M0010775TP

60802308

INTELLIGENT CONTROL SYSTEM FOR CFD MODELLING SOFTWARE

Dominik Sebastian Janes

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

March 2003

To My Wife Anna

And

My Parents

Abstract

In this thesis we show that it is possible to create an intelligent agent capable of emulating the human ability to control CFD simulations and provide similar benefits in terms of performance, overall reliability and result accuracy. We initially consider the rule-based approach proposed by other researchers. It is argued that heuristic search is better suited to model the techniques used by human experts. The residual graphs are identified as the most important source of heuristic information relevant to the control decisions. Three different graph features are found to be most important and dedicated algorithms are developed for their extraction.

A heuristic evaluation function employing the new extraction algorithms is proposed and implemented in the first version of the heuristic control system (ICS 1.0). The analysis of the test results gives rise to the next version of the system (ICS 2.0). ICS 2.0 employs an additional expert system responsible for dynamic pruning of the search space using the rules obtained by statistical analysis of the initial results. Other features include dedicated goal-driven search plans that help reduce the search space even further. The simulation results and overall improvements are compared with non-controlled runs. We present a detailed analysis of a fire case solution obtained with different control techniques. The effect of the automatic control on the accuracy of the results is explained and discussed. Finally, we provide some indications for further research that promise to provide even greater performance gains.

Acknowledgements

I would like to thank my supervisory team at Greenwich University. Professor Brian Knight for his invaluable advice and enthusiasm during my research. Without his support this work would probably still be nowhere near completion. He has also helped me greatly in writing this thesis by offering expert advice on all aspects of technical writing (together with correcting my English).

My other supervisor, Doctor Mayur Patel also assisted me during my studies. He helped me learn how to use SMARTFIRE and kindly allowed me to use his computer to perform many CFD simulations that would otherwise have taken substantially more time. I am also very grateful to my colleague, Doctor John Ewer, who helped and supported me in many ways during my years at the University.

Because of the nature of my research I had to conduct several interviews with CFD experts. My thanks go to the people, who contributed their time and advice. They are (in alphabetical order): Doctor John Ewer, Professor Ed Galea, Doctor Fuchen Jia and Doctor Mayur Patel.

Furthermore, I would like to thank Doctor Mayur Patel and Professor Ed Galea for giving me the opportunity to do my Ph.D research at Greenwich University.

I am very grateful to my friends, Malcolm Kingman and Pauline Quinn, who sacrificed their free time to review this thesis and correct any language and technical mistakes.

Finally I would like to thank Greenwich University for supporting me financially during my studies in UK and for funding this Ph.D. research.

Table of Contents

ABSTRACT **4**

ACKNOWLEDGEMENTS **5**

TABLE OF CONTENTS **6**

CHAPTER 1

INTRODUCTION **11**

1.1 OVERVIEW 11
1.2 RESEARCH QUESTIONS 13
1.3 RESEARCH METHODOLOGY 14
1.4 CONTRIBUTION 16
1.5 MAJOR ACHIEVEMENTS 17
1.6 OUTLINE OF THE THESIS 19

CHAPTER 2

NUMERICAL FIRE FIELD MODELLING **20**

2.1 INTRODUCTION 20
2.2 ADVANTAGES AND LIMITATIONS OF NUMERICAL SIMULATION 21
2.3 COMMON STAGES IN NUMERICAL SIMULATION 22
2.3.1 General problem definition *22*
2.3.2 Detailed problem description *23*
2.3.3 Building a computer model of the problem *23*
2.3.4 Mesh generation *23*
2.3.5 Numerical simulation *24*
2.3.6 Repeat simulation runs *24*
2.3.7 Interpretation and visualisation of the results *25*
2.4 SIMULATION EXAMPLE 25
2.4.1 General problem definition *25*
2.4.2 Detailed problem description *26*
2.4.3 Creating the computer model *27*
2.4.4 Mesh generation *28*
2.4.5 Numerical simulation *29*
2.4.6 Visualisation and Interpretation of the results *30*
2.5 NUMERICAL ENGINE 31

2.5.1 Controlling the solution process ... *35*
2.6 SUMMARY OF TERMINOLOGY ... 38

CHAPTER 3

LITERATURE REVIEW ... 40

3.1 INTRODUCTION ... 40
3.2 CONVERGENCE ACCELERATION TECHNIQUES ... 40
3.2.1 Preconditioning ... *41*
3.2.2 Conjugate Gradient Methods ... *42*
3.2.3 Multigrid methods ... *42*
3.2.4 Fuzzy logic in relaxation adjustment ... *44*
3.2.5 Rule driven system for relaxation adjustment ... *44*
3.3 OTHER TECHNIQUES FOR IMPROVING PERFORMANCE OF CFD CODES ... 45
3.3.1 Group solvers ... *45*
3.3.2 KBS-based mesh generation ... *46*
3.3.3 Latency tolerant algorithms and parallel computers ... *47*
3.4 SMARTFIRE PROJECT ... 48
3.5 HEURISTIC SEARCH ... 49
3.6 SUMMARY ... 51

CHAPTER 4

INITIAL WORK ON A RULE-BASED CONTROL SYSTEM ... 52

4.1 OVERVIEW ... 52
4.2 SMARTFIRE – AN INTERACTIVE CFD SOFTWARE ... 52
4.3 EWER'S RULE-BASED CONTROL SYSTEM ... 54
4.4 KNOWLEDGE ACQUISITION ... 57
4.4.1 Consultations with experts ... *57*
4.4.2 Interview results ... *58*
4.4.3 Analysis and interpretation ... *59*
4.5 NEW RULE-DRIVEN CONTROL SYSTEM (KBS VER 2.0) ... 60
4.5.1 Residual graph assessment ... *61*
4.5.2 Intelligent scheduling of control actions ... *62*
4.5.3 Discussion of initial test results ... *64*
4.6 KBS 2.0 - ANALYSIS ... 65
4.6.1 Adjustments introduce instability ... *65*
4.6.2 Not enough information for control decisions ... *67*
4.6.3 Incorrect control architecture ... *68*
4.6.4 Analysis summary ... *68*
4.7 NEW APPROACH TO THE CONTROL PROBLEM ... 70
4.8 INVESTIGATION OF THE EFFECTS OF CONTROL ACTIONS ... 70

4.9 SUMMARY 76

CHAPTER 5

HEURISTIC SEARCH AS A CONTROL TECHNIQUE 78

5.1 INTRODUCTION 78
5.2 NEW ARCHITECTURE FOR THE CONTROL SYSTEM 78
5.3 HEURISTIC SEARCH ALGORITHM 80
5.4 EVALUATION FUNCTION 88
5.4.1 Convergence speed 91
5.4.2 Irregularities 96
5.4.3 Oscillations 100
5.4.4 Compound evaluation function 106
5.5 FAULT DETECTION 110
5.6 SUMMARY 113

CHAPTER 6

TESTS OF THE PROTOTYPE SYSTEM AND ANALYSIS OF THE RESULTS 114

6.1 INTRODUCTION 114
6.2 PROTOTYPE CONTROL SYSTEM 114
6.3 INITIAL IMPROVEMENTS TO THE ORIGINAL DESIGN 116
6.3.1 Segment detection in convergence prediction method 117
6.3.2 Better divergence detection method 120
6.3.3 Early detection and termination of diverging experiments 121
6.3.4 Enhanced search scheduling strategy 122
6.4 SPEED COMPARISON 123
6.4.1 54kW case with coarse mesh 123
6.4.2 250kW case with fine mesh 126
6.4.3 2.1MW case with fine mesh 129
6.5 FAULT RECOVERY 130
6.6 FULL CONVERGENCE ASSURANCE 132
6.7 INITIAL CONCLUSIONS 132
6.8 SUMMARY 133

CHAPTER 7

SECOND VERSION OF THE CONTROL SYSTEM 135

7.1 INTRODUCTION 135
7.2 IDENTIFICATION OF DIFFERENT STATES DURING SIMULATIONS 135
7.2.1 Diverging graphs 136
7.2.2 Normal graphs 137

7.2.3 Oscillations and slow convergence *137*
7.3 MAJOR HEURISTIC FUNCTION IMPROVEMENTS 138
7.4 SEARCH TREE PRUNING 142
7.4.1 Goal-driven search plans *142*
7.4.2 Dynamic modification of the search plan *144*
7.5 SUMMARY 148

CHAPTER 8

RESULTS 150

8.1 INTRODUCTION 150
8.2 SPEED COMPARISON 150
8.2.1 54kW case with coarse mesh *152*
8.2.2 250kW case with fine mesh *153*
8.2.3 2.1MW case with fine mesh *156*
8.3 FAULT RECOVERY 158
8.3.1 Divergence recovery *158*
8.3.2 Oscillation removal *158*
8.4 ACCURACY ASSESSMENT (STECKLER CASE) 159
8.4.1 Corner stack temperatures *160*
8.4.2 Doorway temperatures *161*
8.4.3 Doorway velocities *162*
8.4.4 Discussion *163*
8.5 SUMMARY 164

CHAPTER 9

CONCLUSIONS 165

9.1 OVERVIEW 165
9.2 EMULATION OF HUMAN CONTROL ACTIONS 165
9.3 SPEED IMPROVEMENTS 167
9.4 RELIABILITY AND FAULT RECOVERY 168
9.5 RESULTS ACCURACY 169

CHAPTER 10

FURTHER WORK 171

10.1 INTRODUCTION 171
10.2 FLEXIBLE EXPERIMENT SCHEDULING 171
10.3 USING "SPREAD-OUT SEARCH" TECHNIQUE 172
10.4 ASSESSMENT OF EXPERIMENT COST VS. EXPECTED GAIN 173
10.5 OTHER AREAS FOR IMPROVEMENT 173

REFERENCES 174

APPENDIX A

RULES USED IN SEARCH TREE PRUNING 186

APPENDIX B

TECHNICAL REFERENCE FOR *SMARTFIRE* 188

APPENDIX C

STECKLER CASE..214

APPENDIX D

MAIN SETUP FILE FOR STECKLER CASE (.INF FILE)...219

APPENDIX E

AUTOMATIC DYNAMIC CONTROL OF CFD BASED FIRE MODDELING SIMULATIONS – INTERFLAM 2001..227

Chapter 1

Introduction

1.1 Overview

Simulations of Computer Fluid Dynamics (CFD) scenarios are very complex numerical problems requiring considerable computing power. There are many factors that influence the accuracy of the results and determine whether correct results will eventually be obtained. The CFD software has come a long way since its first use in research laboratories. Initially the CFD packages were fairly crude, there was no real interface and all the necessary data had to be entered manually into text files. Nowadays the number of industrial applications of CFD grows and the capabilities of modern computers improve rapidly. Most currently available commercial numerical packages contain sophisticated interfaces and numerous tools that assist the user during the whole simulation process, from the set-up to the final visualisation of the results. Some of these enhancements are due to the improvements in computer hardware (e.g. increased speed, advanced graphical capabilities) while others were made possible by substantial research in the relevant domains (e.g. automated mesh generation).

One of the features that was common in early numerical packages was the fact that most programs treated the numerical-processing module as a "black box" that was initialised and then, usually after a very long time, produced the final solution. This approach meant that substantial expert knowledge was necessary to correctly set-up a problem and to choose appropriate control parameters. This was initially acceptable as the problems analysed were small and the required expertise was always at hand since the CFD codes were mainly used in advanced research laboratories. However, as the available computer speed and the capacity of memory chips increased rapidly, it became possible to simulate bigger and more complex scenarios. Unfortunately, these cases turned out to be much more difficult to control and often required tedious monitoring of the simulation process to ensure that the results were correct and produced in reasonable time. This situation encouraged many researchers to develop numerous ways to reduce the complexity of the CFD simulations and improve the performance, stability and ease of use. However, even now, few CFD developers are aiming to provide code interactivity and automated solution control. The emphasis in development is usually directed towards broadening the range of cases that can be run with the software, improving the numerical models and approximations used in the software and providing better quality tools for set-up, meshing and post processing data analysis.

