. whether it is a coarse mesh, ***SMARTFIRE*** recommended mesh or an even finer mesh.

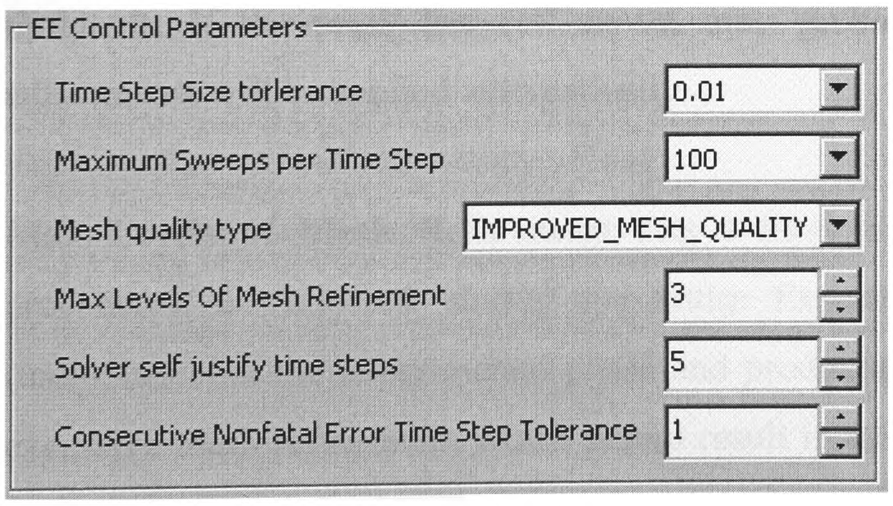
Heat source simplification and **All fires always active** are both again simplification measures. In the example shown in Figure 3, the heat release rate used in the experimental phase is assumed as a constant release throughout the simulated period.

E.g. even if there are some heat sources are actually activated later in the original case, it is assumed (for the purposes of stability and accuracy testing) that they also starting from the beginning of the simulation.

The User can turn off these simplifications. However, to use the simplification measures described above tends to make the Experiment Engine somewhat more reliable. This is because, usually, the experiment phase will not run for the full length of the originally configured simulation as this would be too costly. A partial experimental simulation, not using *heat source simplification* and *all fires always active* options, would miss the effects of fires that start – or reach peak output – at later times than simulated in the experiment phase.

EE Control Parameters

This section of parameters (see Figure 4) defines how the Experiment Engine will respond to changes in the solution state, and to what degree. This set of parameters will affect both the experiment and production phase of the Experiment Engine operation.



EE Control Parameters

Time Step Size torlerance	0.01
Maximum Sweeps per Time Step	100
Mesh quality type	IMPROVED_MESH_QUALITY
Max Levels Of Mesh Refinement	3
Solver self justify time steps	5
Consecutive Nonfatal Error Time Step Tolerance	1

Figure 4: EE Control Parameters

Time Step Size tolerance defines the smallest time step size that can be used to overcome a solution stability and divergence problem. This combo box actually stores a set of values, normally from 10 seconds, 5 seconds etc. to the smallest time step size tolerance. In the example shown here, the tolerance is 0.01 seconds. In the mesh testing stage of the experimental phase, the time step size will be adopted in a top down fashion. I.e. the Experiment Engine will first try 10 second time step size and if problems occur then time step size will go down to the next level (5 seconds). In the time step testing stage of the experimental phase, the Experiment Engine will verify only the time step size that is smaller than the current one. This means that if, at the end of the meshing test stage, the Experiment Engine found a 2 second time step size is acceptable, then in the time step size testing stage; it will only test from time step sizes of 1 second down to the predefined time step size tolerance.

Maximum Sweeps per Time Step defines the maximum sweeps per time step used in the production run. This is a combo box control that stores a list of the available values. The first one in the list (the smallest value, i.e. 50 sweeps, currently hidden in the combo box shown above) will always be used in the mesh testing stage of the experimental phase. In the example (shown in Figure 4) a maximum of 100 sweeps per time step, will be used in the time step size testing stage and production stage of the Experiment Engine run.

Mesh quality type defines how the mesh is being refined – if this is necessary. In this case, it has been set to IMPROVED_MESH_QUALITY, which means that, after each refinement, the Experiment Engine will automatically check for cell aspect ratio problems. If the mesh has cell aspect ratio problems, then the block-wise cell adjustments will be applied automatically.

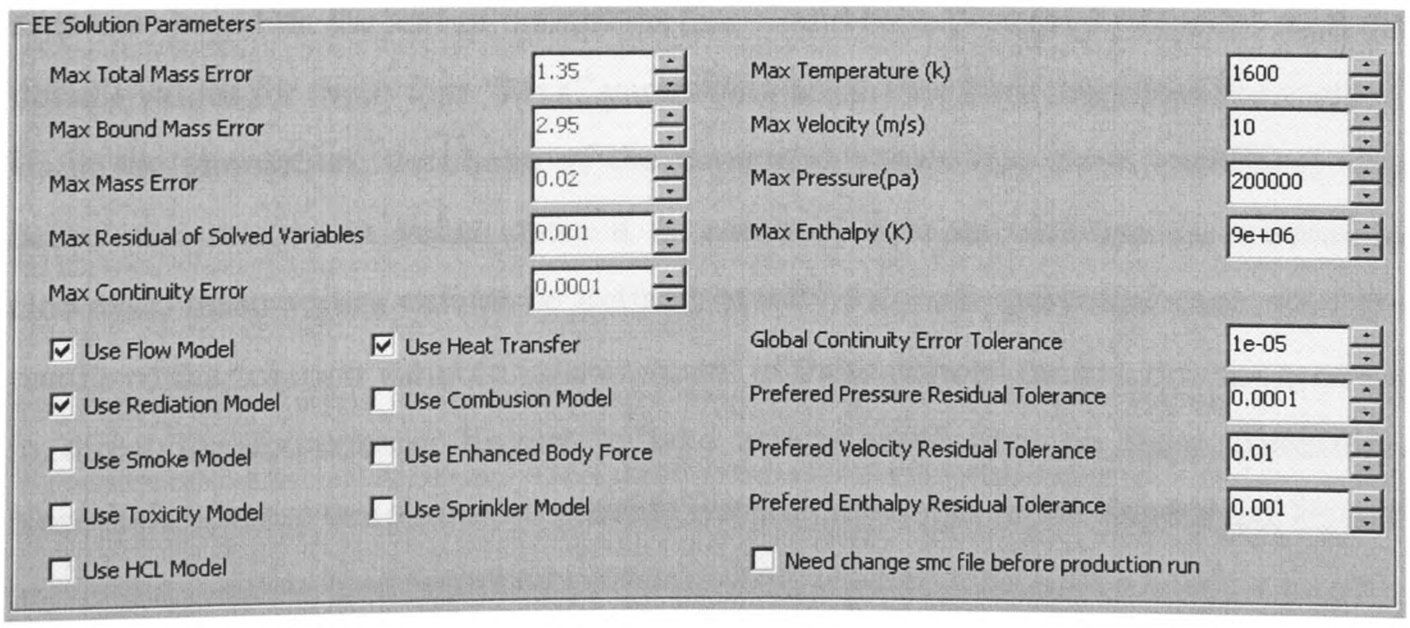
Max Levels of Mesh Refinement sets a limit on how many mesh refinement procedures are allowed, during the entire Experiment Engine solution process (including both the experimental phase and production phase). This option prevents excessive mesh refinement, which would result in using too many cells in the mesh. In the given example shown above, if the initial base mesh has been refined 3 times in the experimental phase, then the production phase of the operation will not be allowed to do any further mesh refinements.

Solver self justify time step informs the Experiment Engine to delay the response to the unstable simulation state in the beginning of both experimental and production phase, i.e. in a fire simulation, the first few time steps are usually somewhat unstable due to the poor choice of initial values, sudden effects of the boundary conditions experienced for the first time and lack of established flows, etc. The situation usually improves quite quickly after an initial period of the simulation. In the Example given in Figure 4, the Experiment Engine will not react to the seeming "poor quality" initial simulation state (if any) in the first 5 time steps unless something goes very seriously wrong (i.e. complete divergence). After this initial period, if the simulation state is still not improved to a satisfactory level, then the Experiment Engine will react to the solution state by either refining the mesh or changing the time step size to a smaller value – or even do both. The exact reaction from the Experiment Engine is dependant on the actual solution state.

Consecutive Nonfatal Error Time step tolerance is another measure that delays EE intervention to give the current solution strategy more time to solve a potential problem, before the EE will take remedial action. For instance, if there is a non fatal error, the EE does not necessarily have to respond to the problem in the next time step since the solution may improve and satisfy the solution criteria. However, in the example given in Figure 4, the tolerance has been set to be 1, which means that the EE will intervene when a problem is detected.

EE Solution Parameters

The last section of parameters (in the bottom half of Figure 2) is to define what constitutes a satisfactory solution status.



EE Solution Parameters

Max Total Mass Error	1.35	Max Temperature (k)	1600
Max Bound Mass Error	2.95	Max Velocity (m/s)	10
Max Mass Error	0.02	Max Pressure(pa)	200000
Max Residual of Solved Variables	0.001	Max Enthalpy (K)	9e+06
Max Continuity Error	0.0001		
Global Continuity Error Tolerance	1e-05	Prefered Pressure Residual Tolerance	0.0001
Prefered Velocity Residual Tolerance	0.01	Prefered Enthalpy Residual Tolerance	0.001

☑ Use Flow Model
☑ Use Heat Transfer
☑ Use Rediation Model
☐ Use Combusion Model
☐ Use Smoke Model
☐ Use Enhanced Body Force
☐ Use Toxicity Model
☐ Use Sprinkler Model
☐ Use HCL Model
☐ Need change smc file before production run

Figure 5: EE Solution Parameters

The first 5 parameters on the top left are currently inactive/greyed out, because these are not used in the current Experiment Engine control regime, although they may be used in a future version.

The model choices, below this, are a replication of the settings that have already been defined in the [scenario] => [problem type...] in normal operation of the ***SMARTFIRE***. These options override the previous choices made by user – in the problem type section – while the EE is operating. The reason for these duplicated model settings, is that the basic philosophy of the EE is to offer the facility for users to try things out. So it is very important that whatever experimentation settings that the users tried should not affect the original problem settings. Even if the user gets acceptable results from the production run, using modified settings, the user's original problem settings are preserved.

The parameters in this section will affect both the experimental and production phase. It is very important to remember to reselect the fire models intended to use in the experimental phase, so the automated production run can then use the appropriate models to produce the required results.

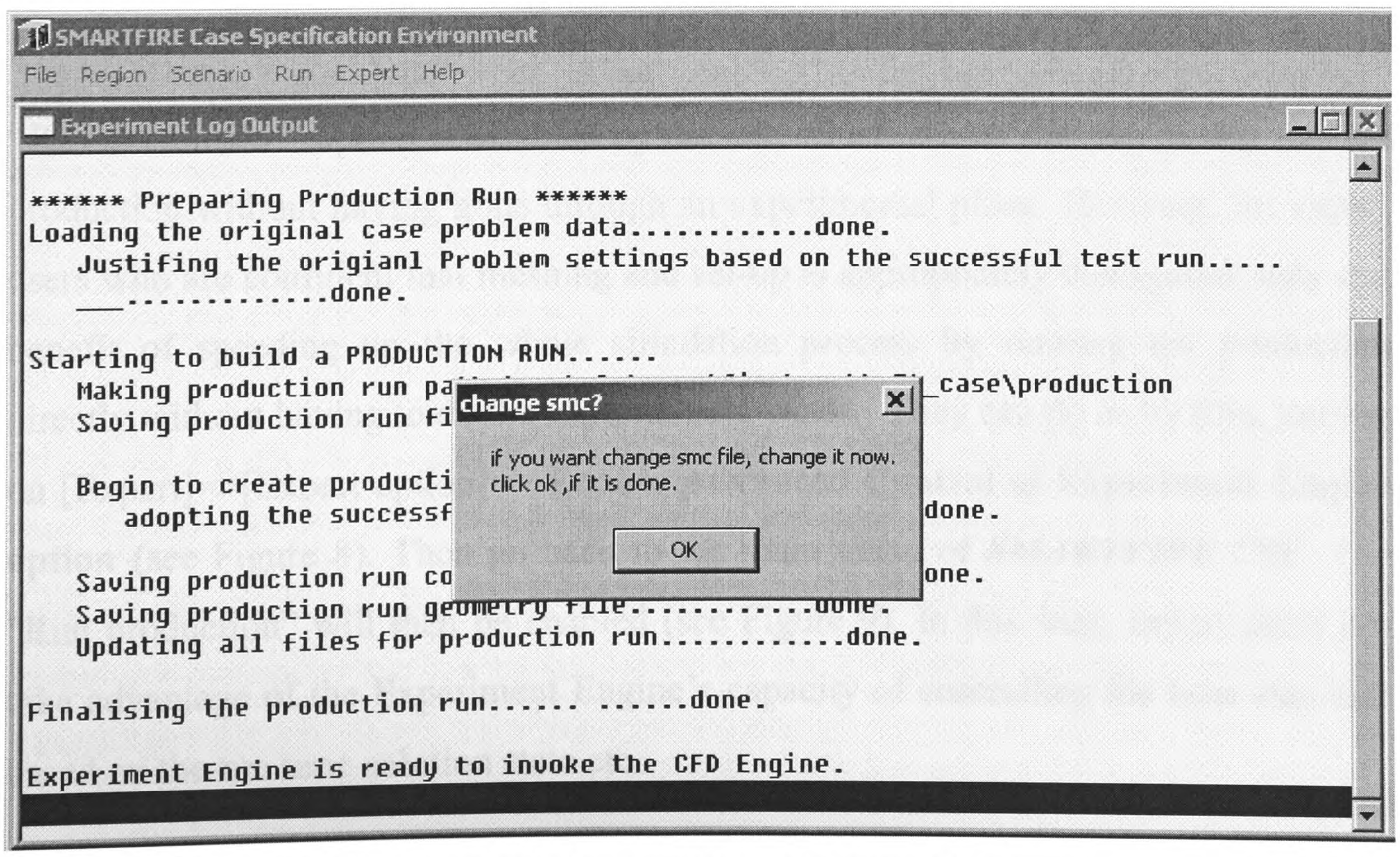
The four "Max" parameters set in the top right corner of Figure 5, namely **Max Temperature, Max Velocity, Max Pressure** and **Max Enthalpy** have been selected as very important variables that help to indicate a valid solution state (for a fire scenario) based on the sort of conditions that would be experienced in reality. And the default values for these four "Max" parameters are drawn from empirical fire science. If, in the simulation, the range of the computed values (for these quantities) falls below the maximum value, then it is assumed that the solution is reasonable. However, these values cannot be defined exactly for each individual case, so if the results of the solution variable values are out of these defined ranges, it is not sensible to direct the Experiment Engine to take action based only on these conditions. Nonetheless, these conditions do provide warning messages in the Experiment Engine log output window (see Figure 6).

The next group of parameters are concerned with the solution quality and accuracy. Aside from the divergence of the simulation – which will need EE intervention, violation of the following five parameters will definitely provoke an Experiment Engine reaction.

Global continuity error tolerance is the overall indicator of suitability of the mesh and time step size.