One of the research projects that does concentrate on providing a high degree of control by allowing continuous user interaction to optimise the performance and stability of the simulation is the SMARTFIRE package from The University of Greenwich (Ewer-00, Petridis-95 and Ewer-93). SMARTFIRE is a CFD system reengineered from a legacy FORTRAN code that puts special emphasis on user-friendly interface, real-time progress monitoring capabilities and tools for comprehensive control of the simulation process. SMARTFIRE displays all the relevant information during numerical computations, allowing the user to monitor the simulation, detect problems and make modifications as necessary. This was an important improvement but there were still major problems that could not be fully resolved with this approach. Firstly, CFD simulations often take a very long time, which makes it virtually impossible for a human expert to comprehensively monitor any non-trivial case. Secondly, there is still insufficient knowledge available about which control actions should be applied in particular circumstances. An automated system using rule-driven architecture was implemented in SMARTFIRE with some success (Ewer-98, 99c) but the rules employed proved to be ineffective in complex scenarios although initial experiments showed that substantial benefits could be gained by executing efficient and correct control actions.

This dissertation describes the development of an automated control system with the aim of maximising the performance gains while at the same time improving the reliability, ease of use and efficiency of the numerical software. Intelligent Control System (ICS) uses a heuristic search technique with a comprehensive evaluation function (specifically developed for this application) to determine the best adjustments to the control parameters. The evaluation function employs several pattern recognition algorithms that extract relevant features from residual error graphs. Additional Artificial

Intelligence (AI) techniques are used to improve the overall efficiency of the control procedure.

1.2 Research questions

The main objective of this project is to provide an answer to the following research question:

- ***To what extent can we emulate human ability to control a numerical fire modelling software?***

It is understood that human experts can optimise a numerical simulation by performing various control actions based on their assessment of the current simulation state but there is little information available about the techniques used for this purpose. Therefore the first goal of this work is to identify and formalise the procedures for simulation assessment and proper control actions. Furthermore, the factors that influence experts' decisions have to be identified and their real value verified. When this knowledge is obtained and refined then the appropriate architecture for an automated system capable of using that information to emulate human control actions must be devised. Having the correct architecture it is then necessary to develop algorithms for automatic extraction and assessment of the features, which were deemed relevant in assessing the simulation state. Solutions to all these problems should serve as the building blocks for the automated control system

The initial requirements for the complete control system are as follow:

- A fully implemented system should constantly monitor the simulation progress and be able to perform purposeful and effective control actions.
- A control agent must detect all anomalous states during the simulation and trigger appropriate recovery procedures.
- The AI system should deliver tangible benefits in terms of performance, reliability and ease-of-use while not compromising the accuracy of the final solution.

Therefore, the improvements provided by AI control system will have to be analysed with special emphasis placed on the following issues:

- *Does the system provide improvements in terms of simulation speed as compared to non-controlled simulations?*
- *Can such a system assure the convergence of every time step throughout the whole simulation process?*
- *Does the system recover from solution excursions/faults?*
- *How does the automated control affect the solution accuracy?*

It is believed that an appropriate set of control actions can substantially reduce the simulation time and increase the overall stability and reliability of the simulation process. The potential reduction in execution time is expected to be substantial, as Ewer (Ewer-99c) showed (using a very simple 2D case) that even a basic control system was able to reduce the execution time by 50%.

1.3 Research methodology

At the very beginning it was necessary to develop a better understanding of the problem and to gain experience with the fire simulation software (SMARTFIRE). This involved running several simulations to become familiar with all the steps necessary to obtain the final solution (see Chapter 2). Performing complete simulations was essential to understanding of how much expertise was required to control a fire simulation correctly and efficiently.

The next stage of the research focused on determining how other, more experienced users, used and controlled SMARTFIRE. A prototype control system developed and implemented in SMARTFIRE by John Ewer (summary available in 4.3) was analysed. This was the starting point that subsequently led to the formal process of knowledge acquisition, aimed at identifying the techniques used by the experts to control the simulation process (4.4). Furthermore, a review of the available literature was conducted to assess how other researchers tackled the problem of convergence acceleration and automatic solution control to ensure that this research was not repeating the work of others (3.2-3.3).

The knowledge acquisition and subsequent analysis resulted in the development of an enhanced version of Ewer's rule-based system (KBS 2.0 – see 4.5). However, the simulations of standard fire cases revealed the limitations of the rule-based system and it became apparent that a different approach was necessary to obtain satisfactory results (4.6). Several generic types of control action were tested on a range of cases and the results were analysed. Consequently, a new architecture based on heuristic search was proposed (Chapter 5). This approach (intelligent search with elements of trial and error) was closely modelled on the techniques used by the human experts to control real simulations. A literature study of heuristic methods was performed to look for research that shared common features with the problem of simulation control. A general overview of heuristic methods is given in 3.5 while the details of the most relevant heuristic systems are presented in 5.3 and 5.4.4.

The construction of a suitable heuristic evaluation function was an essential part of the new architecture. Further interviews with experts and the analysis of the results of many experiments (4.8) identified three different features of the residual graphs that were most relevant to the control process. Consequently, dedicated feature extraction algorithms were developed and gave rise to a prototype three-part evaluation function (5.4).

This new approach was first implemented in a prototype system (ICS 1.0 – see 6.2), which was further improved and then tested on several test cases (6.4, 6.5 and 6.6). A number of issues were identified and prompted further analysis, which resulted in significant improvements. The cost of the search algorithm was substantially reduced and the evaluation function was further improved. Statistical analysis gave rise to goal-driven search plans and dynamic plan modification. These improvements were incorporated in ICS 2.0 (and are described in Chapter 7).

ICS 2.0 was fully tested and then used to produce the final results of this thesis (Chapter 8). The summary and the conclusions are presented in Chapter 9.

1.4 Contribution

This research demonstrated that a sufficiently sophisticated intelligent software agent was capable of using methods similar to those employed by human experts to effectively control numerical software. A number of diverse AI techniques were used in order to successfully emulate human control actions. It was revealed that, due to the complexity of the problem, a simplistic rule-based approach was unable to provide satisfactory improvements and therefore several different AI paradigms had to be employed to comprehensively model human control techniques.

The research produced an intelligent software system that emulated human control actions using new control methods, which were discovered in the course of the work. The agent uses a heuristic search with a comprehensive evaluation function constructed using the knowledge elicited from experts and inferred from experiments. Diverse algorithms were developed to model human assessment procedures as closely as possible:

- Fourier Transform and digital filters to assess amplitude and duration of residual error oscillations.
- Linear approximation augmented with segment identification was applied to convergence forecasting and divergence detection.
- Algorithmic graph approximation was used for irregularities assessment.

The final system provided significant benefits by reducing the processing time and enhancing the reliability of numerical simulations. ICS proved to be very competent in recovering from faults and ensuring full convergence throughout all time steps. These very important improvements show that the heuristic search, modelled on an intuitive search routinely performed by humans, can be effectively used as a control technique. Consequently, a complex control problem was solved using techniques from AI domain.

Furthermore, the detailed statistical analysis of the effects and nature of various control actions and their combinations revealed new knowledge that was subsequently acknowledged by experts. It is worth noting that initially a few experts described some of the conclusions as counterintuitive although eventually agreed that they were valid.

The tangible benefits obtained by ICS suggest that residual errors were correctly identified as the main source of information required to control the simulation effectively. However, the results also indicate that by extracting additional information the system could be made more efficient and perhaps provide even bigger performance gains.

ICS proved very competent in dealing with exceptional situations like divergence or excessive oscillations. The recovery procedures used by the system were always able to recover from divergence and ensure that all the time steps converged.

The physical results were also analysed to assess whether the ICS has any impact on their accuracy. It was concluded that the ICS-controlled simulation produced physically sound results, which were in good agreement with non-controlled simulation using the same mesh, and with the golden-standard simulation. However, the results were not identical. A golden-standard case (a non-controlled simulation using a very fine mesh and high number of iterations) was used to determine which simulation was more accurate but the results proved to be inconclusive. Consequently, the experts' assumption that full convergence of all time steps guarantees absolute accuracy could not be indisputably confirmed and should be further investigated. Additional research is also needed to reveal the cause of the observed differences in results between the automatically controlled simulation and the non-controlled one.

1.5 Major achievements

This research exceeded the initial expectation and actually delivered a commercially viable solution to the complex control problem. It not only successfully modelled a human control technique but went further and discovered new techniques for controlling a CFD system, which were subsequently implemented to provide further improvements. It was demonstrated that a reduction in processing time in excess of 50% could be achieved while concurrently delivering considerable enhancements to the reliability of the simulation. Furthermore, the research results indicate that even better performance could be achieved by enhancing the current architecture and using a more sophisticated evaluation function.

Another main achievement is the comprehensiveness of the control technique. The system is remarkably robust, which means that most simulations can be left unsupervised and ICS can be trusted to control the whole process efficiently and accurately. This feature is of paramount significance for new users or persons who are not CFD experts. Providing that they are able to set up a case correctly, they can rely on ICS to control the simulation and deliver accurate results in reasonable time. Such enhancements in ease-of-use can lead to wider acceptance of the CFD software by non-experts and encourage its use for a variety of industrial applications, e.g. all stages of product development (design, manufacturing and testing). Furthermore, due to the enhanced stability and tangible reduction of processing time, ICS could also be an invaluable tool for CFD experts by helping them simulate complex cases in shorter time and with less manual intervention. The system's ability to automatically recover from divergence relieves the expert from the tedious task of constant monitoring of the simulation state while full convergence assurance guarantees the accuracy of the final results. The speed factor is also very important as, even though experts can potentially outperform ICS, this is usually only possible if they commit a lot of resources and spend considerable time continually fine-tuning the numerical solution. This is certainly not a practical approach, especially as the simulations often take several hours or even days.

As part of this research, a comprehensive analysis of the control methods was also conducted to try to expand and formalise the knowledge elicited from the experts. This resulted in better understanding of the effects of various control actions and revealed facts that were not immediately apparent to the experts. This knowledge was used to enhance the currently used control procedures and recommendations that can be applied independently from ICS were produced.

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 CFD codes and perhaps even to numerical packages outside the CFD domain using similar numerical solvers. The proposed application of heuristic search should be sufficiently generic to suit other similar control problems. Of course, the evaluation function would have to be adapted or even completely rebuild and other components of the system substantially modified (e.g. the KBS system governing the dynamic modification of the search plan might require a different set of rules). However, the general principle should still be valid. Since ICS was designed to closely emulate human control actions, then as long as human experts use similar procedures with other numerical packages (which is believed to be the case), an adapted ICS should still be able to provide tangible improvements.

1.6 Outline of the thesis

Chapter 1 of this thesis provides an overview of the research problem, and outlines the contribution made. It presents main achievements of this work and the benefits in potential applications. Chapter 2 provides more details about CFD simulations and presents terminology used throughout this work. Chapter 3 reviews current research into convergence acceleration and stability enhancements of numerical methods. It also presents related research into control systems that employ similar AI techniques. Chapter 4 documents initial attempts to control CFD software by a rule-driven system and contains analysis of the reasons that contributed to its failure. Chapter 5 introduces a new architecture based on a heuristic search. There is a detailed description of the knowledge elicitation process that led to the search-based solution and the development of the heuristic evaluation function. Chapter 6 examines a prototype of the new control system (ICS ver 1.0) and the initial results. It identifies the shortcomings of the prototype and outlines the ways of overcoming them. The system is further enhanced and uses additional AI techniques: goal-driven search and simple planning with dynamic rule-driven plan modification. A detailed description of these improvements and the final design of ICS ver 2.0 are presented in Chapter 7. Chapter 8 compares and analyses the results of non-controlled simulations vs. ICS-controlled ones. Chapter 9 presents the conclusions. Directions for future work are detailed in Chapter 10.

Chapter 2

Numerical fire field modelling

2.1 Introduction

Although CFD came to prominence fairly recently it quickly found its way to an overwhelming number of diverse industries ranging from nappy production to jet aircraft design. But before we go into more detail, we should try to answer the fundamental question. What exactly is CFD? A brief definition is offered by Shaw (Shaw-92):

> ***Computational Fluid Dynamics (CFD)*** *can be described as the use of computers to produce information about the ways in which fluids flow in given situations. CFD embraces a variety of technologies including mathematics, computer science, engineering and physics, and these disciplines have to be brought together to provide the means of modelling fluid flows. Such modelling is used in many fields of science and engineering but, if it is to be useful, the results that it yields must be a realistic simulation of a fluid in motion. At present this depends on the problem being simulated, the software being used and the skill of the user.*
>
> *'Using Computational Fluid Dynamics'*
> *C. T. Shaw*

Although we are constantly surrounded by fluids (normally in the gaseous form) we are not always aware of their presence, which might create a misleading picture about the usefulness and applicability of CFD. The truth is, virtually every major industry uses CFD in one way or another. Therefore the following list is by no means exhaustive but focuses on examples that best emphasise the diversity of CFD applications:

- **Aircraft design** – assisting in wing and body shape design
- **Car design** – aerodynamics, engine design
- **Weather forecast** – predicting the weather and natural disasters (floods, storms and even volcano eruptions)
- **Soldering and moulding** – improving the efficiency and reliability of technological processes

- **Safety engineers** – determining the effects of fire and explosions
- **And many, many more...**

This chapter provides an overview of CFD with special emphasis on its applications in fire field modelling. It also introduces the concepts and terminology used in this dissertation. It does not attempt to present an exhaustive explanation of CFD but tries to place this research in a broader context and to provide the necessary background information for readers from outside the CFD domain. For more comprehensive and in-depth treatment, one should consult any of the introductory books on CFD (Anderson-95, Shaw-92 or Wendt-92).

2.2 Advantages and limitations of numerical simulation

There are many reasons why a computer simulation is currently a method of choice for a variety of applications. One of the most important factors is, of course, money: a simulation usually costs a fraction of corresponding experimentation cost. Furthermore, it is much quicker and allows efficient testing of various configurations and conditions, which would otherwise require a tedious and expensive set-up for each separate experiment. Another area where computer simulation shows its advantages is where the experiment is either difficult or very dangerous to conduct. Extreme conditions like very high temperature or pressure can be simulated with ease. Dangerous factors that make conducting the experiments impractical, e.g. production of toxic substances or a possibility of explosion do not affect the simulation – one can safely and cheaply create and observe the results of any potentially disastrous action.

With all these advantages it might be tempting to conclude that the real experiments are obsolete and that a computer simulation is the best and only tool – both in science and industry. However, things will probably never become that simple. The main problem is that the computer-generated results are only as good as the physical model employed. If the model does not describe the reality accurately enough simulation results will occur, which differ significantly from the real life scenarios. There are also cases where the simulation is so computationally expensive that only experiments can provide accurate results in reasonable time. The classic example of such problem is turbulence.

Currently, the turbulence cannot be efficiently simulated apart from very simple cases and even then very powerful computers are required. A number of simplified models exists, which have been created specifically to make simulating turbulence feasible but the associated assumptions and approximations often make the results too inaccurate to be of any use.

Furthermore, computer simulations require considerable skill and experience in order to set them up properly and then run efficiently. And even if everything goes well and the simulation produces the desired results, these are usually in a form of a huge array of numbers, which have to be post-processed and then interpreted to form any conclusions. Of course, all these problems are very well known and many researchers are working to resolve or alleviate some of these issues. Consequently, we can safely assume that computer simulations will become even more popular in the future.

2.3 Common stages in numerical simulation

To make the concept of a simulation more concrete, this section presents the details of each simulation stage starting with the problem formulation and then all subsequent stages that lead to the final results and their interpretation. Since the CFD simulations are inherently complex, this overview aims to provide more information about the range of skills required to perform a successful simulation.

2.3.1 General problem definition

A CFD problem can be defined in many different ways. The definition may include a very detailed description of the whole environment and various factors that are believed to have influence on the results. On the other hand, a problem can also be described with a single sentence (e.g. “A medium-sized room with a single window, door and a small fire in the middle”). Of course, the fewer details there are in the description the more assumptions have to be made about the domain.

2.3.2 Detailed problem description

Regardless of the amount of information present in the initial specification it is always necessary to build a full and detailed definition of the problem that is being solved. If all the necessary data was already provided in the description then the task is very straightforward – the specification may need little more than reformatting to fit into the required template/layout. However, if the data is incomplete then the missing pieces of information have to be reconstructed by making educated guesses about the domain. For example, if the problem specification does not include the initial temperature, an arbitrary default value will be chosen. The same pattern applies to all information, like domain dimensions, standard pressure, fire output, etc. The process of choosing appropriate default values requires experience and extensive knowledge, often from a variety of fields (not only computational-modelling). It is a very important part of the set-up since incorrect problem specification can invalidate the final results.

2.3.3 Building a computer model of the problem

The next step is to translate the problem definition into an equivalent computer model. It is important to differentiate this phase from the previous one (creating the detailed description of the problem) as computer models have various limitations and the original specification often has to be significantly simplified to fit the model requirements. For instance, complex geometry may have to be represented by a set of cubes while changes in fire growth are approximated by a heat output curve. Again, substantial experience is required to make appropriate decisions to minimise the adverse effects on the quality of the final results and to avoid performance problems.

2.3.4 Mesh generation

Before a numerical simulation can be performed the domain has to be meshed, i.e. divided into discrete cells. The quality of the mesh is one of the most important factors that determine whether the simulation will be successful. An inappropriate mesh may adversely affect the results and even cause the computational engine to fail while a correct and well designed mesh can reduce the simulation time and significantly improve the results accuracy. For more details about meshing one should consult a dedicated book (e.g. Knupp-94).

2.3.5 Numerical simulation

At this stage iterative solvers are employed to perform the actual simulation and produce the results. This is normally the most time-consuming part of the whole process although the actual time required depends on a variety of factors. The following list is by no means exhaustive but is intended to show the diversity of factors that determine the simulation time:

- Mesh quality/accuracy
- Number and size of time steps
- Required accuracy of the solution
- Computational power available

This is the crucial part of the simulation and therefore it is described in more detail later in this chapter (section 2.5).

2.3.6 Repeat simulation runs

This phase is not required but occurs quite frequently in numerical analysis of complex scenarios. Often the first run does not produce satisfactory data, takes too long or diverges and therefore produces meaningless results. In such cases the computer model of the problem and/or the control parameters are revised after which the numerical simulation is restarted. Occasionally, obtaining the correct results requires a lengthy process of iterative adjustments that eventually lead to an acceptable solution.

2.3.7 Interpretation and visualisation of the results

The raw results produced by the numerical engine can consist of a flat file (or files) containing many numerical values. In order to extract any useful information from the data, the results have to be post-processed and then interpreted. For very simple scenarios the interpretation can be trivial but for complex cases covering a long period of time only sophisticated visualisation techniques allow a full analysis of the results. One of the very effective visualisation techniques is a 2-dimensional (or even 3-dimensional) animation of all time steps using the data produced by the numerical engine. However, normally the animation is not necessary and usually the results are presented on a set of graphs showing the changes in the relevant variables, perhaps complemented by plots of crucial variables in important sections of the domain. It is however important to remember that the results do not just "pop out" from the numerical engine but that they have to be post-processed in order to allow a full analysis of the data.

2.4 Simulation example

We will now focus on an example case and present the full simulation process starting from the very early "draft" specification, through all the stages to the eventual visualisation and interpretation of the results. This case is neither a template for setting up and performing any simulation nor does it purport to present all factors that should be considered while setting up a similar case. It is provided here exclusively to illustrate some of the practical issues and concepts that are commonly encountered while performing numerical simulations.

2.4.1 General problem definition

We set out to model the flow in a small room (3m x 3m x 2.2m) with an electric heater in the middle of the floor (Figure 2-1). The room has a single door and one window. Both the window and the door are open. The walls are made of brick and the roof is made of concrete and all are well insulated. We are interested in the resulting flow (velocities) and temperature distribution.

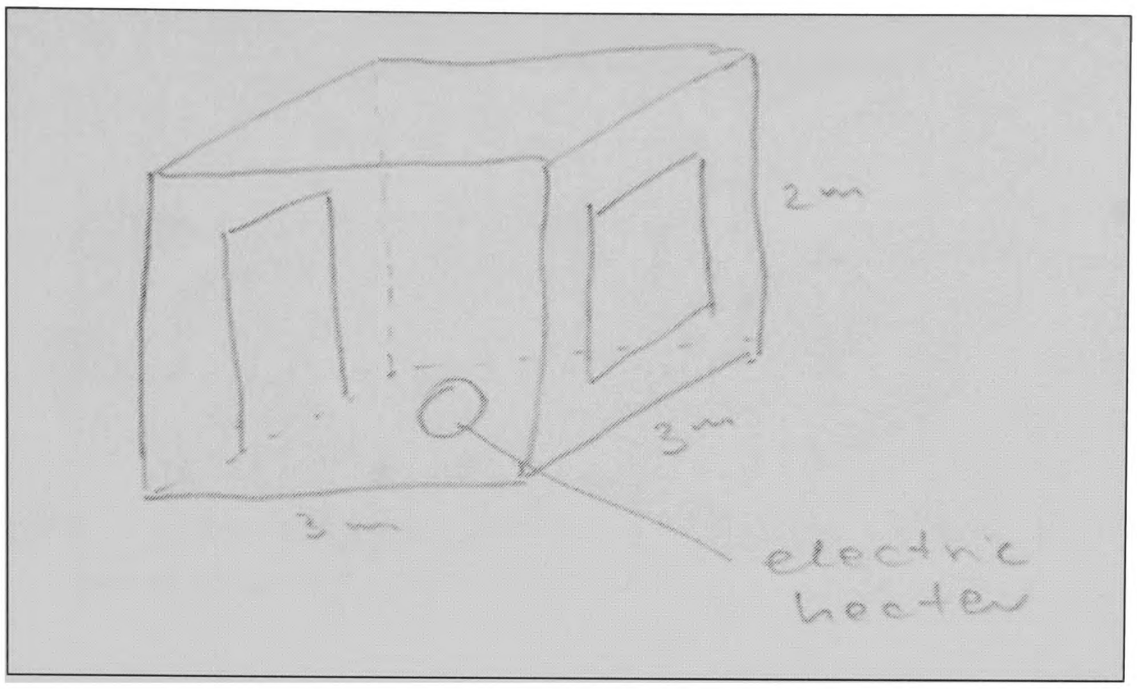


Figure 2-1 Initial case specification (drawn by hand)

2.4.2 Detailed problem description

After analysing the initial problem definition we now have to add the missing details in order to obtain a complete scenario specification. All the gaps in case description have to be filled by making reasonable assumptions. In this case the amount of missing information is small (as the scenario is very simple) but there is still a surprising number of 'guesses' that have to be made. The following list contains only some examples (with the default values chosen in brackets):

- Ambient temperature *(303.75 ^{O}K)*
- Pressure *(1013 HPa)*
- Wall heat characteristic *(adiabatic)*
- Any other items in the room? *(none)*
- Exact window position *(in the middle of the wall)*
- Window size *(1.0m x 1.0m)*
- Power of the heater (*1000W*)
- And so on...

Figure 2-2 shows how the initial description was transformed into a full case specification.