Preferred Pressure Residual Tolerance, Preferred Velocity Residual Tolerance and Preferred Enthalpy Residual Tolerance is again overriding the single convergence tolerance settings in the problem type section for normal operations. It seems more reasonable to set residual tolerance for specified group variables instead of setting one for all tolerance values.

The last option named "**need to change smc before production run**" is intended for use by Expert users in complex situations. For instance, to manually define heat sources in the .smc file for the production run. Once this option is selected, the Experiment Engine will wait until the user confirms the required changes have been made to .smc file by dismissing the following window showed in Figure 6, before the Experiment Engine enters the production phase of the EE solution process.


SMARTFIRE Case Specification Environment
File Region Scenario Run Expert Help
Experiment Log Output
change smc?
OK

```
****** Preparing Production Run ******
Loading the original case problem data............done.
   Justifing the origianl Problem settings based on the successful test run.....
   ___.............done.

Starting to build a PRODUCTION RUN......
   Making production run pa                                 _case\production
   Saving production run fi

   Begin to create producti
      adopting the successf                                 done.

   Saving production run co                                 one.
   Saving production run geometry file............done.
   Updating all files for production run............done.

Finalising the production run...... ......done.

Experiment Engine is ready to invoke the CFD Engine.
```

if you want change smc file, change it now.
click ok ,if it is done.

Figure 6: Need to change smc before production run option

Command Buttons

They are four buttons at the bottom of the Experiment control window (Figure 2) as shown in Figure 7.

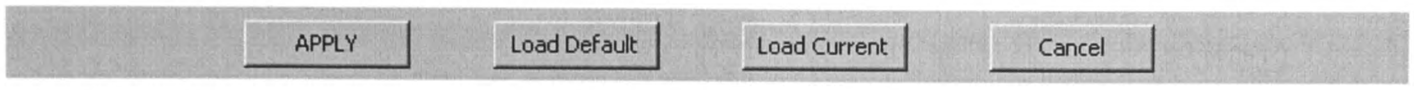
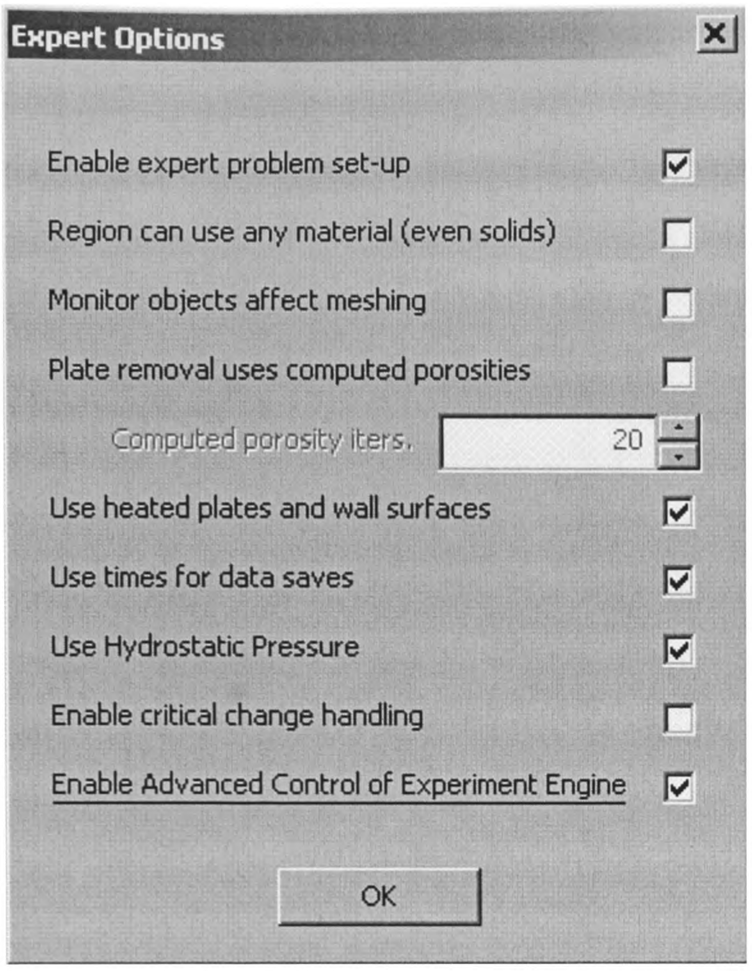


Figure 7: Command buttons on Experiment control window

The user confirms any changes made by clicking the [**APPLY**] button, and the case specific “smf_exp_this.ini” file will be updated. This file will be loaded when the user clicks the [**Load Current**] button. When [**Load default**] button is clicked, the Experiment Engine will always load the global default parameters stored in the file “smf_exp.ini” in ini folder. Finally the [**Cancel**] button dismisses any changes made. In summary, the Experiment Engine Solution Controls always maintains two sets of parameters at the same time, i.e. the global set in “smf_exp.ini” and the local set in “smf_exp_this.ini”. If the user sets poor choices for the parameter values that hinder the EE control, then these controls will allow them to go back to a stable configuration.

Advanced Experiment Engine Controls

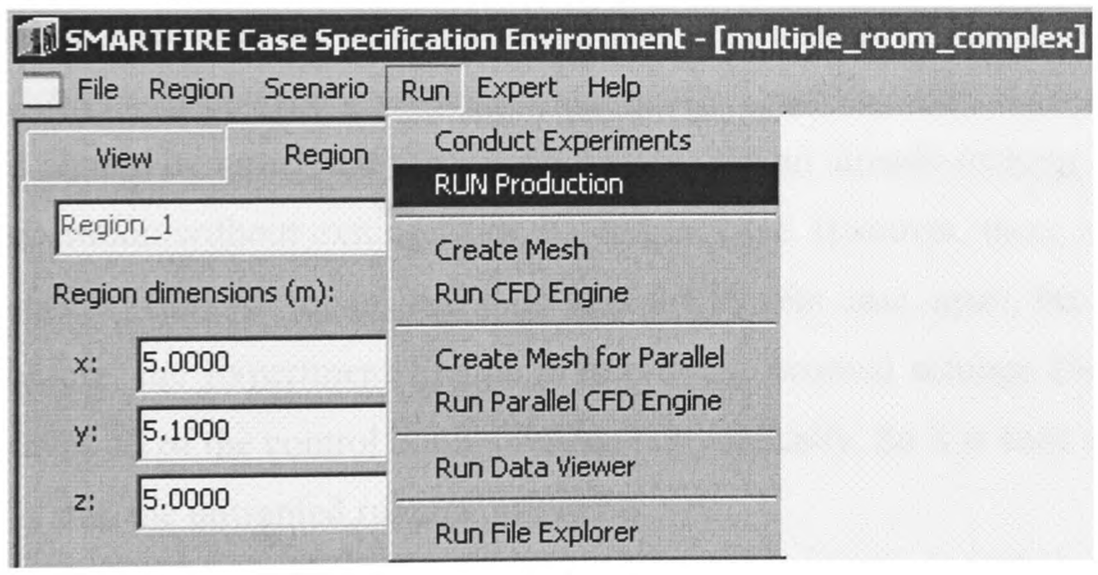
“*Run Production*” option is disabled by default, which means the user cannot run the production without having gone through an experimental phase. However, for expert users who are confident that meshing and set-up is appropriately configured, they can benefit of speeding up the whole simulation process by running the production directly without having to do the experimental work. They can do so by first, turning on [Expert]=>[Expert option]=> **Enable Advanced Control of Experiment Engine option** (see Figure 8). Then go back to the main menu of ***SMARTFIRE*** CSE. The “Run production” will then be enabled (see Figure 9). In this way, expert users can take advantage of the Experiment Engine’s capacity of controlling the time step size based on the run time solution state etc.



Expert Options

- Enable expert problem set-up ✓
- Region can use any material (even solids) ☐
- Monitor objects affect meshing ☐
- Plate removal uses computed porosities ☐
- Computed porosity iters. 20
- Use heated plates and wall surfaces ✓
- Use times for data saves ✓
- Use Hydrostatic Pressure ✓
- Enable critical change handling ☐
- Enable Advanced Control of Experiment Engine ✓

Figure 8: Expert options window



SMARTFIRE Case Specification Environment - [multiple_room_complex]

Region_1

Region dimensions (m):

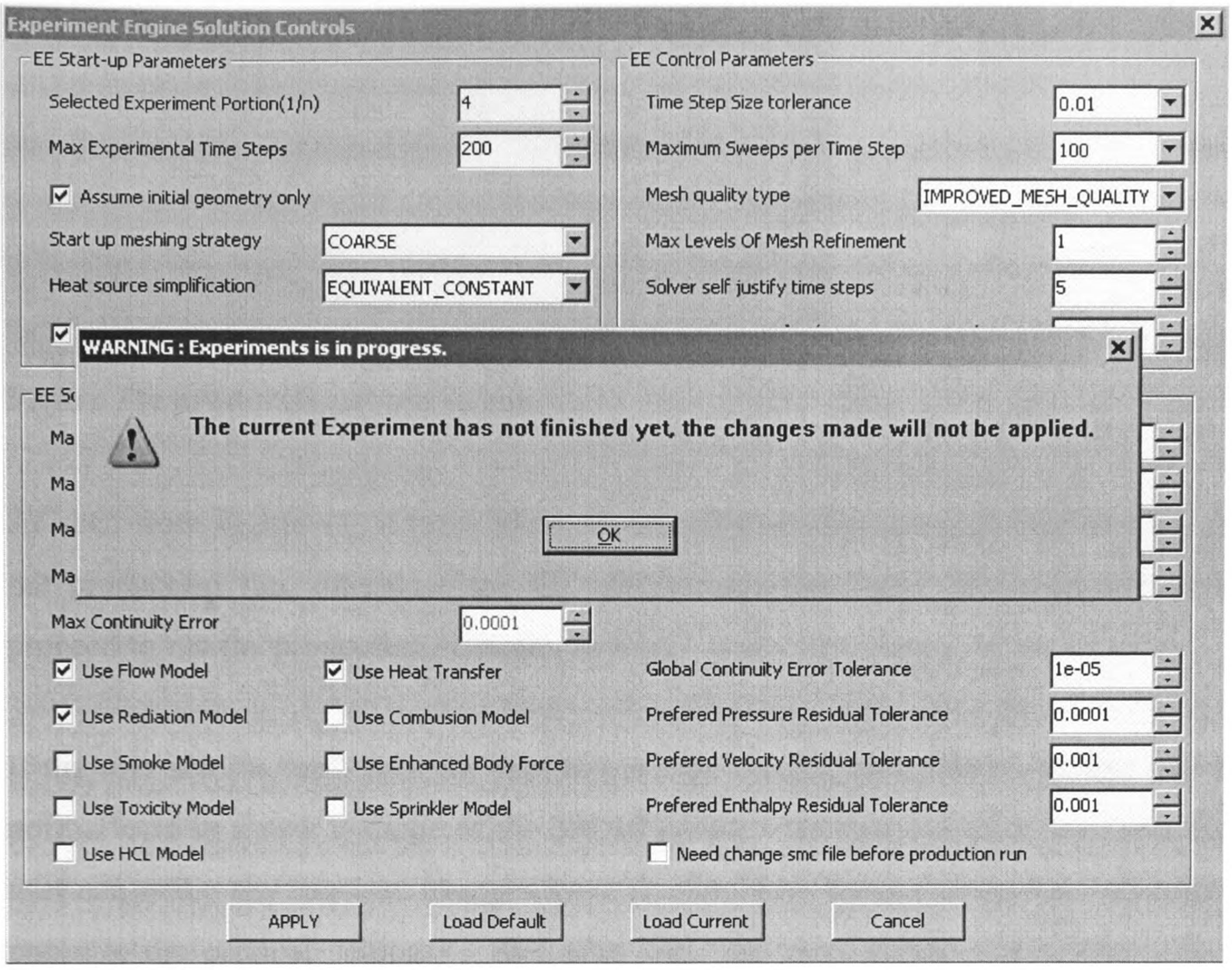
x: 5.0000

y: 5.1000

z: 5.0000

Figure 9: Enabled Run Production option

When the **Enable Advanced Control of Experiment Engine** option is selected, it also enables the run time changes of the Experiment Engine Solution Control parameters, while the experiment engine is running. Otherwise, it is not allowed to make any change on the EE parameters, once the Experiment engine is running (see Figure 10).


Experiment Engine Solution Controls

EE Start-up Parameters

Selected Experiment Portion(1/n): 4
Max Experimental Time Steps: 200
☑ Assume initial geometry only
Start up meshing strategy: COARSE
Heat source simplification: EQUIVALENT_CONSTANT

EE Control Parameters

Time Step Size torlerance: 0.01
Maximum Sweeps per Time Step: 100
Mesh quality type: IMPROVED_MESH_QUALITY
Max Levels Of Mesh Refinement: 1
Solver self justify time steps: 5

WARNING : Experiments is in progress.

The current Experiment has not finished yet, the changes made will not be applied.

OK

Max Continuity Error: 0.0001
☑ Use Flow Model
☑ Use Heat Transfer
☑ Use Rediation Model
☐ Use Combusion Model
☐ Use Smoke Model
☐ Use Enhanced Body Force
☐ Use Toxicity Model
☐ Use Sprinkler Model
☐ Use HCL Model

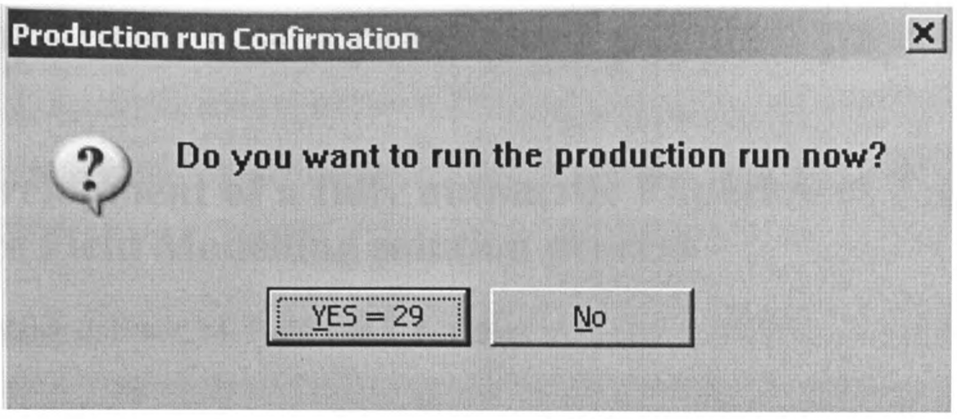
Global Continuity Error Tolerance: 1e-05
Prefered Pressure Residual Tolerance: 0.0001
Prefered Velocity Residual Tolerance: 0.001
Prefered Enthalpy Residual Tolerance: 0.001
☐ Need change smc file before production run

APPLY | Load Default | Load Current | Cancel

Figure 10: changes of solution controls at run time

It should be noted that there is no way to stop an already running Experiment Engine operation without exiting from the current case. However, there is no harm that user selects [file] => [open] and then load the current case again, because, as mentioned before, the Experiment Engine preserves the original settings absolutely and it also saves all of the control configurations automatically. So it is easy to use this approach to stop the unwanted running of the EE.

The final point is that when the Experiment Engine makes transition from the experimental phase to the production run, it will give the user a chance to make last minute manually amendments – if required.