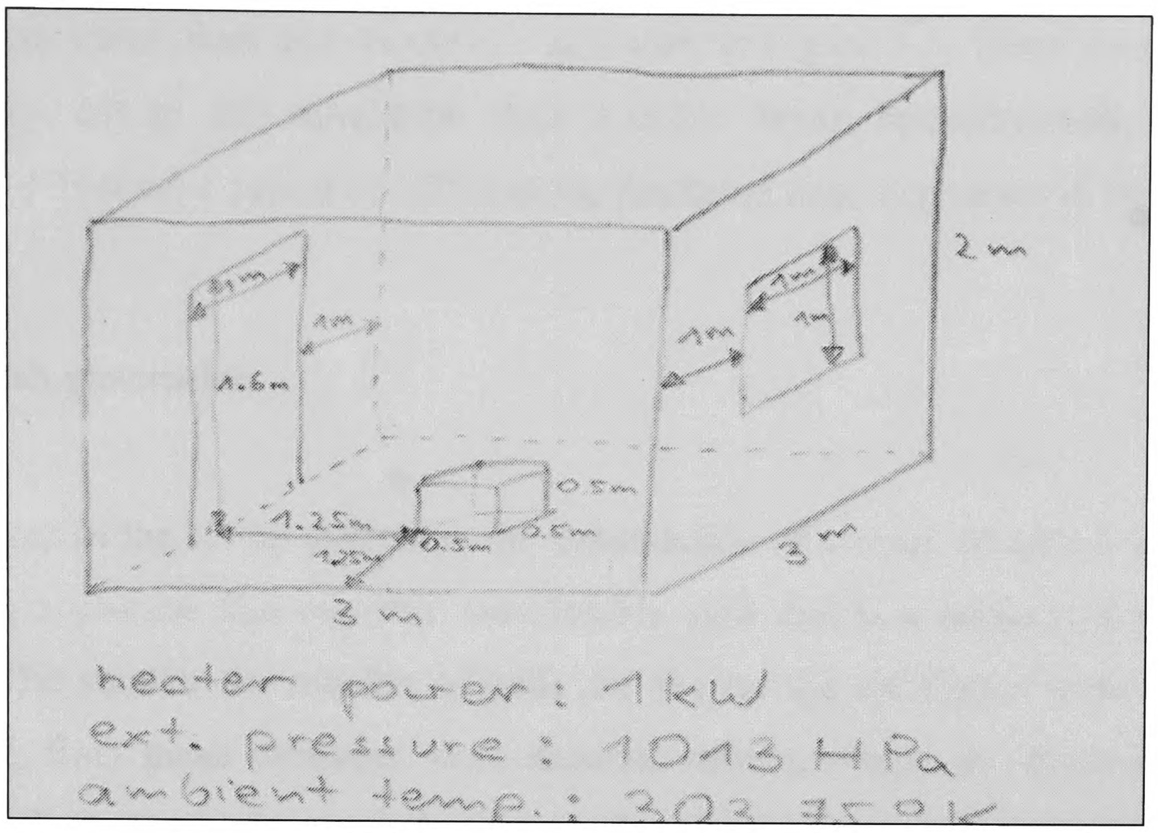


Figure 2-2 Detailed case specification (only part shown)

2.4.3 Creating the computer model

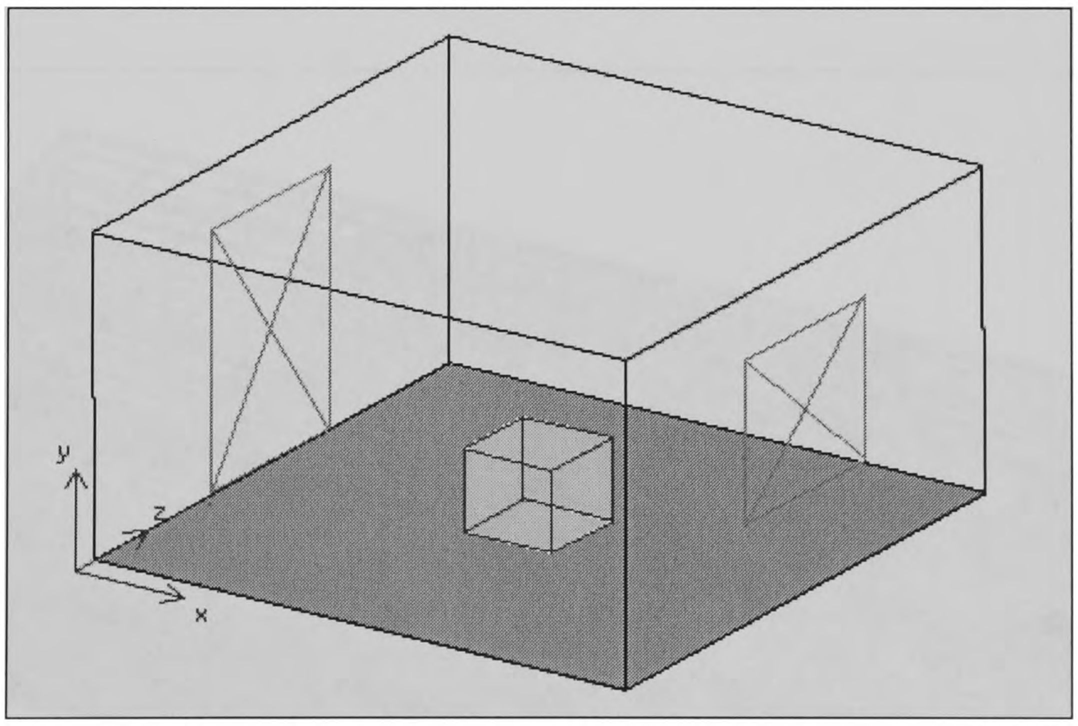


Figure 2-3 Computer model of the room

In this phase, apart from creating a computer model of the room geometry (usually greatly simplified), we also have to decide on the physical models that are to be used. For instance, the heater will be modelled by a simple heat release curve. In this case the curve is extremely simple as the heat output is constant and equals 1kW. The heater itself is represented as a cube since its real shape is irrelevant for our purposes. For simplicity we assume that the heater does not have a fan. The room becomes a simple box with two vents (door and window) – as shown in Figure 2-3. There are no changes in geometry during the simulation (like window being opened/closed, etc.). The simulation will cover a period of 100s and the preferred time step size will be 1s.

2.4.4 Mesh generation

The next step in the set-up process is the construction of a mesh (Figure 2-4). Domain meshing is a process that requires considerable skill and is a subject of very active research. The smaller the number of cells the shorter the simulation time but on the other hand, finer mesh produces more accurate results, which are closer to real-life conditions. In our case the mesh is finer in the regions where we expect to have the most complex flow – around the heat source, close to the vents and walls. However, the cells are much larger in regions that are believed to have little impact on the overall flow. Furthermore, two extended regions have been created outside both vents (window and door) to correctly model the flow to and from the room.

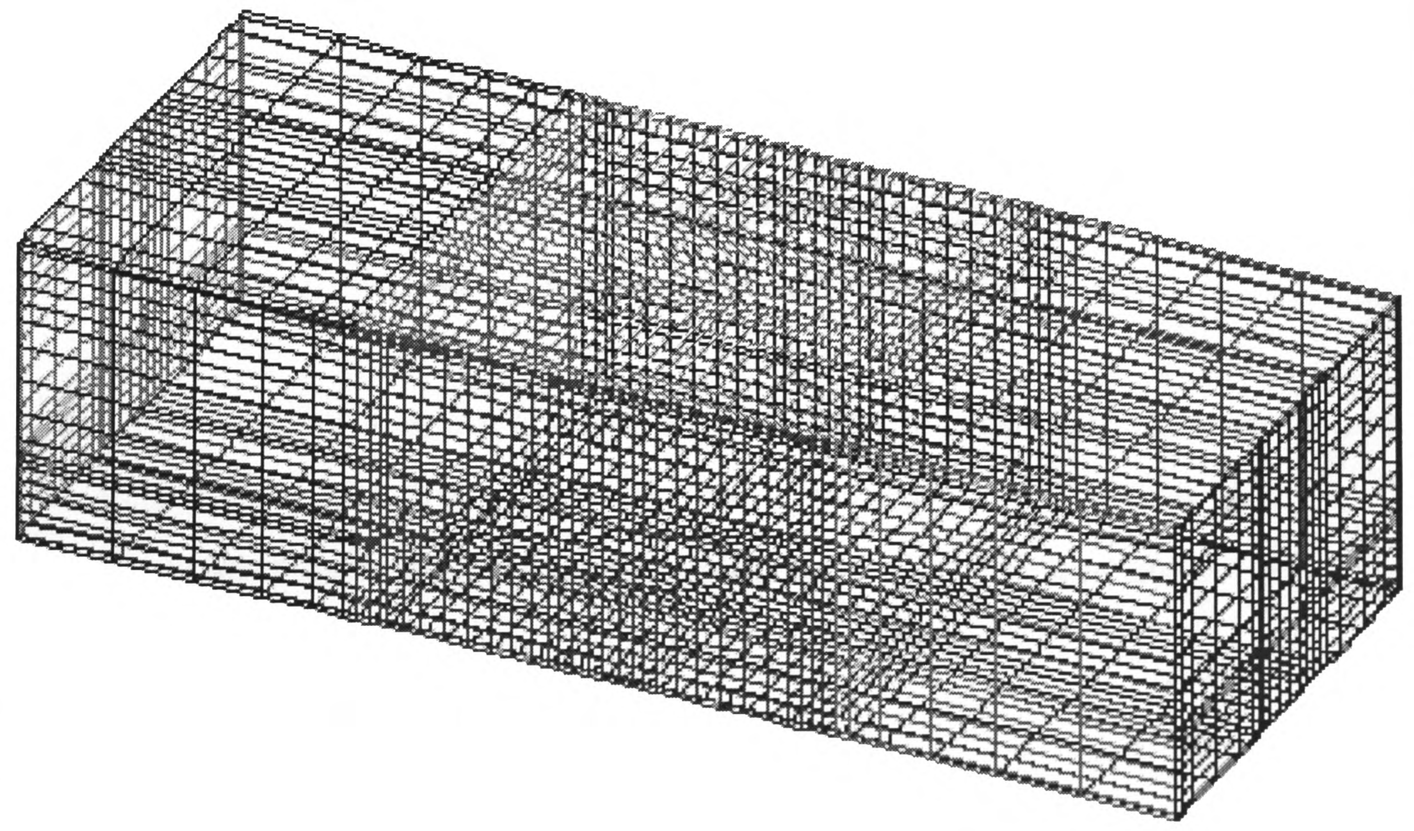


Figure 2-4 Meshed domain (with extended regions)

2.4.5 Numerical simulation

Results are produced during this stage of the process. Appropriate control parameters are initially chosen and adjusted during the simulation (if allowed by the software) to obtain correct results in reasonable time. Usually, at the beginning of the simulation one chooses the time step size (here 1s), number of time steps (usually determined by the simulation period required – 100 time steps in this case) and relaxation parameters. Dynamic modification of these parameters during the simulation may have significant effect on the performance and accuracy of the results as the conditions in the domain may change significantly during the simulation. Considerable experience and thorough understanding of the simulation processes is necessary to perform optimal control actions. The experts usually monitor residual errors to assess the current simulation state – a typical residual graph is shown in Figure 2-5 and is further explained in section 2.5.

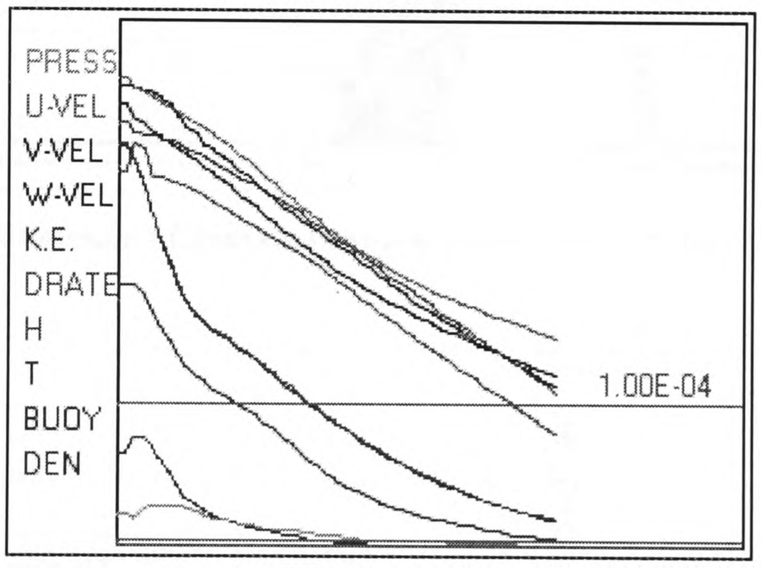


Figure 2-5 Example residual error graph

The graph in Figure 2-5 shows a single time step and the residual values are shown on the vertical axis while the horizontal axis represents the iteration number. This convention is used throughout the thesis in all graphs where the axis are not shown.

2.4.6 Visualisation and Interpretation of the results

The results may be presented in a virtually unlimited number of ways dependent on what is considered the most important result. In this case we were interested in velocities in the centre of the door at the end of the simulation (t=100s) and the temperature distribution in the middle section of the room at the same time. The velocity profile is shown on a graph in Figure 2-7 while the snapshot of the temperature distribution is shown on the 2D slab from the centre of the room (Figure 2-6). The results clearly show what everybody knows intuitively: a person using the heater with both the door and the window open has little chance of warming up the room.

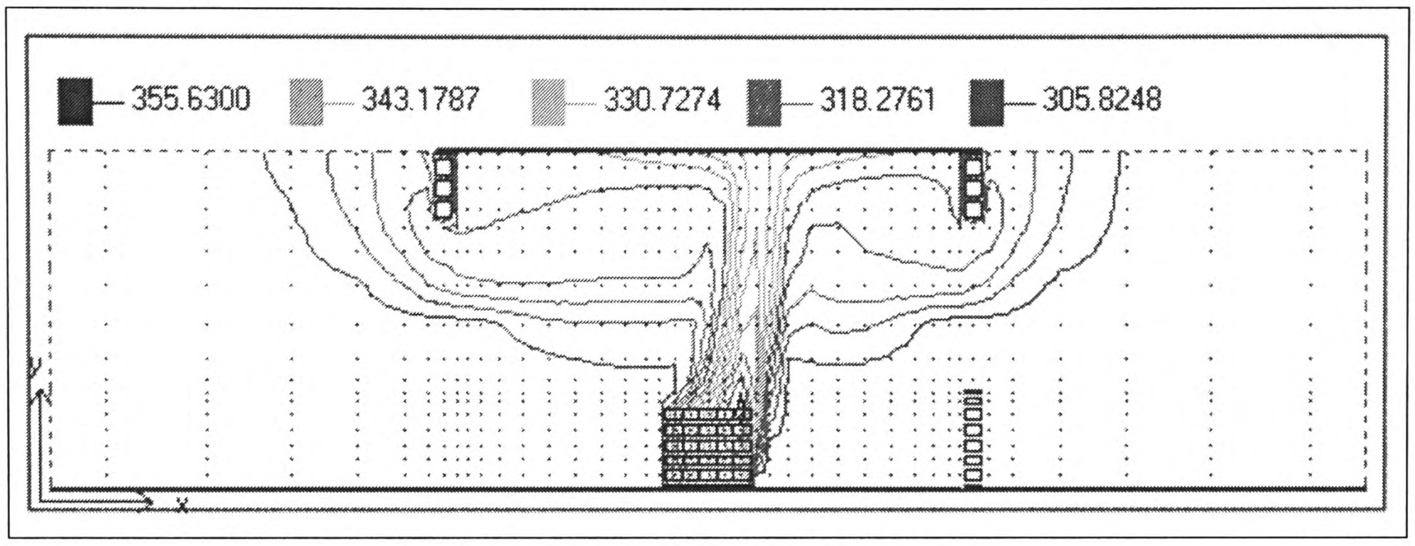


Figure 2-6 2-D section of domain showing temperature distribution at t=100s

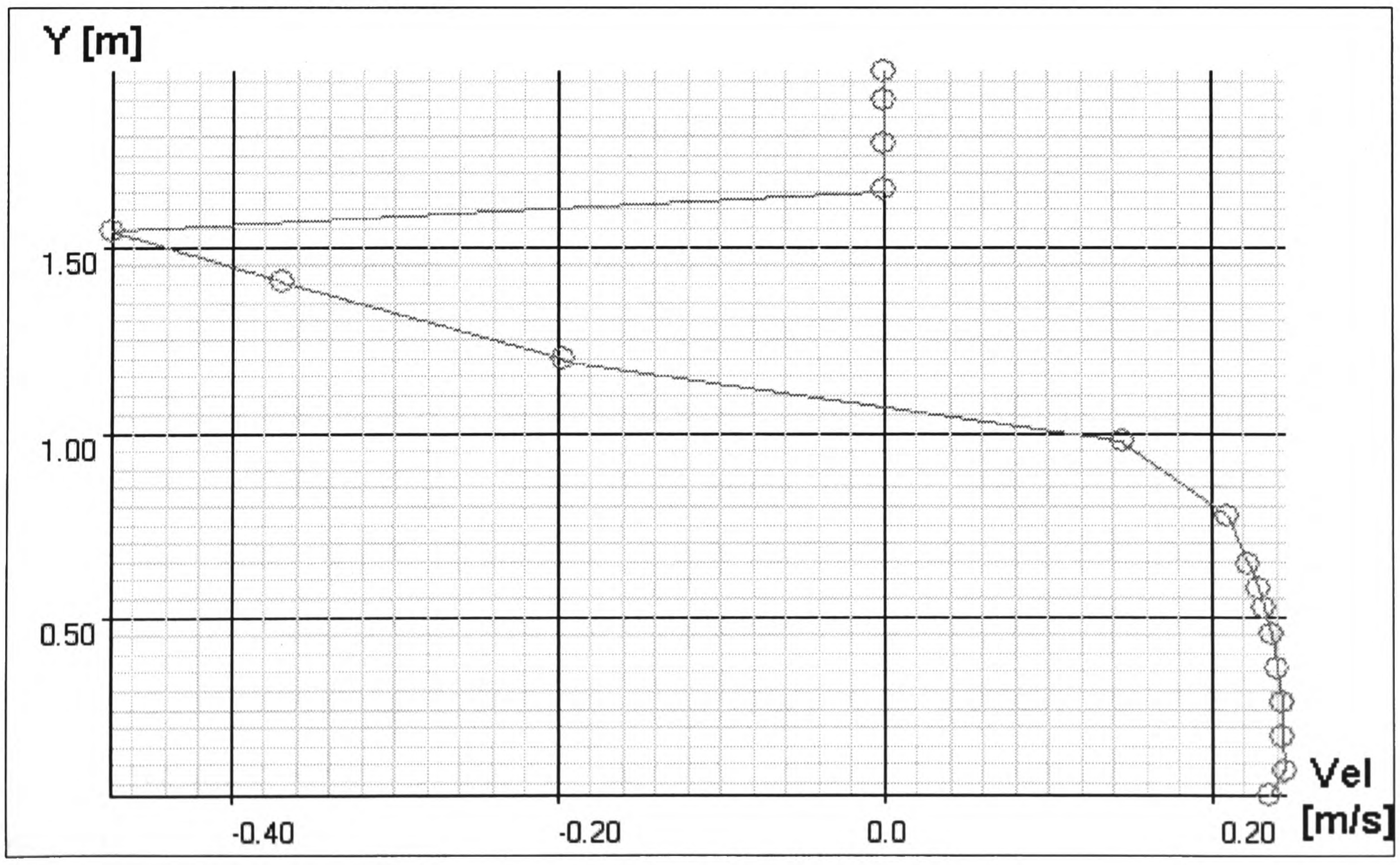


Figure 2-7 Velocities through the door at t=100s

2.5 Numerical engine

We have shown that several different factors contribute to the success of a numerical simulation. Nevertheless, the actual number-crunching phase remains the essential element of the whole process. Even the best set-up simulation will not produce any results if the numerical engine does not work properly. The computations must be both efficient and reliable, otherwise the results obtained would be inaccurate or impossible to obtain within a reasonable period of time. Since this dissertation focuses on an intelligent system, which dynamically controls the iterative solver, this stage of the simulation is presented here in more details.

In CFD there are two main types of problem being solved: *steady state* and *transient*. We will initially concentrate on the former and use it to explain common concepts in numerical simulation. In the steady state case one is only interested in the final stage when the simulated domain reaches equilibrium and not in the preceding, intermediate phases. A very simple example of such problem is a rectangular plate with constant boundary conditions (i.e. constant temperature on the edges). If this case is simulated as a steady state then the initial state or any time-dependant variables are not important and only the final stable temperature distribution in the whole plate is of any interest. Consequently, the desired result of the simulation is a set of numbers that represent the temperature distribution over the whole plate when it reaches a stable final state. Note that the simulation does not determine when this state is reached but only what is the final temperature distribution.

A numerical simulation can be described as an iterative search for progressively better approximations of the solution. There is, however, one obvious problem associated with this approach: since the final solution is not known in advance (obviously - if it was known then we would not have to run the simulation) then it is difficult to measure the accuracy of the current approximation. This brings us to another very important term in CFD: a *residual error (residual)*, broadly defined as a difference between two consecutive approximations. The actual formula varies between models and implementations but the underlying principle remains similar: the residual error is a convenient measure of the current result quality. The error is normally computed separately for each variable in every cell and then averaged over the whole mesh to produce a single residual value for each variable.

Figure 2-5 presents a typical residual error graph over the number of iterations performed. One can see that during the first iterations the residuals are relatively big. This is perfectly normal since we start from an arbitrary “guess” which is likely to be substantially different from the actual solution and therefore the simulation state changes significantly at the beginning as each approximation is quickly getting nearer to the correct state. In the final stage the residual error diminishes since the simulation is close to the correct solution and consecutive approximations change very little. The simulation is believed to have *converged* (i.e. found the correct solution) if the residual error is lower then the predetermined *tolerance*. The tolerance value is necessary since it is unrealistic to expect the error to disappear completely. Fortunately in practical applications it is never necessary to obtain the results with absolute accuracy (absolute accuracy can be obtained by solving the equations analytically but this is only possible for very simple cases). Looking at the graph displayed in Figure 2-5 it is clear that the residuals are about to converge to the predetermined tolerance (10^{-4})

One should also remember that Figure 2-5 shows an example of a typical well-behaved residual graph and that other graphs often look very different, especially if the simulation experiences problems in finding the correct solution or becomes unstable. Examples of real-life graphs are presented in Figure 2-8