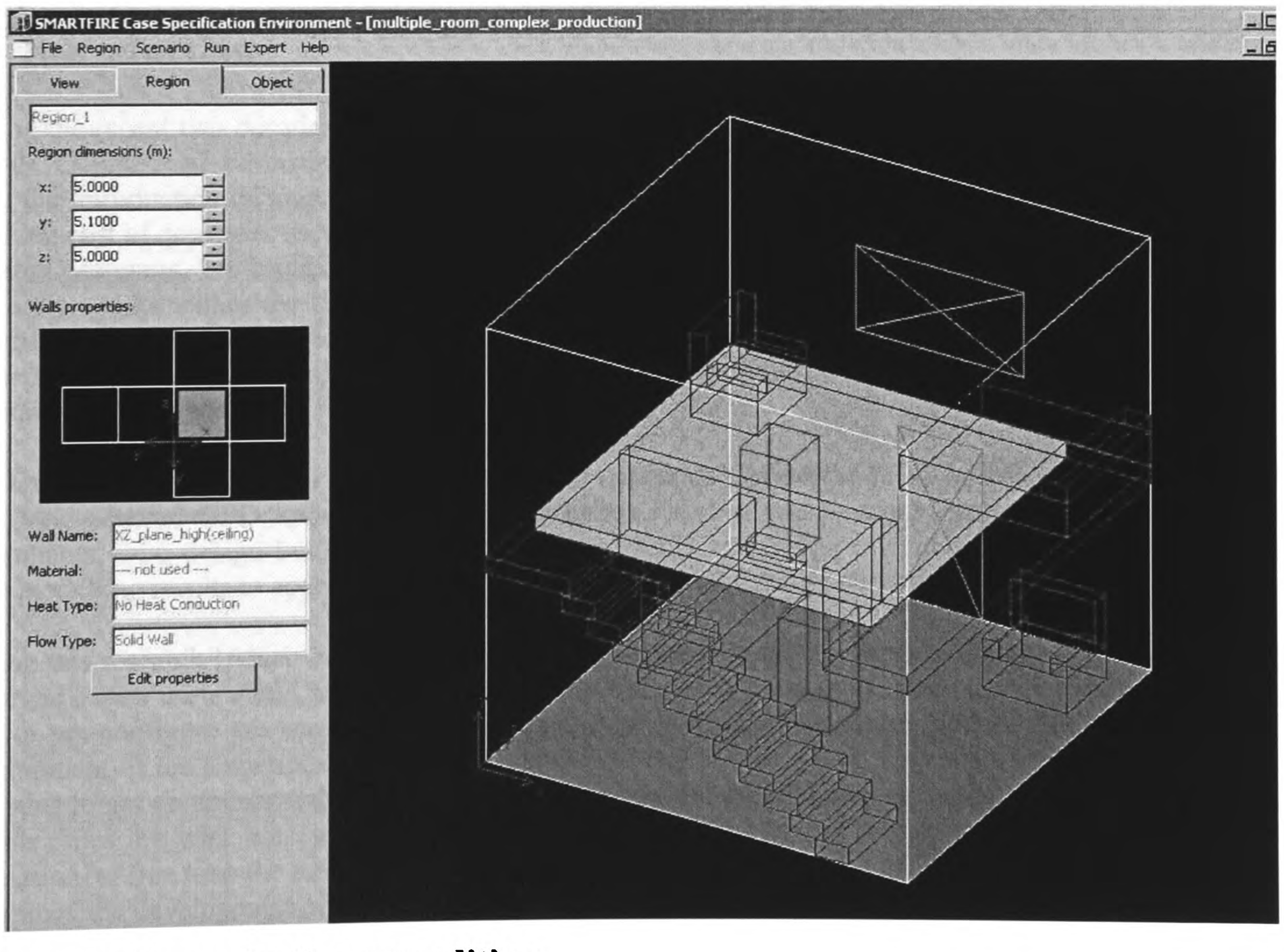

Production run Confirmation

Do you want to run the production run now?

Figure 11: production about to run

As seen from Figure 11, it gives users 30 second to decline immediately running of the production run, otherwise the EE will assume that "yes" is the answer, and proceed to run the production scenario, as it is.

If the user selects "no", then the production case specification will be opened like a normal case in a new instance of the ***SMARTFIRE*** CSE (see Figure 12), so that the user can make any required changes there (It should be noted that the EE will again preserve the original settings). And after that, the users should enable the expert control of the Experiment Engine and run the production directly.


SMARTFIRE Case Specification Environment - [multiple_room_complex_production]

Region_1

Region dimensions (m):

x: 5.0000
y: 5.1000
z: 5.0000

Walls properties:

Wall Name: XZ_plane_high(ceiling)
Material: --- not used ---
Heat Type: No Heat Conduction
Flow Type: Solid Wall

Figure 12: production run editing

Appendix C – a proposed publish paper

Development of a fully automatic Experiment Engine for the *SMARTFIRE* Fire Field Modelling solution process

Y. Wang, J. Ewer, M. Patel and E.R. Galea

Abstract

Fire Safety Engineers, Building Designers and Regulators need to feel confident in the predicted results of using CFD fire field modelling techniques before the results are used for fire safety design purposes. The correct and appropriate approach to problem set-up, mesh discretization, and the choice of solution control parameters, can greatly influence solution stability and the quality and nature of the results. In order to address these issues, a new framework has been developed (called the Experiment Engine) for solution process control of the *SMARTFIRE* Fire Field Modelling (FFM) Environment.

It is demonstrated that the Experiment Engine can enhance simulation stability, makes the solution process operate more optimally and provides reparative intervention if a solution failure is detected.

The Experiment Engine (EE) is fully automatic and has demonstrated considerable robustness on a range of fire field modelling scenarios. The EE combines a number of techniques from earlier research developments and provides a framework for coupling the solution control and model specification support techniques. The EE integrates and makes use of existing research including case classification and recognition, adaptive mesh refinement and solution status assessment methods (as used by an earlier solution control system - called the ICS). The EE offers significant enhancement to the FFM simulation, in terms of providing simulation stability and a reliable outcome and also demonstrates a higher rate of successful and converged fire simulation than would be achieved by using statically prescribed solution control before the simulation started.

1. Introduction

During the recent two decades, huge progress has been made in the research of Fire Field Modelling (FFM) [1]. The development of sophisticated fire models (e.g. radiation models, smoke models and combustion models etc.), the introduction of more efficient solution algorithms, together with increasing processing power and the reducing cost of computer hardware, have led to the widespread use of FFM techniques by fire safety engineers, building designers, fire brigades, and building regulators. FFM is now widely used for research, development and design tasks within the fire safety and building industry. CFD based FFM codes offer powerful tools for fire-safety analysis and research of building designs. Their use still requires a high level of skill and understanding of the operation of the software, in order to obtain accurate and meaningful results in these generally complex scenarios [2].

The Experiment Engine (EE) has been developed in order to support the FFM software users, especially those who lack detailed CFD knowledge, but nevertheless have to use or make decisions based on FFM (e.g. Consultants, who design buildings may have limited CFD knowledge or regulators, who are presented with CFD modelling results to approve designs but do not know if the model was run appropriately).

During this research (which has used expert CFD/FFM users to specify various control scenarios) it was noted that even expert users could not always successfully run a simulation in the first attempt and were almost never able to pre-configure the scenario to run optimally for the entire simulation period. The premise behind the development of the Experiment Engine (EE) is that an expert user would typically use a number of simplified trial runs to get an understanding-of and gain insight into the specific case to be modelled. The knowledge and outputs from the trial runs would then be used to more appropriately set up the production version of the simulation, to fine tune the meshing and hence to obtain the required solution. This is the key strategy which has influenced the development of the EE.

The EE attempts to emulate an expert user's ability to be able to set up and take appropriate control actions as and when necessary, and to recover from any simulation failure. This process involves both an experimental and a production phase. In the experimental phase, the original problem is simplified in order to minimize the overheads of performing experiments. Once the experiments are complete, the mesh is adaptively refined (based on solution error norms) and control parameters are changed when the solution has not converged, until the results are found to converge to a prescribed accuracy. The newly established set-up, meshing and control parameters can then be applied to the original scenario specification in order run the full production simulation.

This paper first summarizes the previous work has directed and influenced this work. This is followed by a discussion of the overall framework of the process. The paper then details the sub-processes and other essential building blocks, such as (1) the case classification and recognition techniques, (2) the mesh justification algorithm used to tackle problems of poor cell aspect ratios, (3) the adaptive mesh refinement techniques and (4) the solution status assessment techniques developed within the Intelligent Control System [3]. Finally this paper presents a number of examples to demonstrate the benefits of using the EE system. There is also a discussion of areas for future work. The Experiment Engine is implemented in the *SMARTFIRE* Environment [4-9]. However, these techniques could easily be applied to other CFD software and other CFD application areas other than FFM.

2. Previous Work

A review of literature relating to the methods discussed in this paper has yielded many works regarding adaptive methods, error estimates and, more recently, the development of Intelligent Controls system (ICS) to improve solution processing. However these techniques typically concentrate on only a small part of the solution process and do not take the solution process as a whole, in order to offer complete support to users.

2.1 Adaptive Meshing Techniques

Adaptive meshing techniques [10, 11, 12] are a well recognized solution to the problem of obtaining suitable mesh quality. These methods have been studied for quite some time and common procedures that have been studied include:

- Local refinement and/or coarsening of a mesh (h-refinement).
- Relocating or moving a mesh (r-refinement), and
- Locally varying the polynomial degree of the basis (p-refinement)

These strategies have been used singly or in combination. R-refinement alone is generally not capable of obtaining a solution with a specified accuracy. If the mesh is too coarse, it might be impossible to achieve a high degree of precision without adding more mesh cells or altering the basis. P-refinement is the most powerful. Exponential convergence rates are possible when the meshing is sufficiently smooth. H-refinement is by far the most popular and widely-employed technique. H-refinement can also increase the convergence rate. These adaptive procedures have mainly been applied to Finite Element Methods, while *SMARTFIRE* uses a Control Volume (CV) formulation. Both Finite Element Analysis (FEA) and Control Volume simulations require high quality mesh generation. This is particularly true for FFM, where extremes of temperature, pressure and flow rates are common. The mesh refinement techniques for FEA are completely applicable for CV formulation in the current context. Usually posterior error estimates [13] are necessary to terminate an adaptive procedure. However, it is rare to find optimal strategies for deciding where and how to refine or move a mesh or to change the basis [19].

Recently Tristano[20] et al. presented a framework for a automatic solution process integrated with adaptive mesh refinement techniques, to obtain FEA results with a desired accuracy. This work concentrates more on the refinement of tetrahedral meshes than in attempting to create a robust process that runs on real world simulations and offers complete support to users.

2.2 Other Simulation Optimization Techniques

Janes [14] developed an Intelligent Control System (ICS) which demonstrated some ability to automatically adjust the solution control parameters in order to ensure the stability and convergence of FFM simulations. The ICS technique left meshing issues completely out of control process. The assumption, for the ICS research, was that the mesh had already been generated to a sufficiently high quality so that the simulation would be able to reach a solution simply by making appropriate changes to control parameters during the simulation. This often required considerable expertise to ensure that a sufficiently high quality mesh was first generated and there were likely to be complex scenarios where even a high quality mesh would not be adequate for the demands of the simulation.

This earlier research investigated how the time step size and the outer-loop iteration limit had significant influence on the rate of convergence and numerical accuracy. It is generally accepted that using large time steps can save CPU time for transient cases, but this generally comes at the expense of numerical accuracy. To some extent, this can be mitigated by increasing the outer-loop iteration limit. However, a larger outer-iteration limit will result in a longer CPU time if the limit is frequently reached during processing. There is a trade off between time step size and outer-loop iteration limit, in terms of rate of convergence and numerical accuracy. Liu [21] et al. conducted a study on the influence of the time step size and outer-loop iteration limit for the integration of Reynolds Averaged Navier-Stokes equations, to improve the accuracy and efficiency of the numerical simulation of transient flow phenomena. They concluded that, in the view of convergence, a small time step size is always favourable to a large one. A large time step size, with more outer-loop iterations, is not recommended because it tends to give poor convergence performance. Smaller time step sizes, with fewer outer-loop iterations, are recommended. This view was also supported by Janes's ICS research.

3. Process Description

An overview of the iterative EE solution process is illustrated in Figure 1.

The EE framework in *SMARTFIRE* links the Case Specification Environment (CSE) and the Interactive CFD Engine components. These components now communicate via data passed through the well defined Windows message interface. In this way, it is possible to maintain the CSE process and the CFD Solver process independently. The two component processes then interact with one other when the EE is operating. This componentization has also enabled the rapid prototyping of new developmental ideas.

The EE architecture and mode of operation, have been carefully planned to avoid complete failure of the CFD process (See Figure 1). Various techniques are used to protect the iterative experiments from producing poorly defined problem set-ups. Since every experiment has an associated processing "cost", all of the building blocks of the EE have undergone considerable optimisation and are configured so as to minimize these costs. The EE has to provide high quality feed back to the CSE in order to ensure that an appropriate and correct problem specification is generated for the production run, so that, ultimately the correct solution will be achieved with adequate quality.

The EE controlled solution process is described as follows: First, a default test case is formed as a result of case recognition and from applying simplification strategies (e.g. Simplification of any complex Heat Release Rate curve, Modification of the simulation period and time stepping, etc.) that are intended to give an adequate representation of the stability of the final case without all of the inherent complexity or the need to simulate a long period to reach the stability challenges.

The default test case is passed to the CFD Engine for solution. At the same time, the CSE and the CFD Engine establish a connection. The control parameters for experimentation (i.e. the predefined solution error tolerance, convergence tolerances etc.) are wrapped as messages and passed to the CFD Engine. At this point, the EE has taken full control of running the two components.

As the CFD Engine is running, the solution status and results are constantly monitored by a routine that checks the status at the end of each iterative sweep. In the event of detecting convergence, reaching the configured number of iterations or encountering some fatal error, the current solution status and the results summary are extracted and wrapped as messages that are sent back to the Test Controller (TC) – an EE component that controls the tests. This TC then decides if any action needs be taken because of the message or if it is acceptable to allow the CFD Engine to continue processing.

The TC compares the solution status with the experiment control rules. If the reported error is bigger than the prescribed error tolerance, then the TC will activate a re-meshing of the scenario using the local mesh refinement sub-process. This action will lead to a refinement of the current mesh according to the error distribution that was found on the previous unrefined mesh. Details of the localized mesh refinement sub-process are given later.