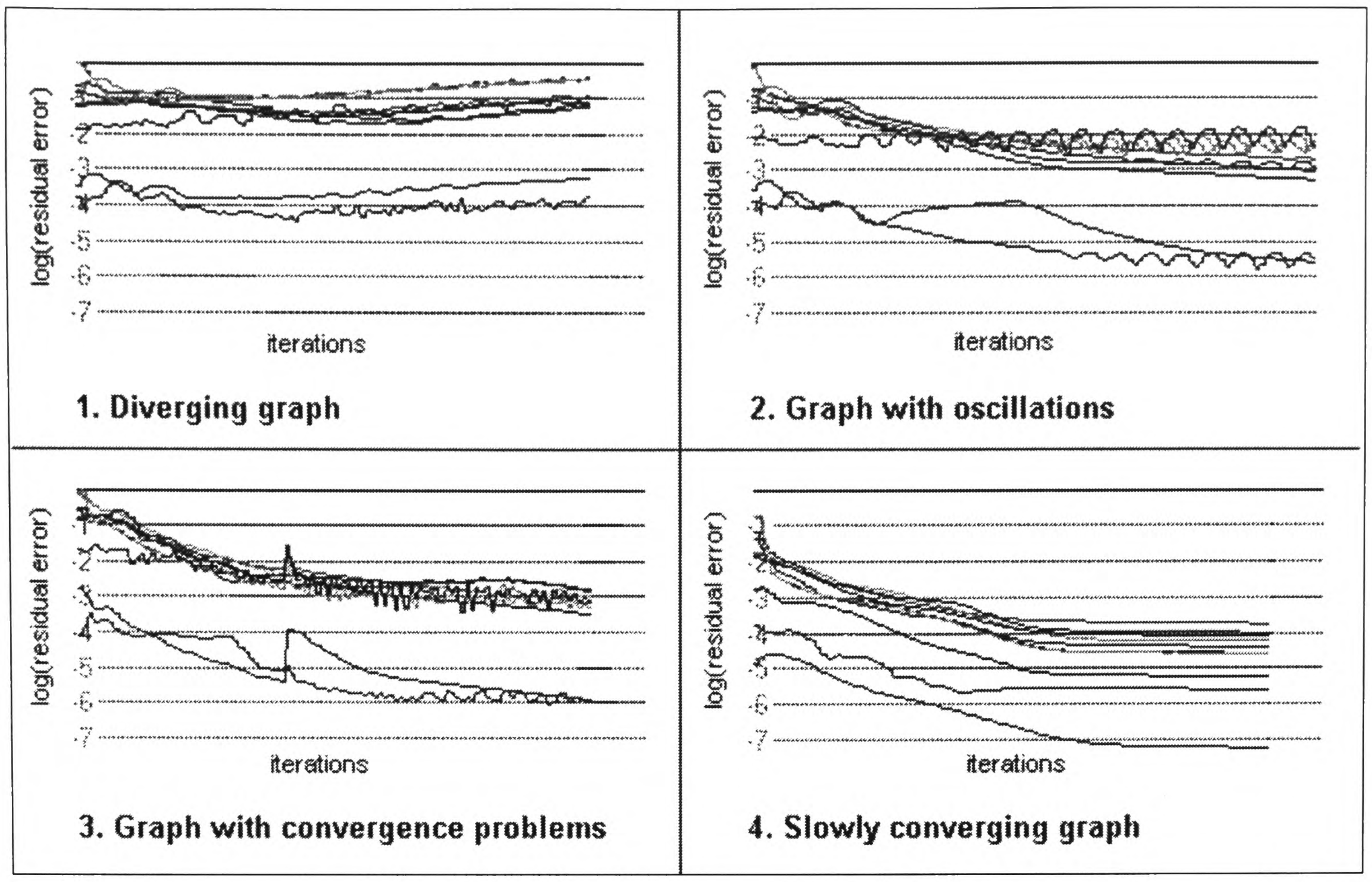


Figure 2-8 Residual graphs from real simulations

Using several residual errors as a measure of accuracy is more difficult but still very convenient. The experts have different opinions about how to define the convergence using several residuals. Some believe that all residuals should reach the tolerance while others are only interested in the "most important" variables like pressure or velocities. Nevertheless, it is generally agreed that the simulation is progressing well if all the residuals are decreasing and there is a good chance of reaching tolerance within a reasonable period of time. It is difficult to avoid expressions like "good chance" and "reasonable period" since the actual values depend very much on the case and application. For some scenarios 24h of processing per time step might be acceptable while for others anything above a minute would be too slow. It is often useful to compare the performance between different time steps from the same problem, which brings us to *transient cases*. Transient cases are more general than steady state ones since they introduce the time variable into the simulation. The following comparison of the steady state and transient cases should explain the difference:

- **Steady state problem**: start with guessed values for variables ϕ and proceed to obtain the values of ϕ at a point when the simulation reaches a steady state (the flow does not change)

- **Transient (unsteady) problem:** start with values of ϕ at time **t** and a guess for ϕ at time **t+Δt**, then find the values of ϕ at **t+Δt**.

Usually, a transient case consists of several consecutive time steps but a scenario with a single time step of a finite length can also be considered transient. Another important difference must be stressed: in transient cases the initial domain state (values of solved variables at the beginning of the simulation) affects the results while in the steady state the initial conditions often have no impact on the final outcome (although they can affect the performance). Consequently, transient cases require more thorough and detailed set-up procedures. In this project we will deal exclusively with transient cases, as they are more general and also more difficult to control. Figure 2-9 presents snapshots of three different time steps from a transient simulation in a simple room. We can clearly see how the flow develops through time and the plume starts to lean over until it reaches equilibrium. In many cases, the result of the final step is equivalent to the result of a steady-state simulation but since the transient simulation also produces the results from the intermediate phases, it allows us to analyse the flow development. Fire modelling relies heavily on transient simulations, as they make it possible to observe the effects of various events happening in the domain: windows breaking, flashover occurring or perhaps the effect of sprinklers.

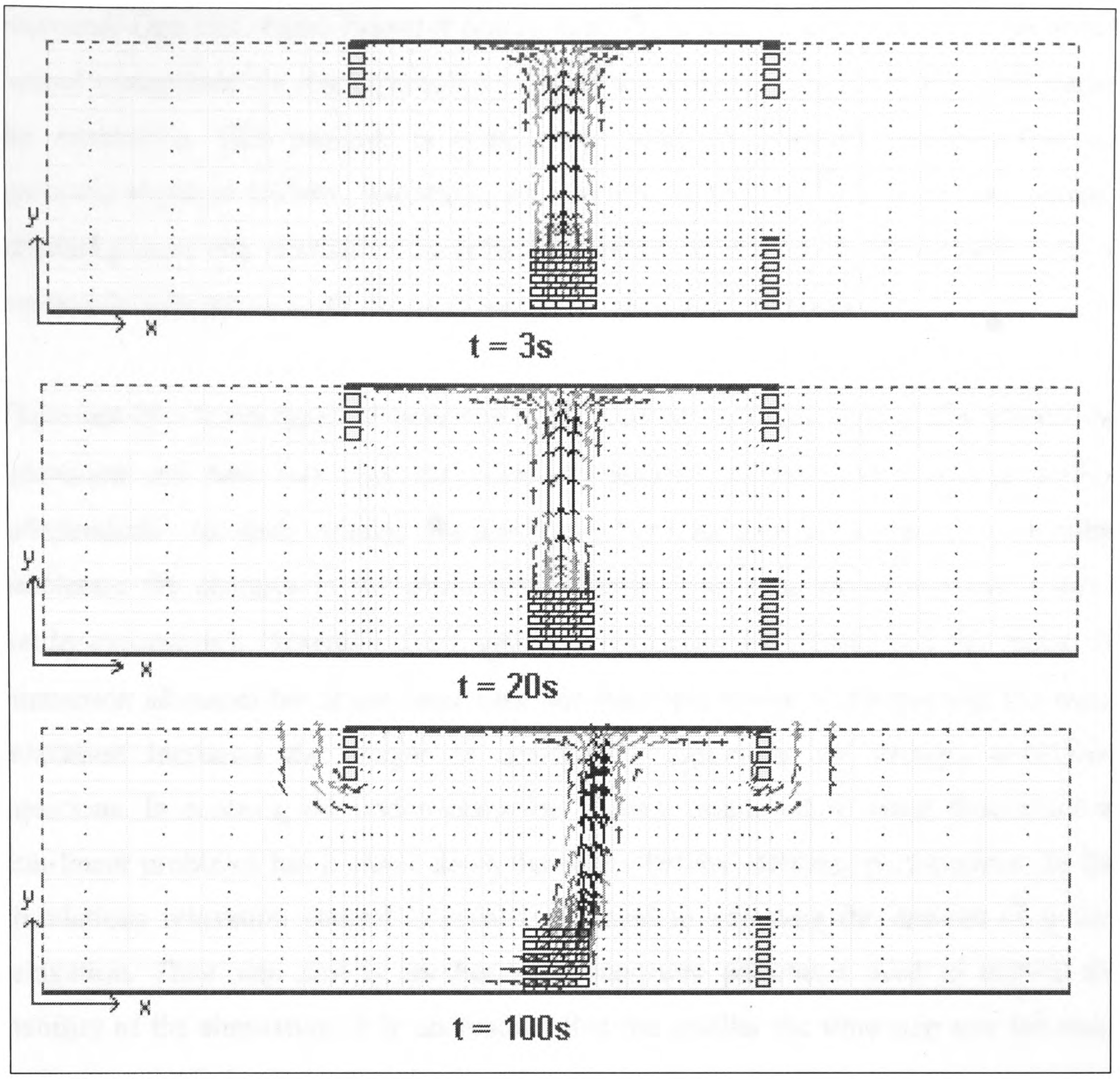


Figure 2-9 Velocity in the room at different points in time (2D section)

2.5.1 Controlling the solution process

One of the most important issues in numerical simulations is performance. Numerical modelling is computationally very expensive and requires fast computers with vast amounts of memory. Of course, new and more powerful computers help to mitigate this problem but it will never disappear completely. As more computing power becomes available more complex scenarios can be simulated, and since there is no practical limit to this complexity performance will always be an important factor.

Another common problem affecting the simulations is the possibility of divergence. The iterative solvers provide no guarantee that the correct results will eventually be produced. One can obtain incorrect results even for a very simple scenario if the initial control parameters are inappropriate and there is no attempt to rectify the error during the simulation. This problem is much more acute in complex scenarios that use advanced physical models. Sometimes the control parameters have to be continuously adjusted during the simulation to reflect changing conditions in the domain. This is commonly referred to as *the dynamic control of the solution process*.

There are two main types of parameters that can be modified during the simulation: *relaxation* and *time step size*. Relaxation is usually expressed as α and is defined independently for each variable. By modifying the relaxation parameter one can either accelerate the changes in the associated variable (*over-relaxation*) or slow it down (*under-relaxation*). Generally the bigger the relaxation coefficient (α) the faster the simulation advances but at the same time becomes less stable. Consequently, too much relaxation increases the danger of divergence, especially for strongly non-linear equations. In contrast, the under-relaxation is often employed to avoid divergence in non-linear problems but it slows down the solver hereby affecting performance. In fire simulations relaxation control is usually confined to adjusting the amount of under-relaxation. *Time step size* is another very important parameter used to control the stability of the simulation. It is understood that the smaller the time step size the more stable the simulation becomes. On the other hand experts also believe that a bigger time step provides better performance. Consequently, the actual time step size is usually a compromise between speed and stability. Of course in real simulations there are other factors that influence the choice of the time step size, e.g. if one requires the results at specific points in time or when very fast (or very slow) physical processes are being simulated.

Unfortunately, it is not fully understood how the control actions should be applied and experts often invent their own informal rules to assist them in modifying control parameters. These rules depend on the software used and the particular application domain. The general mode of operation of fire field modelling software is that the flow field and pressure fields are unknown at the start of the simulation. The heating due to the fire sources and consequent density changes lead to buoyancy forces that drive the flow. The difficulty with this technique is that the initial stages of a simulation are comparatively unstable and generally require significant under-relaxation to prevent instabilities from causing divergent solutions. However, although tight under-relaxation may be appropriate at the beginning of a simulation, the same parameters can have a detrimental effect on the quality or efficiency of the simulation in later stages where small changes compounded by excessive under-relaxation can falsely stagnate the solution.

The obvious solution is to apply significant under-relaxation at the start of the simulation and then, when the processing appears to have stabilised, to apply less under-relaxation for the remainder of the simulation. However, this technique is far from ideal because similar instabilities can occur later as particular flow features develop. Some flow features which can destabilise a solution are changes in orientation of fire plumes or ceiling jets, changes in height of the neutral plane and the creation or destruction of a re-circulation region within the flow field.

As the complexity of CFD software and modelling capabilities increase, there will be additional difficulties introduced by the temporal effects associated with more sophisticated behaviour such as flash-over, breaking windows, opening doors, secondary ignition and fire spread. None of these destabilising effects are handled by crude batch mode software without considerable manual intervention that is both tedious to apply and prone to errors. Ideally, automated intelligent agents are required to monitor the solution status and to make control decisions, based on the solution status, so that processing continues both optimally and in a stable manner. This dissertation concentrates on the development of the intelligent control agent capable of emulating the human ability to control the numerical solver and consequently the whole simulation process. It is believed that such an agent can substantially improve the performance and perhaps obtain more accurate results than are normally achieved in non-controlled simulation. Finally, fully automated control should make complex simulations easier to run and therefore be more accessible to non-CFD experts.

2.6 Summary of terminology

Several CFD-related terms are used in this dissertation and therefore this section contains a brief explanation of the terminology.

Convergence – a time step is converging if the residual errors are diminishing consistently and approaching required tolerance. The time step converges when all the residuals are below the specified tolerance (convergence condition).

Divergence – a time step/simulation is diverging if at least one residual (usually more) is steadily increasing or has been increasing and remains significantly higher than the required tolerance.

Mesh - a grid of points or a set of volumes, at which the relevant variables are calculated. Numerical methods can only calculate the results at finite number of discrete points in the domain and therefore require a mesh to define these points.

Numerical simulation – a method for modelling physical processes by iteratively solving a set of differential equations that govern these processes. Very expensive computationally and therefore normally performed on computers.

Relaxation parameters – special coefficients that control the convergence speed of iterative solvers. Reducing the relaxation stabilises the numerical solution while adding more relaxation speeds up the convergence.

Residual error – a measure of the accuracy of the current approximation. Usually calculated separately for each solved variable and defined as a difference between two consecutive approximations, averaged over all mesh cells.

Solved variables – physical quantities being calculated during the simulation, e.g. pressure, velocity, radiation, etc.

Steady state case – a simulation that starts with guessed values for variables ϕ and proceeds to obtain the values of ϕ at a point when the simulation reaches a steady state, i.e. all the flow properties stabilise and reach equilibrium.

Time step – a single stage in a transient simulation, which covers a short period of the simulated time. Transient simulations usually produce results at several discrete points in time.

Transient case – a simulation, which finds the value of ϕ at $\mathbf{t+\Delta t}$ based on the value of ϕ at time $\mathbf{t}$. This process is normally performed repeatedly to produce results for several time steps. Each time the results of a preceding time step are used as the initial guess for the next step.

Chapter 3

Literature review

3.1 Introduction

This chapter presents a brief overview of literature relevant to this project. By reviewing related publications we ensure that we were not repeating work that had already been done elsewhere. It also puts this project in the broader context and provides useful background information. Firstly, we concentrate on convergence acceleration techniques that are sometimes used in numerical software. We also explain a few other methods for improving the performance of CFD simulations that are not classified as convergence acceleration algorithms.

Secondly, we present a brief description and the history of SMARTFIRE (a fire modelling package) together with a prototype rule-based control system developed by John Ewer. SMARTFIRE was used as a testing vehicle for all versions of our control system while the results of Ewer's research served as a starting point of this investigation.

Finally, we present various projects that use heuristic search techniques to solve complex problems. The final version of the control system uses heuristic method and therefore it was deemed appropriate to include a brief description and the history of these techniques and explain how they are used to solve a wide range of problems.

3.2 Convergence acceleration techniques

Consistent advances in computer hardware over the last two decades, which seem to confirm Moore's Law (doubling of computational power every 18 months) led some to suggest that there is no need for sophisticated convergence acceleration algorithms in CFD software and that more effort should be directed towards developing better models incorporating additional physics. Unfortunately, the addition of new models usually results in a problem that is more difficult to solve and therefore, the simulation can actually take longer despite the availability of faster hardware. Furthermore, even with existing models there are still many cases that cannot be solved in a reasonable time and will remain unsolved for the foreseeable future. The following excerpt presents just one of many examples (Moin-97):

> Consider a transport airplane with a 50-meter-long fuselage and wings with a chord length (the distance from the leading to the trailing edge) of about five meters. If the craft is cruising at 250 meters per second at an altitude of 10,000 meters, about 10 quadrillion (10^{16}) grid points are required to simulate the turbulence near the surface with reasonable detail.
>
> What kind of computational demands does this number of points impose? A rough estimate, based on current algorithms and software, indicates that even with a supercomputer capable of performing a trillion (10^{12}) floating-point operations per second, it would take several thousand years to compute the flow for one second of flight time!

Currently, the turbulence can only be simulated accurately for very simple scenarios (like flow in a pipe) and even then computations have to be performed on massively parallel supercomputers. A popular alternative approach is to use approximate models, which are partially based on empirical data and average the small eddies which allows for a much coarser mesh and consequently shorter simulation time.

The need for more efficient algorithms becomes even more necessary when one seeks solutions to many intermediate pseudo steady state problems, i.e. “snapshots” of the solution state at different points in time. This is often the case in fire research where the details of fire development and spread are usually more useful to the researcher than the final steady state solution. In response to all these problems several different convergence acceleration techniques were developed over the last 30 years.

3.2.1 Preconditioning

One of the most established method for accelerating the solution of linear systems is preconditioning. The principal idea is to replace the original system of equations by the preconditioned set of equations that are easier to solve (remove 'stiffness'). Given a linear system:

$$\mathbf{A}\mathrm{x} = \mathrm{b}$$

one can apply a preconditioner **P**, to create the following system:

$$\mathbf{PA}\mathrm{x} = \mathbf{P}\mathrm{b}$$

P is a matrix that approximates $\mathbf{A}^{-1}$ but is easy to compute (unlike the $\mathbf{A}^{-1}$ matrix). Preconditioning is often used in conjunction with other convergence acceleration techniques (e.g. multigrid). A more detailed explanation of preconditioning techniques is beyond the scope of this thesis but there are numerous books and papers that describe this subject comprehensively (Choi-97, Lee-93 and Turkel-87). ILU preconditioners are one of the more commonly used In CFD applications (Zingg-97, Cai-97, Rausch-95 and Venkatakrishnan-93). An important type is a local preconditioner, i.e. a preconditioner that depends only on values at the current grid point with no influence from neighbouring grid point values (Lee-97, Ollivier-95, Morano-93 and Pierce-96).

3.2.2 Conjugate Gradient Methods

Another interesting algorithm for solving large linear systems is Conjugate-Gradient method, which is a substantial enhancement over the method of steepest descent. It was discovered independently by Hestenes (Hestenes-51) and Stiefel (Stiefel-52) and it was subsequently generalised to non-linear problems by Fletcher and Reeves (Fletcher-64). It would be difficult to provide a concise definition of this technique in this limited space and therefore for details one should refer to a comprehensive explanation provided by Shewchuk (Shewchuk-94).

3.2.3 Multigrid methods

Multigrid strategies are derived from computational methods but are generally considered as convergence acceleration techniques, rather than solution methods themselves. A multigrid strategy accelerates the solution of a set of fine grid equations by computing corrections on a coarser grid. This is based on the observation that the local variations in the solution are very quickly resolved by simple iterative methods on a fine grid. However, it is much more difficult and very inefficient to remove the global (low-frequency) errors on the same fine grid. Consequently, a multigrid method uses several, progressively coarser meshes (grids) to accelerate the elimination of the global errors. When the local errors are eliminated within the first few iterations on the fine grid there is a significant degradation in convergence rate. At that point the solution is transferred onto a coarser grid where some of the global errors in the fine mesh become local ones and are quickly resolved. The corrections computed on a coarse mesh are then interpolated back on the fine mesh. This method can be applied recursively using a set of progressively coarser grids. Mutigrid methods can be used with any existing relaxation technique and with both linear and non-linear equations. A comparison of multigrid against other convergence acceleration techniques is presented in (Mavriplis-98) while a detailed description of multigrid can be found in (Wesseling-92).

Mutligrid methods are currently one of the most popular convergence acceleration techniques. The first publication with multigrid algorithms appeared in 1964 (Fedorenko-64) but the real interest in these methods was started by independent research of Brandt (Brandt-73, Brandt-77) and Hackbush (Hackbush-76). They both published efficient and robust algorithms for multigrid methods and presented a sound theoretical analysis of this technique. Since then multigrid methods have received increasingly more attention and they have found their way into a number of different applications and numerical packages.

Zhang is one of the researchers that specialise in CFD applications of multigrid techniques. In his thesis (Zhang-97) he concentrated on the development of further improvements to standard multigrid methods with special emphasis on CFD applications. He developed efficient multigrid acceleration techniques that are particularly well suited to providing high accuracy numerical solutions in CFD. Some of the acceleration techniques have been shown to be essential for certain problems to converge. Zhang's techniques are easy to parallelise and do not require the coefficient matrix to be symmetric and positive making them easier to apply to a wide range of practical cases.

3.2.4 Fuzzy logic in relaxation adjustment

A system that uses fuzzy logic to adjust the relaxation parameters in CFD simulations was developed in Japan (Tatsuya-96). It was initially tested with heat conduction cases but subsequently the rules were improved to deal with fluid flow simulations. The system was embedded in PHEONICS and proved to provide stable convergence. Unfortunately, no further details are currently available as the paper was published in Japanese

3.2.5 Rule driven system for relaxation adjustment

A different approach to the convergence acceleration was proposed by Ewer (Ewer-98). Ewer developed a rule-based system for automatic solution control during the simulation. The system's main goal was to improve the convergence rate but it also employed a simple algorithm for divergence avoidance. The control decisions were based on automatic assessment of the most recent residual errors and involved small relaxation adjustments. The architecture was based on a set of rigid rules that governed the control actions.

Ewer presented an example (simple 2D transient case) where the system reduced the total number of iterations by 50%. He also demonstrated that the results were similar to the performance improvement obtained by a human expert controlling identical scenario interactively. No significant degradation in the accuracy of the final results was observed. However, Ewer stated that the control architecture did not scale very well and failed to provide similar improvements when applied to more complex 3-dimensional cases with larger heat output rate. He suggested that a more sophisticated control technique might be necessary for 3D cases due to the many degrees of freedom present in such scenarios.

Ewer's approach was adopted as a starting point of this work and therefore his system is explained with more detail in Chapter 4.

3.3 Other techniques for improving performance of CFD codes

3.3.1 Group solvers

One of many interesting concepts implemented within Smartifre is the use of group solvers (Ewer-00). This idea represents a natural enhancement to the standard JOR and SOR solvers (Pantakar-80). It focuses on the fact that during a standard simulation the computational effort is equally divided between all the cells. Therefore even the cells that are positioned far from the main flow receive the same attention from the solver as the ones in the most active region. If there are many such cells then a significant amount of computation time is not used towards advancing the solution. This problem is especially acute in fire research where complex geometries are often used in simulation of fire spread (e.g. multi-storey buildings). Inevitably, a large part of the domain remains relatively inactive throughout most of the simulation.

Group solvers provide a way to partition the domain into regions with different levels of activity and then perform an increased number of iterations in active regions than in other areas. Ewer proposed two different types of partitioning:

- **Static**, where the cell membership to the particular group is determined at the beginning of the simulation and does not change. This technique is useful in directing the computational effort away from non-important regions like sealed rooms or cells very far away from the heat source.
- **Dynamic,** where the cell membership is constantly verified during the simulation and can change dynamically. The membership criteria can be very flexible. In one of Ewer's examples the cells with absolute velocity less than 10% of the current maximum domain velocity are configured as "Calm" group while all other cells are classified as "Active". Each group has a different number of internal iterations assigned (the more active the group the more iterations are performed).

Both static and dynamic groups can be used in the same simulation. In his paper Ewer (Ewer-99c) presents an example where the use of group solvers reduced the overall simulation time by 37%. It must be noted that this technique does not explicitly accelerate convergence but improves the performance by reducing the computational cost of calculating solution for non-essential regions.

3.3.2 KBS-based mesh generation

Although the mesh generation is performed before the CFD simulation is even started it is still one of the most important issues in CFD modelling as poor mesh quality has detrimental effect on the simulation results and can also severely impair the performance (in extreme cases making it impossible to obtain the correct solution). The area of automated mesh generation was researched by Taylor (Taylor-97a) who created an expert system for mesh generation and integrated it within the SMARTFIRE package.

A human expert constructs a grid by first analysing the layout and physical properties of the domain. The expert would normally create a fine mesh where significant changes are expected (vents, heat source, plume area) while using bigger cells (coarser mesh) in the areas that are considered less relevant (e.g. distant from the main flow). Creating a good mesh requires considerable experience and therefore presents a serious obstacle for novice users of CFD applications.