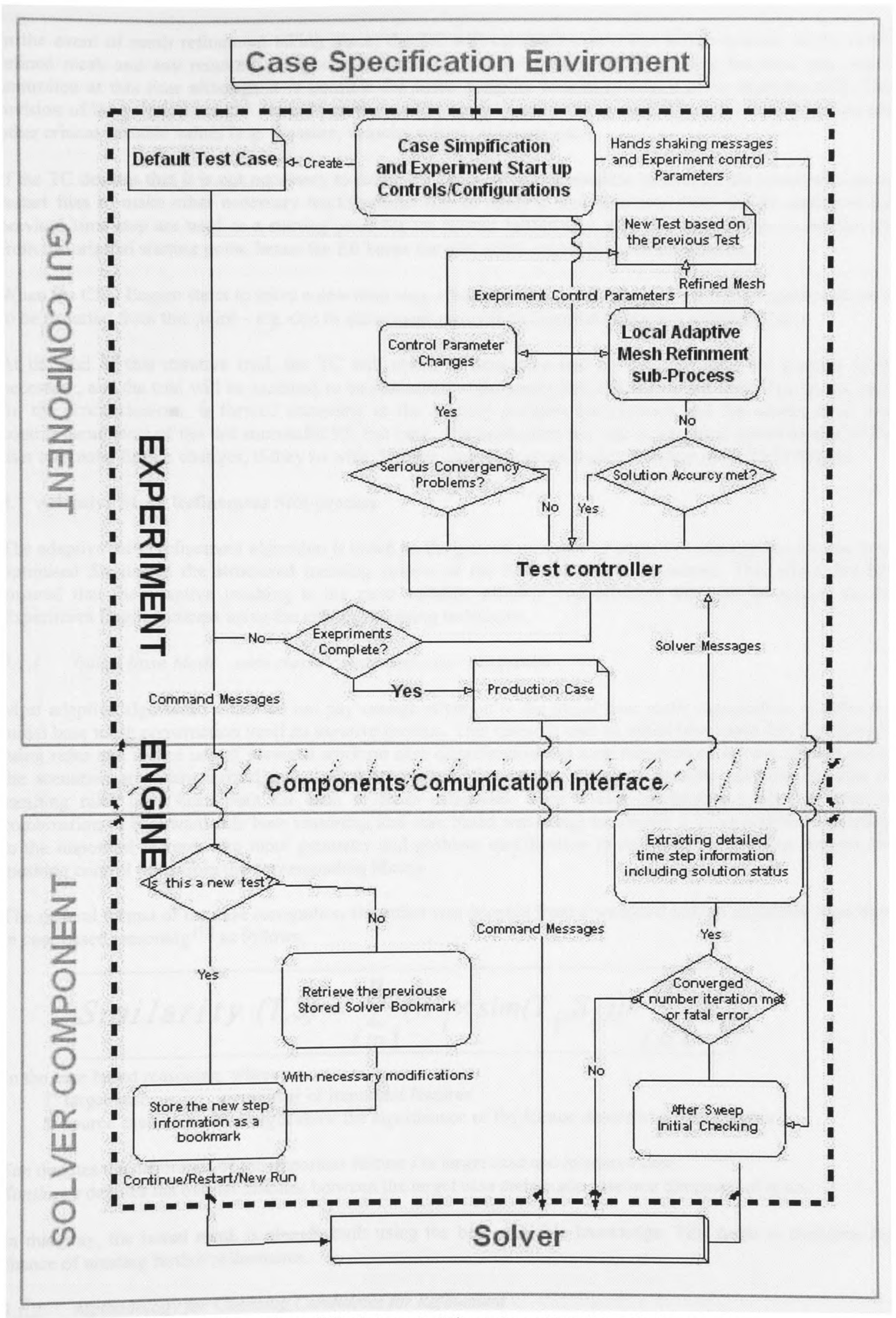

Case Specification Enviroment
GUI COMPONENT
EXPERIMENT ENGINE
SOLVER COMPONENT
Default Test Case
Create
Case Simpification and Experiment Start up Controls/Configurations
Hands shaking messages and Experiment control Parameters
New Test based on the previous Test
Refined Mesh
Experiment Control Parameters
Control Parameter Changes
Local Adaptive Mesh Refinment sub-Process
Yes
No
Serious Convergency Problems?
No
Yes
Solution Accurcy met?
Test controller
No
Exepriments Complete?
Yes
Production Case
Solver Messages
Command Messages
Components Comunication Interface
Extracting detailed time step information including solution status
Is this a new test?
No
Command Messages
Yes
Yes
Retrieve the previouse Stored Solver Bookmark
Converged or number iteration met or fatal error
With necessary modifications
No
Store the new step information as a bookmark
After Sweep Initial Checking
Continue/Restart/New Run
Solver

Figure 1: Overview of the integrated Experiment Engine solution process

In the event of mesh refinement taking place, the TC will construct a new test which consists of the newly refined mesh and any required changes to the solution control parameters (N.B. only the time step size is controlled at this time although it is possible for more complex control strategies to be implemented). This revision of the solution control depends on the quality assessment of the current solver status information and other critical variable values (e.g. pressure, velocity values, residuals, etc.).

If the TC decides that it is not necessary to refine the mesh, then it is possible to retrieve the previously stored restart files to make other necessary modifications (i.e. to apply a new time step size). So the results of the previous time step are used as a starting point for the re-run. In this way, it is not necessary to re-run the test from the original starting point, hence the EE keeps the cost of the experiments to a minimum.

When the CFD Engine starts to solve a new time step, a bookmark is saved. This is used if the experiments need to be restarted from this point – e.g. due to subsequent poor solution performance or a solution failure.

At the end of this iterative trial, the TC will check to determine that no more changes are deemed to be necessary, and the trial will be assumed to be successful at the prescribed end-simulation time. The set-up, used for the production-run, is formed according to the original problem specification and the set-up, mesh and control parameters of the last successful EE test case. The production run can be executed automatically or the user can make further changes, if they so wish. Finally, control is given back to the user in the CFD Engine.

3.1 Adaptive Mesh Refinement Sub-process.

The adaptive mesh refinement algorithm is based on the general principle of adaptive techniques and it has been optimised for use in the structured meshing system of the *SMARTFIRE* Environment. This adaptation has ensured that the adaptive meshing is the most suitable, efficient and effective that can be applied to the Experiment Engine process using the existing meshing techniques.

3.1.1 Initial Base Mesh – case classification and case recognition

Most adaptive algorithms either do not pay enough attention to the initial base mesh construction or make the initial base mesh construction itself an iterative process. This research uses an initial base mesh that is generated using rules and makes use of previous work on case classification and case recognition. In case classification, the scenarios are characterized into typical categories of scenarios. Category specific meshing libraries of meshing rules have been built for each of those categories. Then a case recognition method is used (a combination of backward rule base reasoning and case based reasoning) to classify a given scenario according to the important features like input geometry and problem specification to recognize it, and then retrieve the meshing control rules from the corresponding library.

The general format of the case recognition algorithm was adopted from a weighted nearest neighbour algorithm in case based reasoning [17] as follows,

$$\boxed{Similarity\ (T,S) = \sum_{i=1}^{n} (W_i \times sim(T_i,S_i)) / \sum_{i=1}^{n} W_i}$$

In the case based reasoning, where:

T: target case	*n*: number of important features
S: source case	*W*: weight show the significance of the feature denote as a float number.

Sim denotes the difference of the important feature *i* in target case and in source case.
Similarity denotes the overall distance between the target case and source case in n dimensional space.

In this way, the initial mesh is already built using the best available knowledge. This helps to minimize the chance of needing further refinements.

3.1.2 Methodology for Choosing Candidates for Refinement

Local error estimates are often used as refinement indicators and to produce solutions that satisfy either local or global accuracy specifications. Successful error estimates have been obtained using h-refinement, where the difference between the solutions on different meshes is used to estimate the error. However, this type of error estimation is expensive as it requires the generation of a second solution – on a different mesh. It is better to make use of the solution itself, to derive the estimates of the error in a suitable norm. This can be achieved by using a posterior error estimation. A posterior error estimate provides accuracy appraisals that are necessary to terminate the adaptive procedure. A global error estimate, in a particular norm, was computed by summing each nodal error contribution as,

$$E = \frac{\sum_{i=1}^{n} (e_i)^2 \times v_i}{V}$$

Where n is the number of cells in the mesh and E is the global error indicator.
e_i is the local error indicator at element i and v_i denotes the volume of i cell.
V is the domain volume.

The method of determining where mesh adaption is needed, is to use e_i as an enrichment indicator. It is assumed that large errors come from regions where the local error estimate, e_i, is large and this will count even more when the cell volume is large as well, hence, this is where the mesh should be refined. By using the conjunction of the interrelated global error tolerance and local error tolerance, the global mesh quality is assured and, at the same time, only regions that are likely to cause problems are actually refined. This makes the mesh adaption process more efficient and effective.

3.1.3 Refinement Method

A common strategy for refinement of an element of a structured quadrilateral-element mesh is achieved by bisection which requires mesh lines running to the boundaries to retain the four-neighbour structure as shown in Figure 2(a).

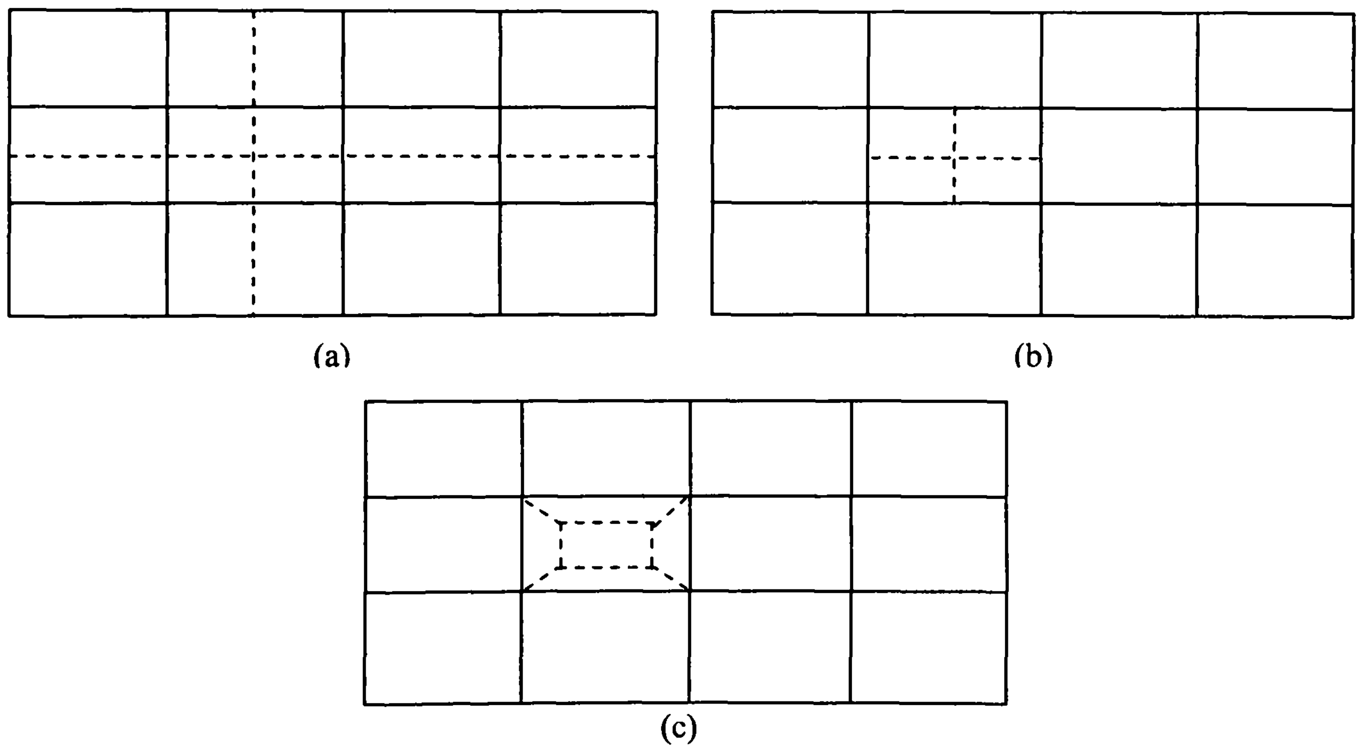


Figure 2: Bisection methods for a structured mesh cell (a) creating a mesh line running between the boundaries, (b) mesh lines are removed by creating irregular cell nodes, or (c), a fully unstructured (hypercube) refinement.

However, this clearly refines many more elements than necessary. The customary way of avoiding the excess refinement is to introduce irregular nodes where the edges of a refined element meet at the mid-sides of a coarser one, as shown in Figure 2(b). The mesh is no longer structured and the standard method of basis construction would create discontinuities at the irregular nodes [19]. This technique would be suitable for FEA and could also be used by exploiting the unstructured nature of *SMARTFIRE*. It is also possible to use a fully unstructured refinement using a hypercube type refinement, as shown in Figure 2(c), although this is likely to introduce a number of accuracy and stability issues due to the shape of some of the refined sub-cells.

To find a simple and efficient refinement method for the existing *SMARTFIRE* automated meshing system, a suitable refinement method was developed which could be directly incorporated into the existing meshing strategy. Essentially, the *SMARTFIRE* meshing system transforms the input geometry specification into a knowledge based form, and then uses a combination of meshing rules (appropriate to similar classes of fire simulation case) and parameters from the meshing library to generate a multi-block structured mesh. Expertise, embedded in the meshing control library, determines the importance of certain qualitatively different block regions in the geometry, and how to divide the geometry into these blocks. Individual meshing parameters are determined for each block. Figure 3 shows an example of the *SMARTFIRE* meshing system generating a mesh for a whole building.

The existing meshing strategy has proven to be well behaved over many years of usage by *SMARTFIRE* users. It was vital that any local refinement method would ensure that all of the best features of the meshing were not jeopardised while taking advantage of the new localized refinement techniques.

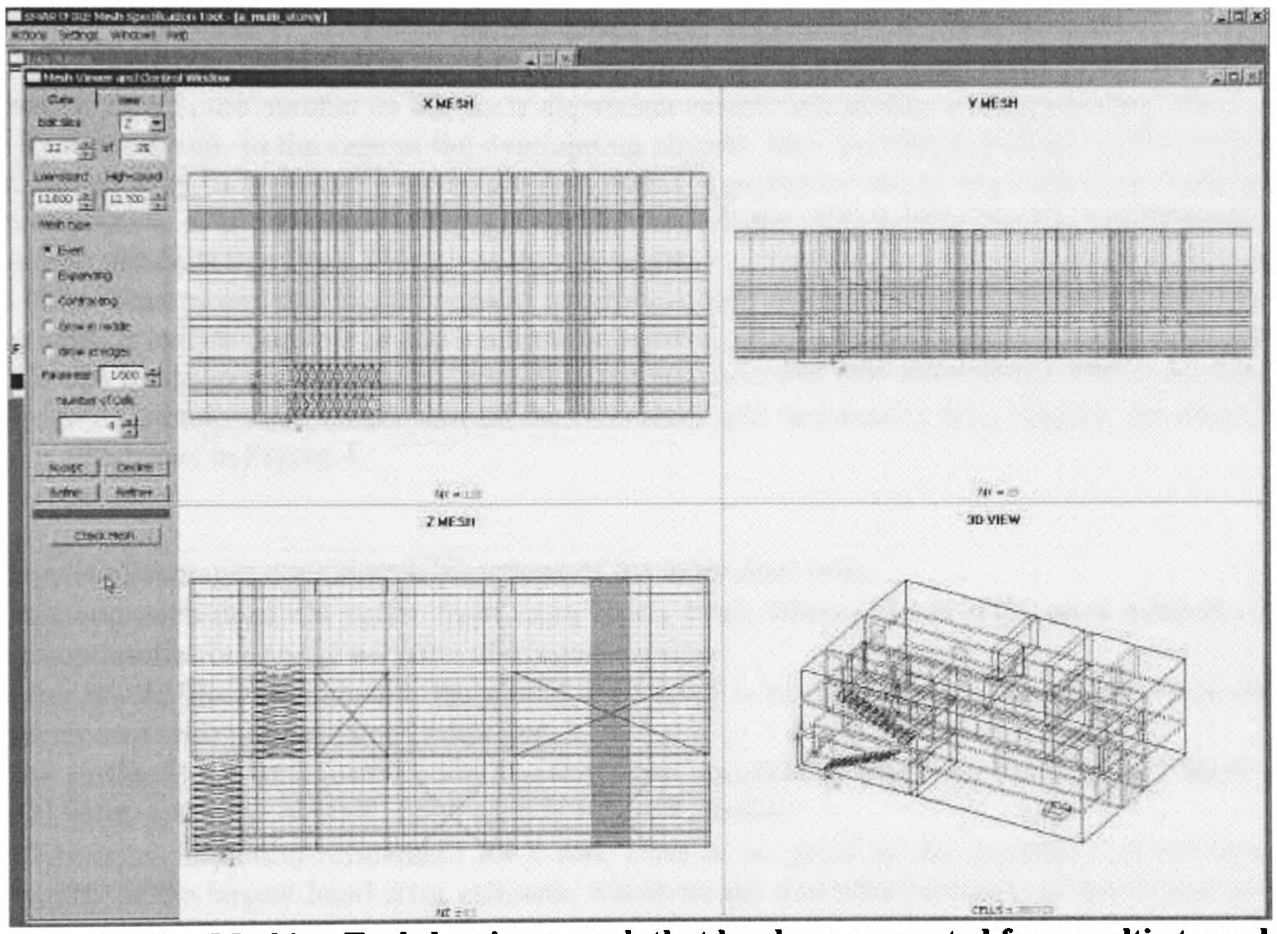
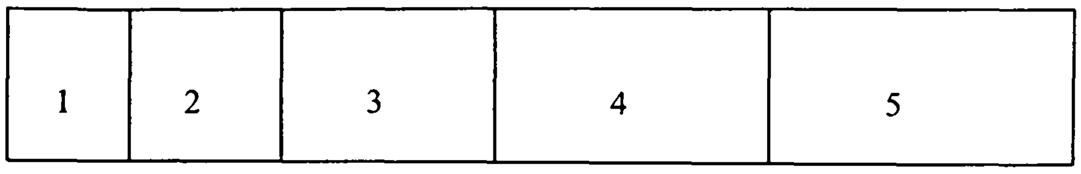
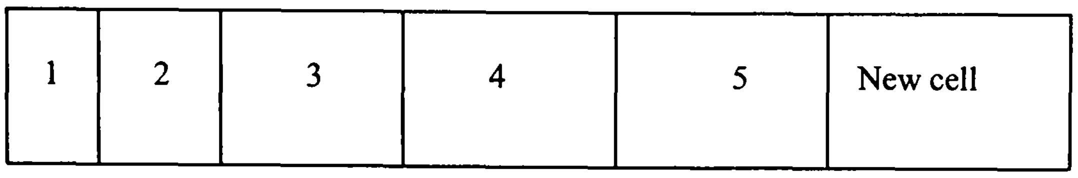


Figure 3: Interactive Meshing Tool showing a mesh that has been generated for a multi-storey building simulation case.



(a) Before refinement



(b) After refinement

Figure 4: Refinement of a block with expanding distribution type

As seen from Figure 3, the number of blocks is dependant on the complexity of the geometry. Each object will occupy at least one block. In the case of the overlapping objects, each overlapping object will occupy more than one block. In this way, it is possible to ensure that within a particular block, the cells have roughly the same physical properties (i.e. if the block is the fire block, then all the cells within the block will have a meshing strategy suitable for cells in a fire). This gives the opportunity to treat blocks as units rather than having to refine individual cells. This means that the refinement can concentrate on the blocks which have a greater percentage of problem cells. It is then possible to add a minimum number of cells (usually one cell at a time is sufficient for most cases), and hence to concentrate on providing the best possible cell distribution within the chosen block. This change of distribution will ensure that all the cells sizes will decrease to some degree. An example of such refinement is illustrated in Figure 4.

This has several advantages over simple bisections of the individual cells.

- This approach does not suffer from extra repair work where a local refinement method could cause discontinuities due to the addition of irregular nodes.
- After whole block refinement, the global error level is tending downwards and the local continuity is guaranteed with minimum cell additions.
- The earlier block-wise justification algorithm (as described in *Repairing the distorted Mesh* section) is still valid, when the overall aspect ratio is still not satisfied.
- When using bisection refinement for a cell, there is no proof of the optimality of enrichment in the vicinity of the largest local error estimate, which means that after bisection of the largest error cells, it still may be necessary to use bisection of its neighbouring cells as well. Conversely, block-wise refinement takes advantage of the overall multi-blocks structure and makes all cells in the target block smaller in size by adding only the minimum necessary number of cells. Hence the target block is refined with a high degree of confidence of further reducing the global errors.
- This local refinement procedure mainly adopts the h-refinement techniques as used in meshing for FEA. However, by using the refinement method as described, we effectively combine r-refinement as well in this refinement method. Regarding Figure 4 again, the refined mesh is effectively the result of the pre-refined mesh being compressed towards the left (i.e. towards the region that was assumed to have the largest solution variation and hence have the strictest meshing requirements). So the refinement method has actually implemented a h-refinement combined with a "static" version of r-refinement without running into the trouble of altering the Solver.
- The distribution/redistribute work has been put into the heart of the refinement method. For such an approach, it is easy to retain a sufficient degree of smoothness of cell distribution. From an r-refinement point of view, the cell points are not moved independently, but rather each point appears to be coupled at least to its neighbours. Also, grid points are not moved too far or too fast, which could cause oscillations.
- After refinement, the original block structure has been maintained with each block still representing the most critical object physical property.

The disadvantages

- Mesh lines still run to the boundaries to retain the regular Cartesian structure, this inevitably means that refinement has used more cells than strictly necessary.
- All the grid points will be redistributed after the refinement. This will make it difficult to re-use values from the pre-refined mesh for the refined mesh.

3.1.4 Repairing the distorted Mesh

The progressive refinement process will inevitably make the aspect ratios unsatisfied at some stage. The mesh will become overly distorted as illustrated in Figure 5 and Figure 6.

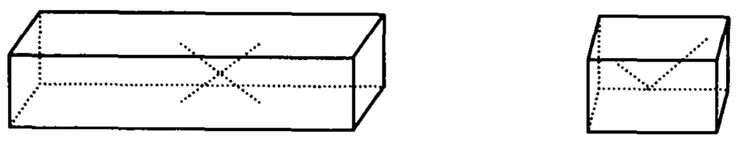


Figure 5: Quality of cells using edge-length ratio metric.

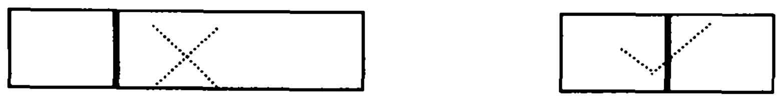


Figure 6: Quality of cells using cell-neighbour-length ratio metric.

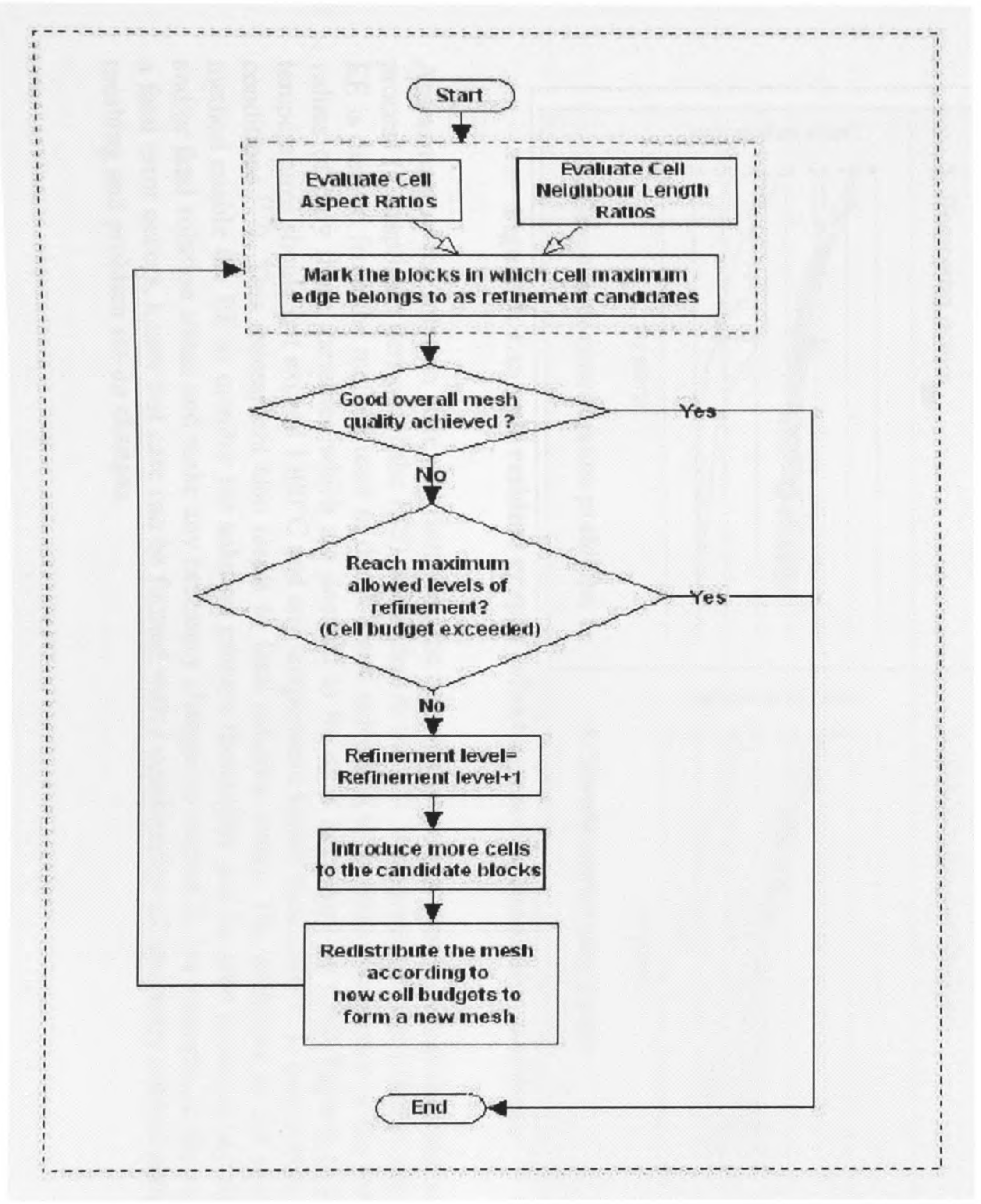

Start
Evaluate Cell Aspect Ratios
Evaluate Cell Neighbour Length Ratios
Mark the blocks in which cell maximum edge belongs to as refinement candidates
Good overall mesh quality achieved ?
Yes
No
Reach maximum allowed levels of refinement? (Cell budget exceeded)
Yes
No
Refinement level= Refinement level+1
Introduce more cells to the candidate blocks
Redistribute the mesh according to new cell budgets to form a new mesh
End

Figure 7: Flowchart of the block-wised mesh justification procedure

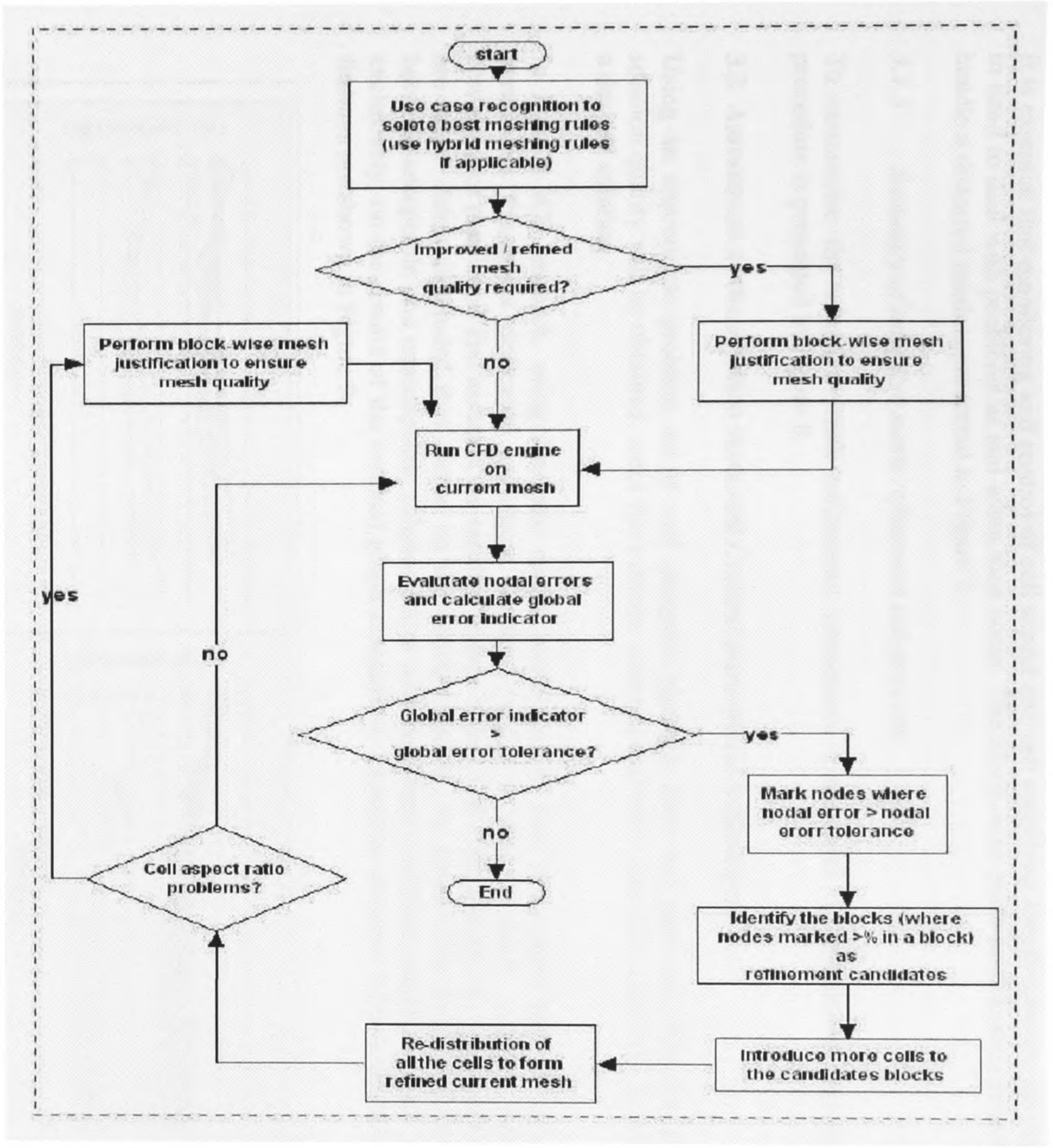

start
Use case recognition to select best meshing rules (use hybrid meshing rules if applicable)
Improved / refined mesh quality required?
yes
no
Perform block-wise mesh justification to ensure mesh quality
Perform block-wise mesh justification to ensure mesh quality
Run CFD engine on current mesh
Evaluate nodal errors and calculate global error indicator
Global error indicator > global error tolerance?
yes
no
End
Mark nodes where nodal error > nodal erorr tolerance
Identify the blocks (where nodes marked >% in a block) as refinement candidates
Introduce more cells to the candidates blocks
Re-distribution of all the cells to form refined current mesh
Cell aspect ratio problems?
yes
no

Figure 8: Flowchart of adaptive local mesh refinement procedure for multi-block structure

It is essential that monitoring and control of cell aspect and cell neighbour length metrics, operate hand in hand to deal with problems as and when they occur. The block-wise mesh justification procedure to handle a distorted mesh is presented in Figure 7.

3.1.5 Summary of adaptive mesh refinement sub-process

To summarise the adaptive mesh refinement procedure, a top-level description of our adaptive procedure is presented in Figure 8.

3.2 Assessment of the solution state and Control parameters adjustment

Using an appropriate problem set-up and adequate meshing does not guarantee that the required solution quality will be obtained, since the solution control parameters also play a key role in obtaining a desired solution.

In Janes's ICS research, using heuristic search techniques to find the almost-best set of control parameters through a dedicated evaluation function (which is able to assess solution state, e.g. convergence problems) that assesses the residual graphs before and after changes of control parameters are made. Janes concluded that, based on the residual graphs, an Expert was able to differentiate between acceptable and unacceptable solution states and the control decisions could be based, almost exclusively, on the results of the residual graph assessment. Examples of some typical residual graph features are shown in Figure 9.

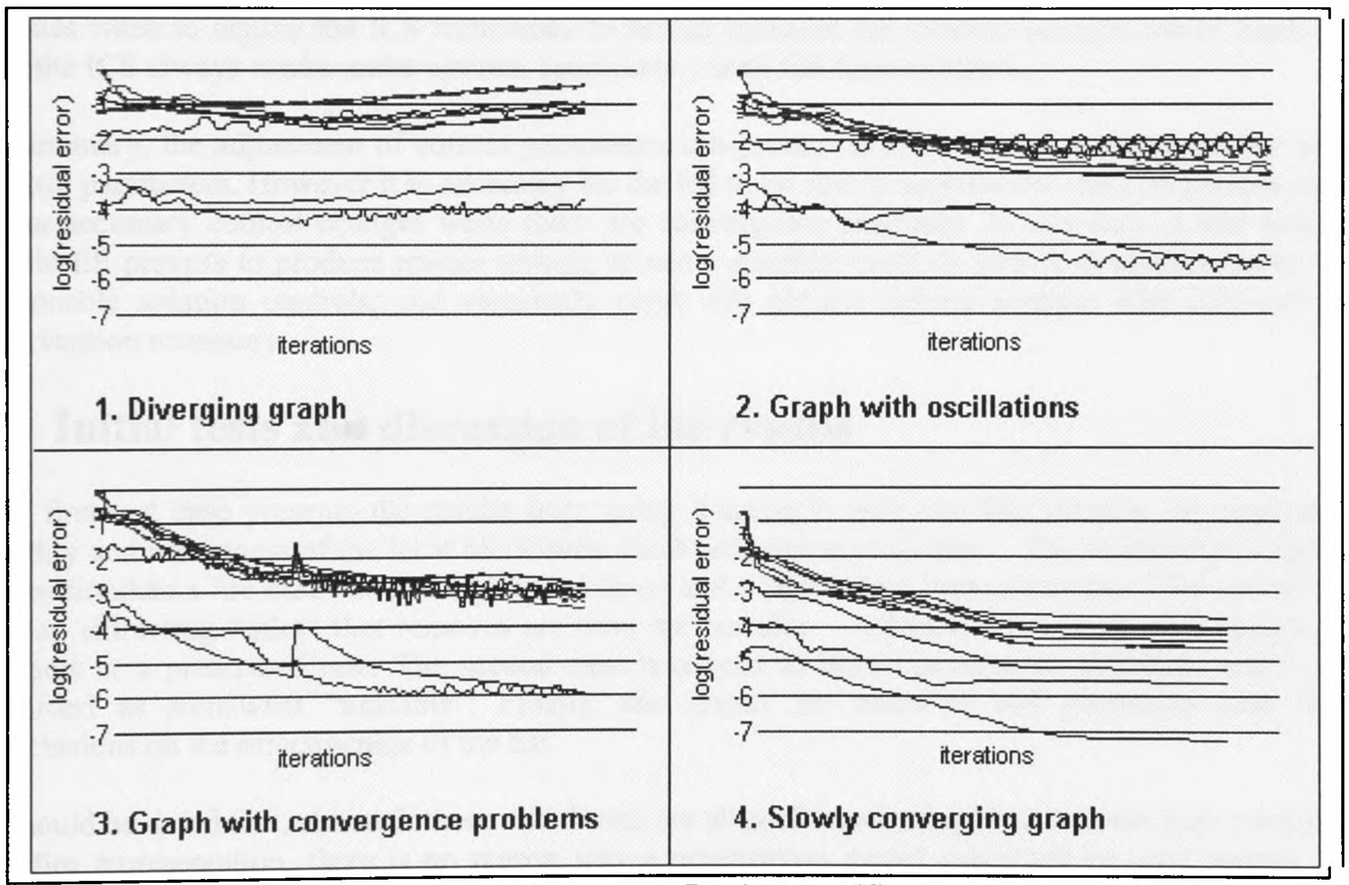


Figure 9: Example residual graphs reflecting specific simulation problems.

Assessment of the solution state and justifying the control parameters has been implemented as a sub-process (a simplified version of the ICS) embedded in the EE. Assessment of the solution state in the EE is derived from the method used in the ICS and extends it to include assessment of the validity of values of only those variables which are deemed to be most important by CFD Experts (e.g. flame temperatures should not exceed 1300°C and any temperature higher than 2000°C is incorrect in normal conditions [17]). This assessment also reacts to fatal solution errors. The additions to the assessment method enable the EE to monitor the solution process thoroughly and be able to detect any abnormal and/or fatal solution status and make any necessary changes to correct it. For example, in the event that a fatal error occurs, a new test case can be formed with a combination of necessary control parameters, meshing and problem set-up changes.

Appropriate control parameters are important outputs from the experimental process. However, unlike the ICS, the changes made on control parameters in the EE, are knowledge based conservative ones and are limited to only the most important parameters which are most likely to improve the solution performance in terms of stability and to make sure that the Solver has a better chance for convergence (i.e. progressively decreasing the time step size when the CFD Engine has convergence problems). This ensures that changing the control parameters is a fast sub-process that will typically have a positive effect, as indicated in the discussion on time step size and outer-loop iteration limits in the previous section.

The cost of experiments means that the EE cannot afford to do a comprehensive search for the best or almost-best set of control parameters, in the way that the ICS attempted. Otherwise, the experiment work would be much more expensive in terms of computational overheads and make it impractical to use.

The purpose of the control parameters change in the EE process is mainly used to maintain a valid solution state rather than trying to find the optimal control parameters. The prime task of the EE is to produce a reasonable set-up, mesh and control parameter selection, as quickly as possible. The EE will then give a much better chance for ICS and/or other optimization techniques (which will also depend on having a high quality mesh) to succeed in the production run than in the experimental phase, due to the fact that there is considerably more confidence in the set-up and mesh produced by the EE, and the control parameters should be suitable after the experiments.

In this way, the EE is not a substitute for the ICS, but rather, the EE is a management tool which decides when to deploy the ICS techniques to further optimise the solution process and to make sure that the ICS always works under optimal conditions – with full fault tolerance.

In summary, the adjustment of control parameters sub-process is not intended to find a perfect set of control parameters. However it is necessary for the EE to be able to monitor the solution process and to make necessary control changes when there are convergence problems. In this way, it will help the whole EE process to produce proper set-ups, to use a suitable mesh, to form a production cases with reasonable solution controls, and eventually users will get the desired solution with minimal user intervention necessary.

4. Initial tests and discussion of the results

The first test case presents the results from using a scenario with two fire sources, to examine the validity and robustness of the local block-wise mesh refinement procedure. The second study uses the EE to simulate a fire case that has high speed flows and a high output heat release rate. The second case has an extracting "inlet" that removes air from the corridor – representing a ventilation system that extracts at a prescribed rate. The second case was seen to have convergence problems and is thus regarded as somewhat "unstable". Finally, the results are analysed and presented with initial conclusions on the effectiveness of the EE.

It should be noted that, although these initial tests are all performed using Heat Release Rate models for the fire representation, there is no reason why a combustion model could not be used instead. The complexity and nature of the algorithms has no impact on the validity of the Experiment Engine monitoring and control regime.

4.1 Validity of the local block-wise refinement algorithm

In order to validate the efficiency and robustness of the local block-wise mesh refinement algorithm, a multiple room geometry (of size 5.0 m by 5.1 m by 5.0 m) was used – as shown in Figure 10. This has two fire sources with the same heat output, both activated at t=0 s. Both fires have growing Heat Release Rate (HRR) curves with peak heat outputs of 500.7 kW as shown in Figure 11. One fire is centrally located on the ground floor, and the second fire is on the first floor. Both floors have obstructions representing furniture such as chairs, a sofa and a bed (these do not combust during the simulation period). There is a single vertical vent (a door of size 1.0 m by 2.0 m height) in the ground floor and a single vertical vent (a window of size 2.0 m by 1.0 m height) in the first floor. The two floors are connected by a staircase.

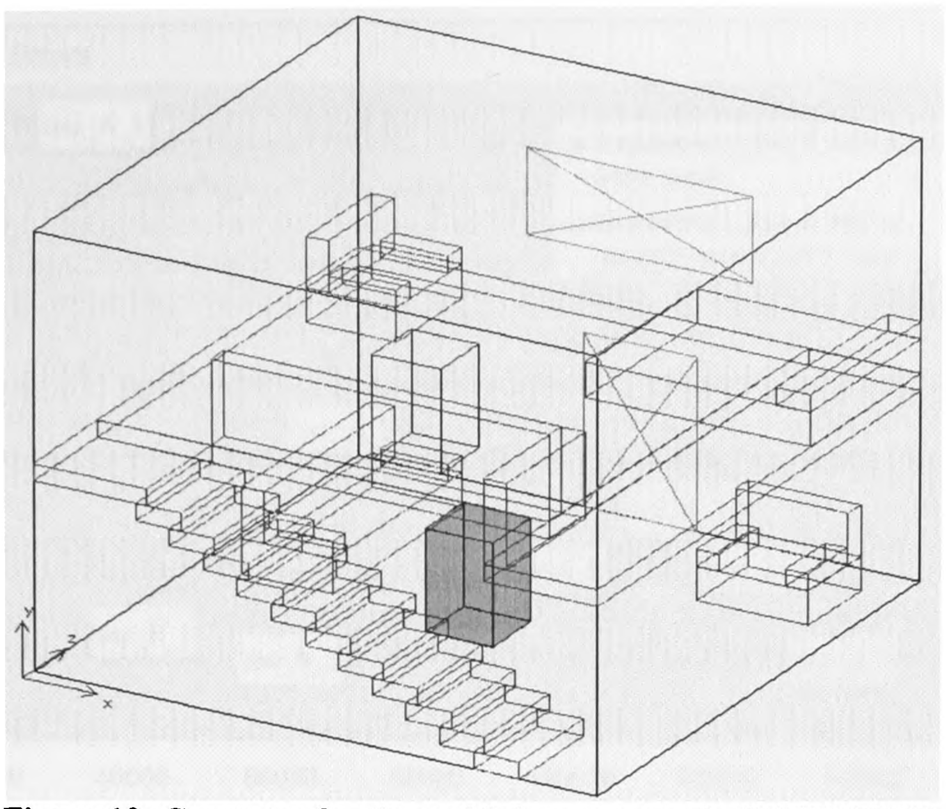


Figure 10: Geometry for the multiple room case in *SMARTFIRE*.

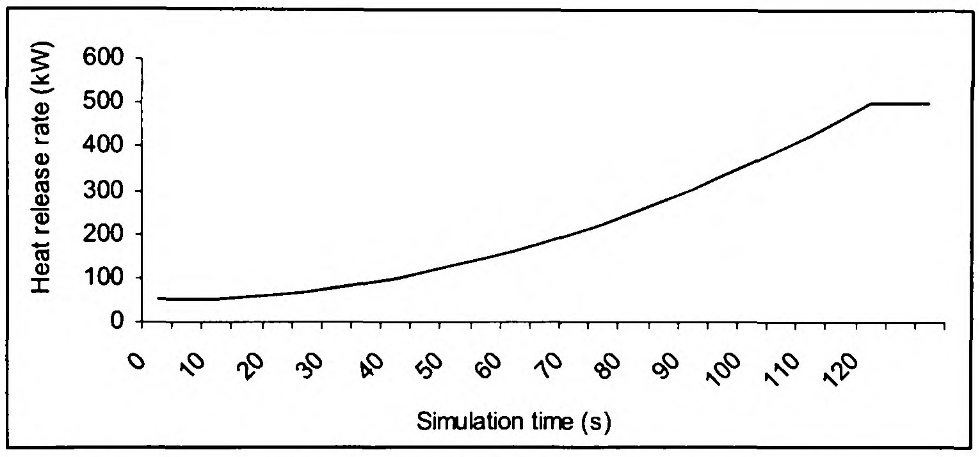


Figure 11: Heat release rate for both of the fire sources in the multiple room case.

Two sets of experiments (set A and set B) have been conducted using the EE with a fixed time step size of 10 s and a maximum of 100 sweeps per time step. Experiment set A started with coarse initial base mesh and experiment set B started with the recommended initial base mesh (i.e. recommendation from the outputs of the case classification and recognition). Each test in the experiment set only runs for a single time step that is simplified to be reasonably representative of the most unstable period of the simulation. New tests are then formed with progressively refined meshes until the maximum nodal error (continuity error) is below a prescribed tolerance of 0.001. The refinement test and continuity error results are shown in Figure 12.

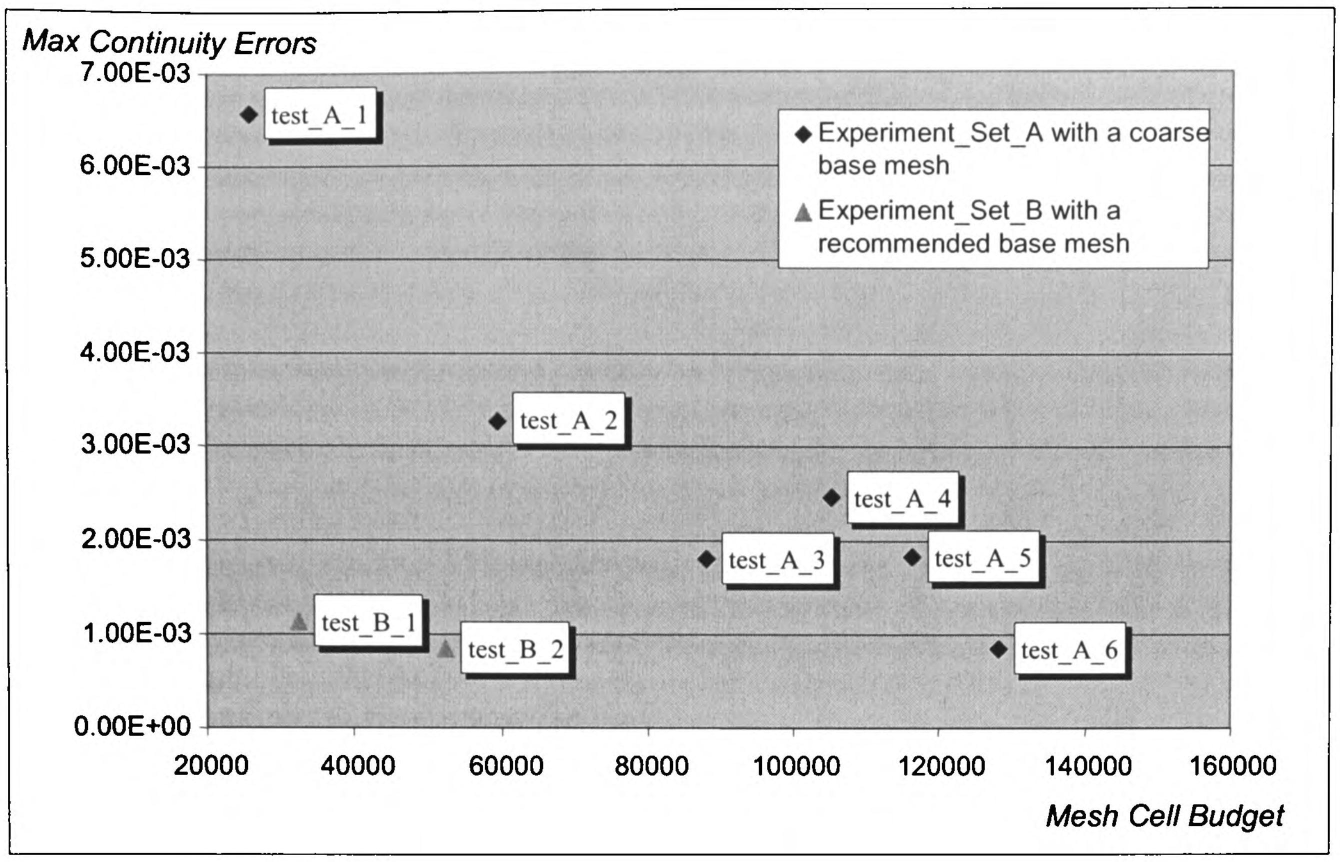


Figure 12: Comparison of the local mesh refinement performance for the multiple room case with different qualities for the initial mesh.

In both sets of experiments, the local refinement procedure performed well in terms of achieving the goal of decreasing the maximum continuity error. The robustness of the local refinement procedure is shown by taking a coarse initial base mesh and still being able to refine it until the prescribed error tolerance is met (A tests). However this approach takes more refinement iterations than using the initial recommended base mesh. It should also be noted that, test A_4 has introduced higher errors than in test_A_3. Conversely, the local refinement procedure performs very well with an initial recommended mesh. It took only two iterations (test B_1 and test B_2) of local refinement before reaching an acceptable mesh which has all nodal errors below the prescribed error tolerance. This demonstrates that the path to the qualified mesh, starting from a high quality initial mesh, is significantly better than when starting from a poor quality initial mesh. This justifies the earlier work on the case classification and case recognition work as important pre-cursors to high quality meshing.

The total cost of experimentation is two time steps for the B tests and 6 time steps for the A tests. The total expense of these extra set-up tests will clearly depend on the total simulated time but this typically runs to tens or hundreds of time steps and so this set-up expense does not add significantly to the processing time.

The solution for the multiple room case proceeds as expected with the hot gasses from the fire developing entrainment in through the lower door and exhausting a plume through the upper floor window. The temperature iso-surfaces at 200, 300, 400 and 500 degrees C (at t=50 s) can be seen in Figure 13.

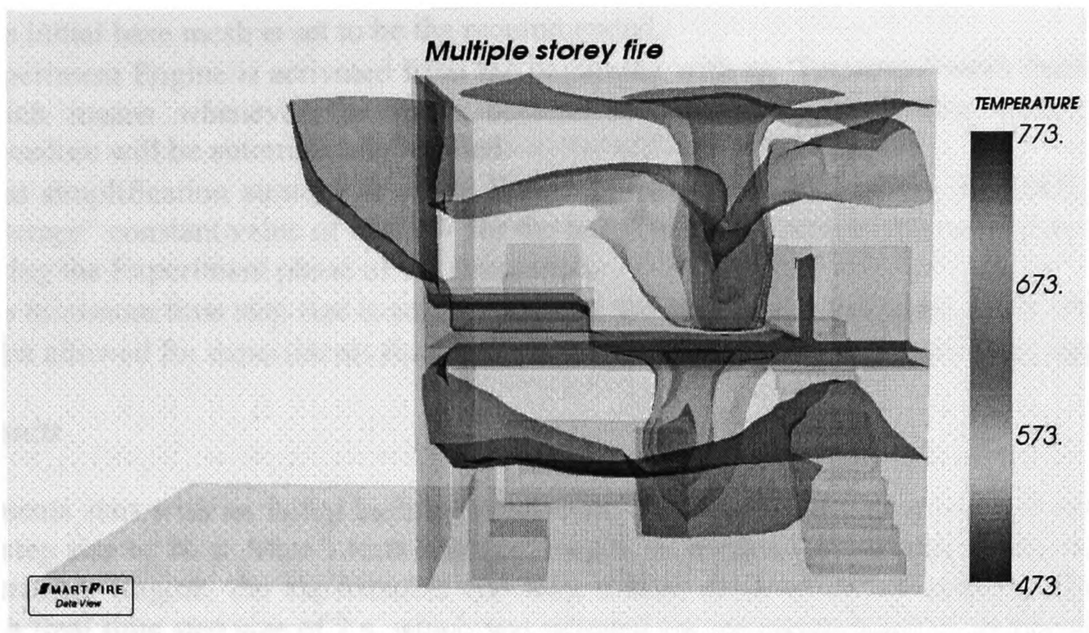


Figure 13: Temperature iso-surfaces for the multiple room fire scenario.

4.2 Corridor and mechanical extraction case

The following example demonstrates the EE supporting a corridor scenario with forced mechanical extraction (an extracting ventilation system) with small vents blowing downwards into all of the rooms. It had previously been observed that this case is problematic to converge using the default solution configuration.

The corridor scenario is a single floor of building with a number of rooms on each side of the corridor. The corridor runs along the full length of the building. Figure 14 shows a wire-frame view of geometry. The dark green (crossed) squares are the duct openings representing the ventilation system. The corridor vent duct has a (extracting) flow rate of 10 m^3/s and the rooms have vents (blowing downwards) at 0.012 m^3/s. The building covers an area of 10.9 m by 9.0 m with a height of 2.4 m. The large volumetric fire has an αt^2 growth rate with α=0.113 and a peak Heat Release Rate (HRR) of 1328kW at 108 seconds of simulation time.

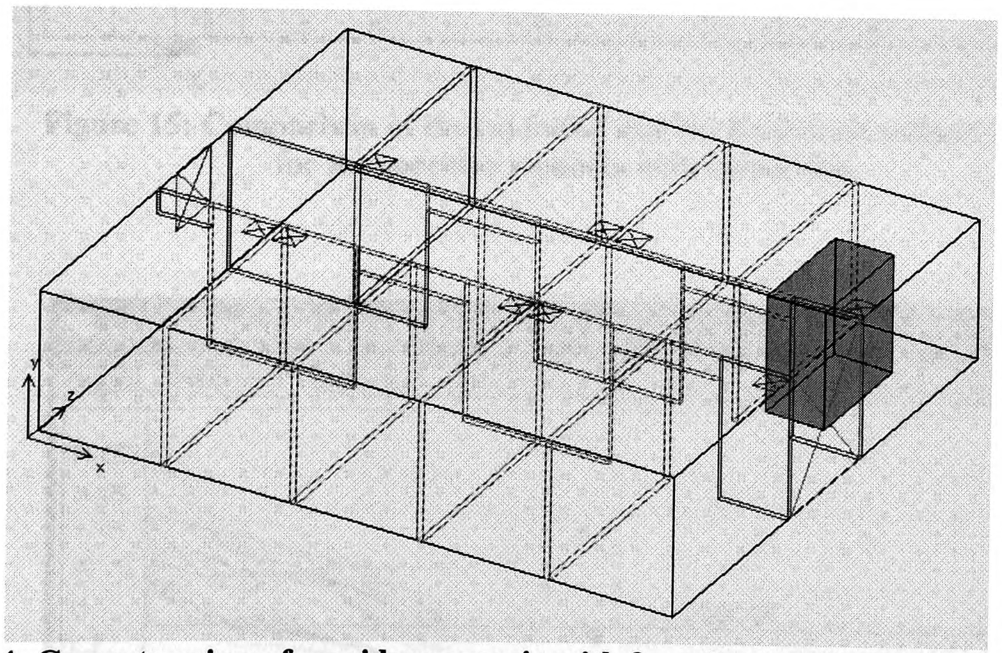


Figure 14: Geometry view of corridor scenario with forced extraction in *SMARTFIRE*.

4.2.1 Model set up

This scenario uses the flow model, heat transfer, radiation and smoke model. For simplicity, the combustion model is not used, since there is no data for the precise nature of the fire source except for a HRR curve.

4.2.2 Experiment Engine Configuration

- The nodal error tolerance is 1e-4.

- The initial base mesh is set to be the recommended.
- Experiment Engine is activated from the beginning with an "Improved mesh quality" strategy which means whenever the mesh became too distorted, block-wise mesh justification procedure will be automatically applied.
- Heat simplification strategy is set to Equivalent Constant with means the HRR is set to an "average" constant value of 450 kW for the test. This is dynamically generated by the EE only during the Experiment phase of the simulation.
- The minimum time step size is set to 0.001 and the maximum sweeps are set to 100.
- Time allowed for experiments time was set at a quarter of the actual simulation time.

4.2.3 Results

The experiments start with an initial base mesh of some 10044 cells (nx=54, ny=6 and nz=31) and an initial time step size of 10 s. After 3 tests and two restarts, in the last test directed by the test controller of the Experiment Engine, the experiments end with a final mesh of 28512 cells (nx=72, ny=9 and nz=44) and a final time step size of 2 s, which was adopted for the production run. It was observed that convergence was achieved in each time step of the production run. The following figures show the residual graph of the end of the production run and a comparison of the initial mesh with the final (most refined) mesh.

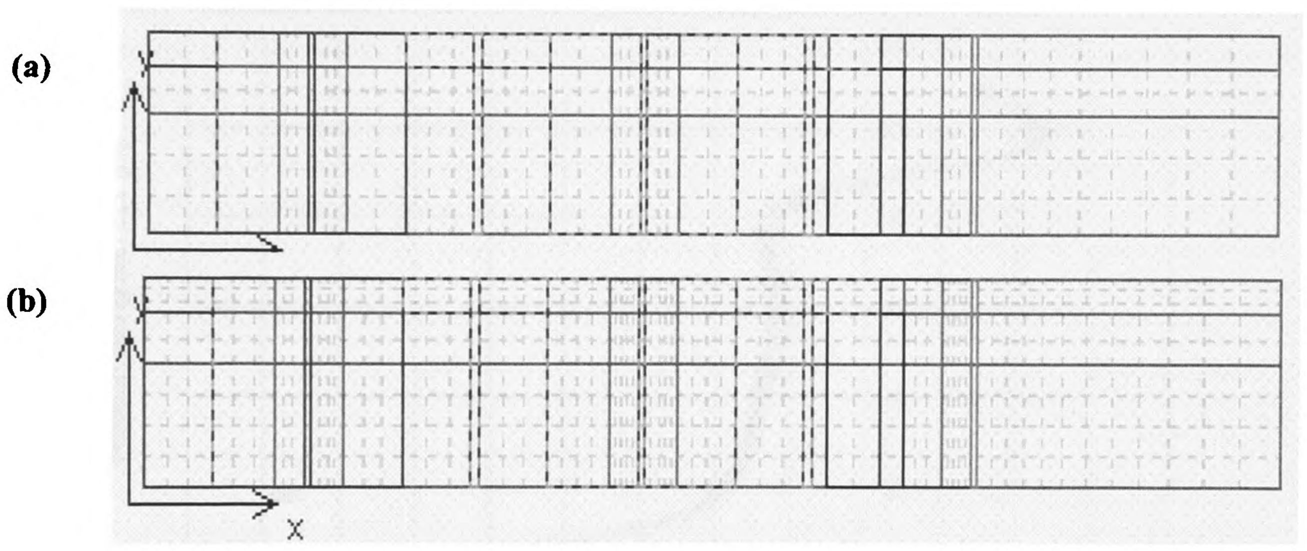


Figure 15: Comparison of the (a) initial and (b) final/most refined y-mesh for the corridor scenario with extraction.

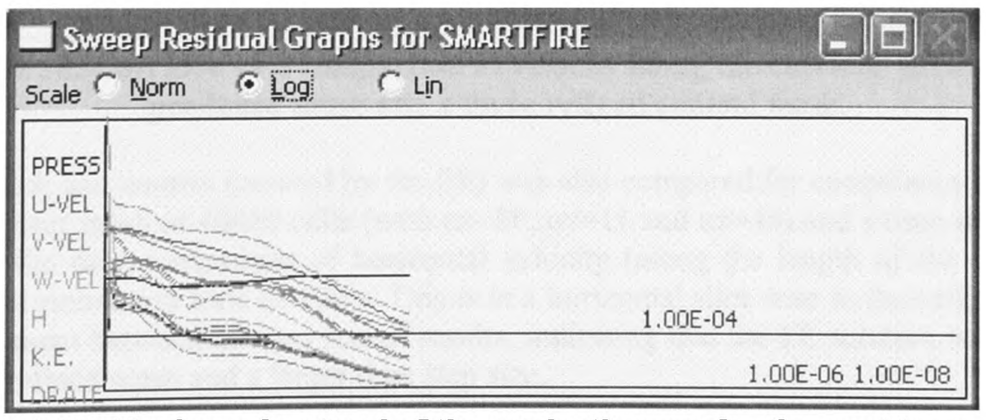


Figure 16: Residual graphs at end of the production run for the corridor scenario with extraction.

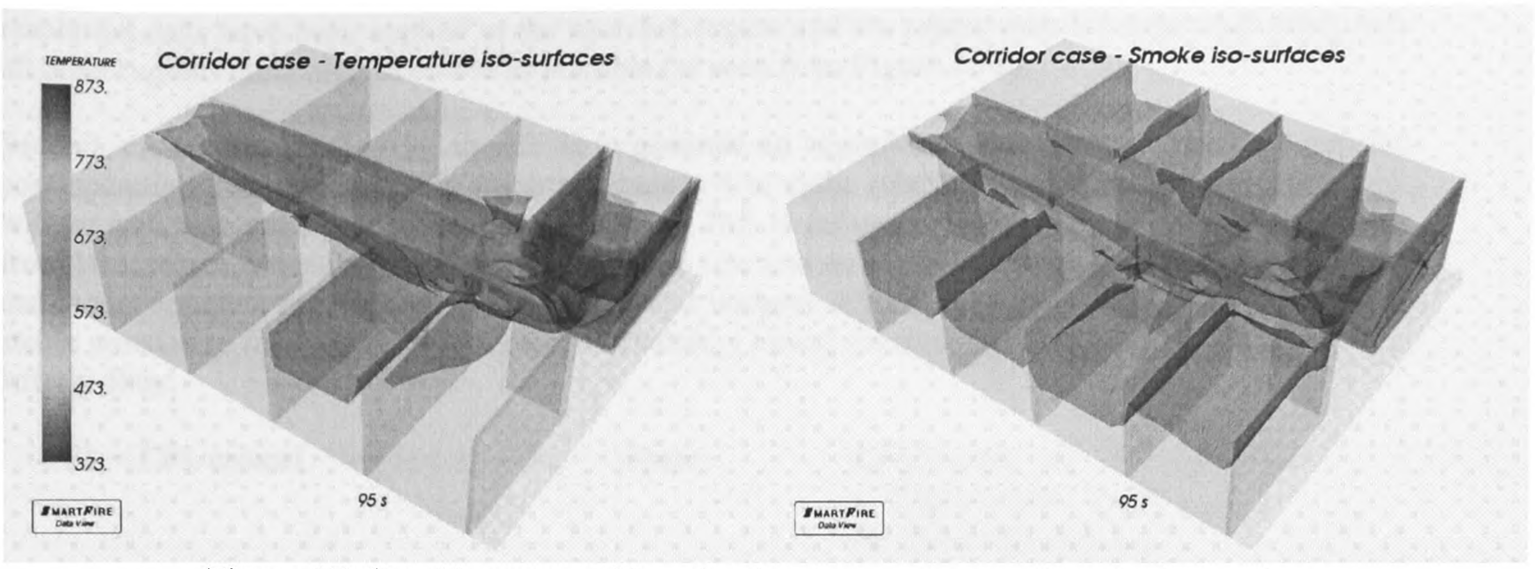

TEMPERATURE
Corridor case - Temperature iso-surfaces
873.
773.
673.
573.
473.
373.
SMARTFIRE
95 s
Corridor case - Smoke iso-surfaces
SMARTFIRE
95 s

Figure 17: Corridor case at 95 s, showing temperature and smoke iso-surfaces.

Figure 17 shows that the simulation proceeds as expected with the corridor extract duct entraining air from the main door and drawing the hot gas and smoke (from the fire room) towards the extract duct. The flow rate for the extraction prevents smoke from spilling out of the main door.

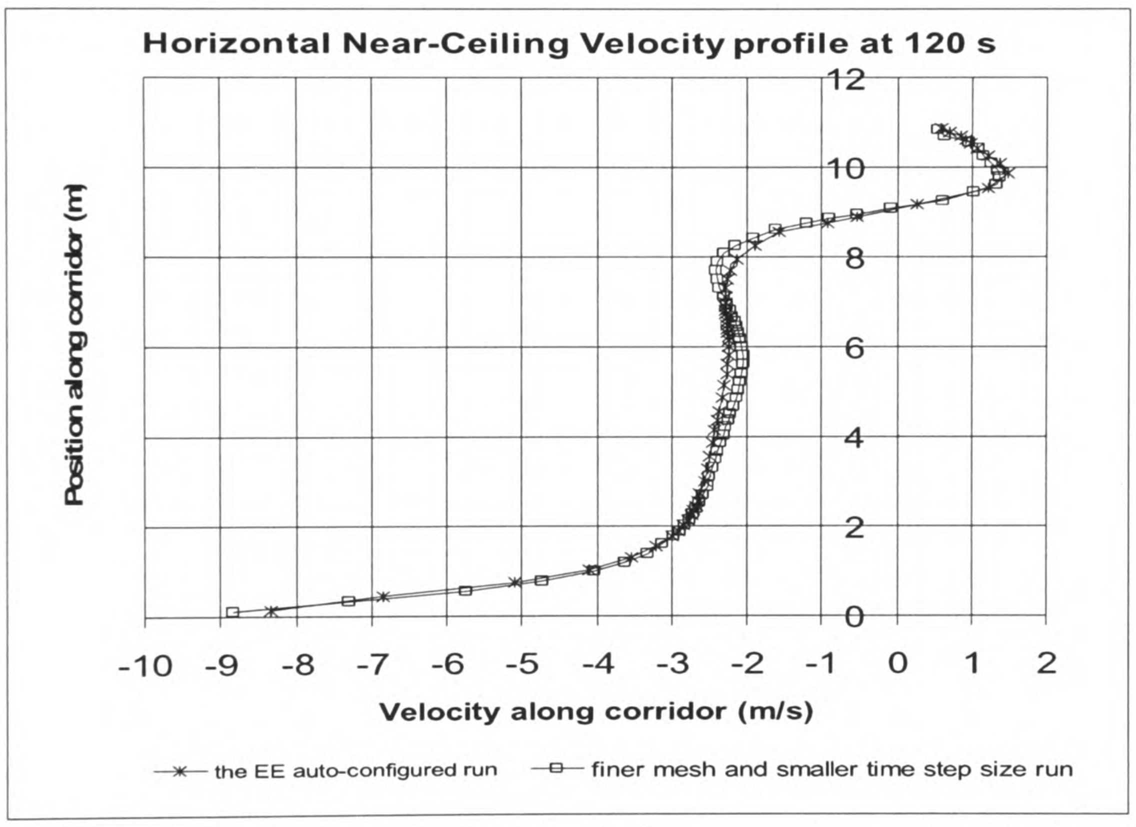


Figure 18: Corridor case comparison of velocity along the corridor between the EE produced mesh and a more refined control mesh.

The simulation mesh and control (created by the EE) was also compared for consistency with a control study that used a finer mesh of 40480 cells (with nx=80, ny=11 and nz=46) and a time step size of 1 s. Figure 18 shows the data comparison of horizontal velocity (along the length of the corridor) with position, at the end simulation time of 120 s. This is in a horizontal slice near to the ceiling. The graph shows good agreement between the two sets of results, indicating that the EE solution has not suffered from using a less refined mesh and a larger time step size.

4.2.4 Discussion of Results

First, it should be noted that it was a surprise that this formerly unstable case was successfully simulated using a comparatively coarse mesh (a final total mesh budget of 28512 cells) and a relatively large time step size of 2 s.

It was observed that the whole solution process was quite stable, since every time step has a well-behaved and converging residual graph. The mesh, produced by the EE, is of good quality since fine resolution cells have been applied to the near fire region and the region near the extraction duct, with all other regions remaining as coarse as possible (as seen from Figure 15-b).

Second, the control strategy for the EE is to generate an appropriate mesh first, and then to find the corresponding time step size. The mesh refinement is to make sure that the maximum continuity error will be well below the predefined error tolerance. This is followed by a time step size search starting from 10 s, which is the almost the largest time step size that an expert user would choose (or be able) to use in any complex fire scenario at reasonable cell budgets. Theoretical study has shown that, to get a stable simulation the largest CFL (Courant Friedrichs Lewy) number anywhere in the flow field must strictly obey:

CFL < CFL critical	Where	$CFL = u\,\Delta t / \Delta L$
		u: Characteristic speed,
		Δt: Time step size, and
		ΔL: Control volume size.

Using Courant limit, it is possible to determine an acceptable ΔL, and corresponding Δt, for the local flow speed.

The experiments were performed on a desktop computer with an Intel Pentium 4 3.0GHz CPU and 2.00GB of RAM. The total CPU time for all of the experiments was 3h 22m 51s and the CPU time for the production run was 5h 24m 08s (i.e. a total combined CPU time of 8h 46m 59s for the EE supported simulation). Hence, the overhead of performing the experiments, for the corridor scenario with forced extraction, is approximately 38.5% of the complete processing time. This is tolerable considering that no user intervention was required. The unsupported control study took a total of 12h 21m 10s. Thus the EE (even including the time taken for the experiments) saves approximately 28.8% of the run-time of the control simulation.

5. Conclusion

This paper describes a new framework for the integration of a fully automatic Experiment Engine into the *SMARTFIRE* solution process. The prototype offers full user support during the whole solution process and has demonstrated the following benefits for Fire Field Modelling:

- The new framework enables run time interaction between the Case Specification Environment and the CFD Engine and allows feed back to the EE controller.
- Performing coarse mesh test assessments enables the EE to improve the set-up and the meshing quality.
- Starting with a "good" quality of base mesh (using Knowledge Based meshing techniques) reduces the need for refinement and, hence further increases the system efficiency.
- Local adaptive refinement procedures enhance the quality of the Knowledge Based meshing and thus further improve the performance.
- Determination of a suitable time step size has gives savings of processing time due to a reduction in the required number of outer-loop iterations.
- The integration of the Experiment Engine into the solution process offers a considerable enhancement in terms of simulation reliability and (in both test cases) is able to provide the user with the requested solution accuracy.
- The Experiment Engine is (in a limited way) able to emulate an Experts' ability to run a new scenario in a logical and efficient manner, and hence provides a good learning tool for novice users.

6. Further work

The Experiment Engine has demonstrated that it is quite robust and effective although, there are still a number of ways to improve the efficiency, reliability, and robustness of the Experiment Engine and its mode of operation.

Currently, only the meshing is adapted and the control parameter changes are limited to time step size, but other aspects of the scenario could also be controlled. One major disadvantage, of the mesh refinement procedure, is that all grid points will be redistributed after refinement. This makes it problematic to re-use the old solution data values, from the pre-refined mesh, into the newly refined mesh. If the mesh has to be changed, then the EE has to form a new test case (with the new mesh) that is sent to the CFD Engine for computation from the very beginning. This is far from ideal, since the change will require discarding previous solution states that are likely to be valid. In the current research, due to time and resource constraints as well as software maintenance issues, it is difficult to alter the existing solver structure. In the future, it is recommended that a re-mapping mechanism should be investigated, that will allow solution re-use from previous computation. This is likely to make the Experiment Engine process much quicker and more cost efficient, provided that the difficulties in restarting from a remapped solution state, can be overcome.

The ultimate aims of the Experiment Engine research is to always provide an acceptable CFD solution without the need for human monitoring or intervention and to be able to deploy other acceleration/optimization techniques (such as the ICS for optimal solution control) as appropriate. The fault tolerant nature of the EE is also desirable for unmonitored operation. The envisaged system will need a sufficiently wide range of responses to provide assurance of obtaining a solution under extremes of some of the problems that can be encountered during Fire Field Modelling. In addition, further testing and validation, of the new features, against diverse types of fire scenario still needs to be carried out – especially for benchmarking studies and cases that are problematic to converge.

Improvements are now being researched and implemented to further enhance and extend the EE as a robust and automatic control technique to take control of the whole simulation process. The EE is now being further developed to replace the existing ICS, due to the inherent limitations of the ICS strategy. The revised EE will continually monitor the solution process after managing the appropriate set-up and meshing strategy during the experiment phase. The EE will then enter a production run phase in order to make run time adjustment of solution control parameters, as necessary. Consequently, the fully automated EE will be able to actively respond to transient events and heat release changes in the production-run phase. This should help to guarantee convergence and will also allow the EE to monitor for any possible changes that will give performance improvement. In this way, the EE will support the simulation process from start to finish, make the whole process as reliable and efficient as possible and free the user from having to monitor and interact with the simulation. The proposed EE improvements will need to be demonstrated using more complex/larger fire scenarios and more advanced fire modelling techniques (e.g. combustion, smoke models, transitional events and flame spread).

Furthermore, the EE should be developed as a tool for mesh independence studies (also known as grid refinement studies). The common method for a mesh independence studies involves performing a simulation on two or more successively refined grids. As the EE has the ability to progressively refine a mesh and generate test cases on the fly, it should be possible to adjust it to be used for automated mesh independence studies.

References

1. Galea E.R., *"On the field modelling approach to the simulation of enclosure fires"*, Journal of Fire Protection Engineering, Vol. 1 (1), 1989, pp11-22.
2. Versteeg H.K., Malalaselera W., *"An introduction to Computational Fluid Dynamics- The finite Volume Method"*, Longman Scientific & Technical, Loughborough, 1995.
3. Janes D., *"Intelligent control system for CFD modelling software"*, PhD thesis, 2004, University of Greenwich.
4. Taylor S., Petridis M., Knight B., Ewer J., Galea E.R. and Patel M. K. *" SMARTFIRE: An Integrated Computational Fluid Dynamics code and Expert System for Fire Field Modelling"*, Fire Safety Science – Proceedings of the Fifth International Symposium, pp 1285-1296, 1997.
5. Ewer J., Galea E.R., Patel M. K., Taylor S., Knight B. and Petridis M., *"SMARTFIRE: An Intelligent CFD Based Fire Model, Fire Protection Engineering "*, vol 10, no1, pp 13-27, 1999.
6. Taylor S, Galea E., Patel M.K., Petridis M., Knight B. and Ewer J, *"SMARTFIRE: An intelligent Fire Field Model"*, Proceedings Interflam 96, Cambridge, UK, pp 671-680, 1996.
7. Galea E.R., Knight B., Patel M., Ewer J., Petridis M., and Taylor S., *"SMRTFIRE V2.01 build 365, User Guide and Technical Manual"*, SMARTFIRE CD, 1999.

8. Ewer J., Knight B. and Cowell D., *"Case Study: An Incremental Approach to Re-engineering a Legacy FORTRAN Computational Fluid Dynamics Code in C++"*, Advances in Engineering Software, Vol. 22, pp 153-168, 1995.
9. *"SMARTFIRE Verification and Validation Report"* Software Version 2.01, Report Version 1.01, Fire Safety Engineering Group, University of Greenwich, Revision Date 25/05/99.
10. Babuska I. Chandra J. and Flaherty J.E., editors. *"Adaptive Computational Methods for Partial Differential Equations"*, Philadelphia, 1983. SIAM.
11. Arney D.C. and Flaherty J.E. *"An adaptive mesh moving and local refinement method for time-dependent partial differential equations"*, ACM Transations on Mathematical Software, 16:48-71, 1990
12. Dorr M.R. *"The approximation theory for the p-version of the finite element method"*, I. SIAM Journal on Numerical Analysis 21(1984), 1180-1207.
13. Verfurth R. *"A Review of Posterior Error Estimation and Adaptive Mesh Refinement Techniques"*, Teubner-Wiley, Stuttgart, 1996
14. Janes D, Ewer J, Galea E, Patel M and Knight B, *"Automatic Dynamic Control of CFD Based Fire Modelling Simulations"*, Proceedings Interflam 2001. Edinburgh, UK, Sepetember, 2001, Vol. 1, pp 811-822.
17. Watson, Ian D. *"Applying case-based reasoning: techniques for enterprise systems"* San Francisco, California. Morgan Kaufmann, 1997
18. Kumar S., Cox G. *"some guidance on "correct" use of CFD models for fire applications with example"*, Proc. Interflame 2001, Edinburgh, UK, Sepetember, 2001, Vol. 1, pp 823-834.
19. Flaherty J.E. *"Adaptive Finite Element Techniques"* Chapter 8, Finite Element Analysis lecture notes. 2000.
20. Tristano, Joseph R. Zhijan Chen, D. Alfred Hancq and Wa Kwok *"Fully automatic adaptive mesh refinement integrated into the solution process"*, Proceedings, 12th International Meshing Roundtable, Sandia National Laboratories, pp.307-314, Sept. 2003.
21. Liu Y., Moser A., Gubler D. and Schaelin A. *"Influence of time step length and sub-iteration number on the convergence behavior and numerical accuracy for transient CFD"*, CFD 2003, 11th Annual Conf. of the CFD Society of Canada, Vancouver, British Columbia, Canada, May 2003.