The automated mesh generation proposed by Taylor relies on Case Based Reasoning augmented by a rule-based system. The system maintains a database of various CFD cases with corresponding meshes (created initially by human experts). During a typical mesh generation session, the best-matching case from the library is identified and retrieved. In the next step, the retrieved mesh is adapted to account for differences between the new problem and the library case. The modified mesh is then presented to a rule-driven system, which validates the mesh against the set of meshing principles (the rules are static and were obtained during the knowledge elicitation process) and further modifications are then performed (so called 'repair phase'). During the 'repair phase' the mesh is only adjusted by a small amount and therefore several iterations are usually required until all the rules are satisfied. The final solution can be added to the existing case library to be used in the reasoning process for subsequent cases. Taylor's system has been integrated into SMARTFIRE as one of several tools designed to make CFD simulations more accessible and easier to use by non-experts (Taylor-96).

3.3.3 Latency tolerant algorithms and parallel computers

Recent advances in computer hardware resulted in powerful parallel computers becoming almost commonplace and by 2004 it is expected that teraflops computers (machines capable of performing 10^{12} floating point operation per second) will become accessible to small group of users (Keyes-97). The next milestone will be a petaflops system (10^{15} op/sec). Such powerful computers consist of many processors (the petaflops systems are expected to have between 10^4 to 10^6 processors) with deep memory structures and therefore require specialised algorithms to take advantage of their computational power. The main issues are inter-processor synchronisation and memory latency (latency is the ratio of time required to fetch a variable from memory versus the time required for a floating-point operation). As the processor speeds have rapidly increased, the memory access has not improved at the same rate and consequently all modern processors use multi-level caching in order to alleviate this problem. However, since most of the CFD simulation operate on large sets of data, the caching strategy is not as effective as in other applications. This has given rise to specialised algorithms that use cache-friendly strategies like data re-use and increased locality. Such algorithms were demonstrated to double or even triple the computational throughput. There are many different techniques used in the development of latency tolerant codes. Some of them concentrate on data re-ordering to improve locality (Cuthill-69, Lohner-97) while others propose special mesh partitioning strategies to minimise inter-process communication to improve the speed of parallel processing. There also exist dedicated latency-tolerant solution algorithms, like multigrid and Newton-Krylow-Schwarz solver (Cai-97).

Most of the latency-tolerant algorithms are still in their infancy but one can expect that in the future more research will be directed towards the development of such algorithms to take full advantage of the computational power offered by teraflops and subsequently petaflops systems.

3.4 SMARTFIRE project

SMARTFIRE is a CFD package developed by Greenwich University, written in C++ and based on numerical methods re-engineered from a legacy Fortran code (also developed in Greenwich (Ewer-00)). SMARTFIRE is a dedicated fire-modelling application that aims to make CFD simulations more accessible to non-experts through the use of an intuitive window-based Graphical User Interface (GUI) augmented by expert systems to guide and assist a novice user throughout the whole modelling process, starting with problem specification and ending on the visualisation of the results. SMARTFIRE uses a 3D unstructured mesh and can solve turbulent or laminar flow problems under transient or steady state conditions. The first version of the system could only model the fire as a user-defined volumetric heat source but it was subsequently enhanced by the addition of the combustion model (Jia-99). Since SMARTFIRE is written in an object-oriented programming language and was designed to form open software architecture, it is a very convenient platform for other CFD-related research. As a result, it plays an essential part in several research projects within Greenwich University (Wang-99, Ewer-98, Taylor-97a).

One of its unique features is the user interface. Unlike many traditional CFD codes which tend to run in a batch-mode, SMARTFIRE is fully interactive and allows the user to observe the developing solution (thanks to the advanced visualisation capabilities), perform on-the-fly modifications and other control actions. Various diagnostic outputs can be monitored and used by the experts for fine-tuning the simulation process.

When the users gained more experience using the new capabilities offered by SMARTFIRE, it became apparent that the performance of the CFD simulation can be significantly improved by real-time adjustments to control parameters (mainly relaxation coefficients and time step size). Ewer presents an example (Ewer-98) where an experienced user managed to reduce the simulation time by 50% by performing small adjustments throughout the whole run. There was, however, one major problem associated with this acceleration technique – it required the expert to continuously monitor the simulation and perform occasional adjustments in order to obtain significant performance benefits. This was obviously not a practical approach (especially for big simulations that could run for days) and therefore further research (Ewer-98) was conducted to determine whether it was feasible to create an automated system capable of emulating control actions performed by human experts. Ewer developed a prototype rule-driven system and showed that it was able to provide significant performance gains on a simple 2-dimensional fire case. This thesis is a sequel to his research into Automated Solution Control although the rule-driven approach was subsequently abandoned in favour of more advanced AI techniques.

3.5 Heuristic search

The convergence acceleration and simulation control methods described in this thesis are based on a heuristic search paradigm and therefore a short overview of heuristics and some of the applications into robot navigation are presented here.

'Heuristics are criteria, methods, or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal. They represent compromises between two requirements: the need to make such criteria simple and, at the same time, the desire to see them discriminate correctly between good and bad choices' (from Pearl-84). Heuristics are commonly referred to as "rules of thumb", i.e. a set of rules that are effective most of the time but not every time. They are normally used when the complexity of the problem is too great to perform a full analysis to derive a definitive solution method. Initially the heuristic approach was mainly used for game playing and puzzles. The Travelling Salesman Problem (TSP) is a classical example that is best solved using heuristics. For TSP, one has to find the cheapest path that visits every node once and only once, returning to the initial node in a graph of N nodes with each edge assigned a non-negative cost. TSP is an NP-hard problem, i.e. all known algorithms require exponential time to solve it in the worst case. TSP is surveyed in (Lawler-85) while one of the most popular heuristics that is being used for solving TSP was proposed by Edmonds and Karp (Karp-72). Other classic problems that are commonly encountered in theoretical AI research are the 8-Queens problem (arranging 8 queens on a chessboard so they don't attack each other (Floyd-67) or n-puzzle (finding a sequence of moves that will arrange n-1 pieces in the predetermined order on a n-field board (Loyd-59, Michie-66)).

As AI expanded substantially over the last three decades, the heuristic techniques have gone beyond theoretical analysis of confined problems and are now used in real-life applications (robot navigation, handwriting and speech recognition, biometrics, etc.). One of the most popular applications of heuristics is robot navigation and, as it has striking similarities to our control problem, a few examples of heuristic navigation systems are presented here.

Elnagar (Elnagar-95) presents a set of heuristics designed to control a free flying robot in a 3D environment. The robot's task is to reach to the goal navigating around the obstacles and staying (within some margin) at the required altitude. Consequently, the basic evaluation function is the sum of obstacle repulsion, goal attraction and level attraction. Elnagar describes two additional heuristics designed to improve the search efficiency and to overcome the local minima problem. In (Autere-97) the authors present a motion planning system for an autonomous robot. They developed admissible heuristics that are computed by solving the planning problem in a simplified space. The efficiency of the heuristic was compared against a simple Manhattan distance heuristic using three cases with varying degrees of freedom and proved to be 10 to 100 times more efficient than the latter.

Many researchers concentrate on applications that could potentially be commonly used in everyday life, like control systems that may in the future lead to a "driver-less" car. Fiorini (Fiorini-98) presents a motion planning system in a dynamic environment (where the obstacles move). The system consists of a heuristic module for real-time trajectory generation and a collision avoidance module that computes a set of feasible avoidance manoeuvres at regular time intervals. The system was tested as a control module in an autonomous moving vehicle. Hiraishi (Hiraishi-98) developed a heuristic navigation system designed to work in a time-constrained environment. Such system is well suited for real time route finding in automobile navigation where the control decisions must be made within a short period of time.

An interesting problem is researched by Koenig (Koenig-98). He describes a motion planning system for a maze where the robot knows the maze layout but does not know its location or orientation in the maze. The robot navigates by interleaving planning and plan execution which allows it to gather information early. The planning is guided by a real time heuristic search.

3.6 Summary

This chapter presents the research projects related to the work described in this thesis. It gives an overview of various convergence acceleration techniques and other methods aimed at improving the speed and robustness of linear methods, with special emphasis on CFD simulations. There is a brief description of various research projects that were conducted within the SMARTIFRE group (with special attention given to the rule-driven control system designed by Ewer). Finally, an overview of heuristics is presented together with several examples where the heuristic search is used in robot navigation.

Chapter 4

Initial work on a rule-based control system

4.1 Overview

Chapter 2 presented a general overview of the fire modelling techniques and described all the generic stages that are required in a successful fire simulation. Each of those stages was then further explained using a simple real case and a real fire modelling software. In this chapter we will present more details about the software that was used to generate that initial example (SMARTFIRE). A simple rule-based control system implemented in SMARTFIRE is described and analysed. We follow the development of its successor, KBS 2.0, and explain the reasons responsible for the failure of both systems. Finally, this chapter reveals why this research moved away from a simple rule-based approach and describes the experiments that were performed to enhance our understanding of control actions to aid the search for alternative control techniques.

4.2 SMARTFIRE – An interactive CFD software

In the recent years the numerical packages started to move away from the "batch-processing", which was prevalent in their early days. It became apparent that the complexity of the scenarios that are simulated nowadays requires a high degree of control in order to obtain correct results in feasible time. A modern interactive numerical engine allows an expert to monitor the current simulation state, detect developing problems and modify the control parameters as necessary. One of the examples of such interactive code is SMARTFIRE, which 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 parameters. Consequently, an expert can substantially reduce the execution time and improve the accuracy of the results by continuously watching the progress of the simulation making necessary adjustments when required. SMARTFIRE is very flexible and easy to use –

the user is presented with an advanced GUI and can start the simulation at a press of a button, in which case all the control parameters take their default values. However, for many scenarios the default parameters are not appropriate and can seriously affect the accuracy of the final results and/or time performance. In the worst case it may be impossible to obtain the final results as the simulation diverges or progresses so slowly that a solution would never be achieved.

Nevertheless, the interactive CFD code remains a very useful tool for experts. But even they cannot fully utilise its potential. The main source of problems is the time factor, as typical CFD simulations take several hours to complete, whilst in some cases this time is extended to days. It is unreasonable to expect that any expert is ready to spend a few days in front of the computer diligently adjusting control parameters and correcting problems as they develop. In fact, even intermittent control is virtually impossible, since it still requires substantial amount of time and effort to investigate various possibilities and assess the results. Of course, if someone is prepared to wait longer for the results then there is often no need to make any aggressive performance-oriented modifications. However, for unstable cases, the control actions are necessary since the only alternative is continual restarting of the whole case with different sets of control parameters until a correct run is achieved.

Unstable scenarios lead to further problems in numerical simulations – it is very difficult to guarantee that every time step fully converges. The convergence is usually defined as the point where all errors fall below the specified tolerance. It is believed that the full convergence of all time steps guarantees the correctness of the final results. However, due to the complexity of convergence assurance experts usually settle for the 'most of the time steps converged or almost converged' solution. Special care must be taken to ensure that there were no diverged time steps as in such case final results can be inaccurate.

As a partial solution to the problems detailed above, most of the interactive numerical engines give some feedback on the internal state of a solution (FLOW3D FLUENT, STARCD). Some codes allow 'bookmarks' to be saved, which can then be reverted to, in case any of the subsequent time steps diverge. All this functionality greatly enhances the productivity and accuracy of numerical simulations. But the main and the most important problem still remains – in order to obtain the maximum benefits the simulation has to be constantly supervised and controlled by a skilled operator. Considering the length of a typical simulation, full control by human experts is virtually impossible. Furthermore, efficient control requires expert knowledge and therefore a novice user is not able to control a CFD simulation properly.

4.3 Ewer's rule-based control system

The first attempt to address the control problem in SMARTFIRE was made by Ewer (Ewer-98). Ewer implemented a simple system for dynamic control that was based on his own experience with running CFD simulations. Ewer's system used a rule-driven approach where the control actions were limited to small relaxation adjustments while decisions were made using a basic state-recognition algorithm. The rules were fairly simple and the control decisions were based on a limited amount of information.

The system tracked the residual errors and used them as the indicator of the simulation state. The local trends in residual errors were examined and assessed. The assessment was very simple as it compared only the gradients from the last three residual errors. The gradients determined if the particular residual was classified as converging or diverging. Depending on the result of the assessment, the relaxation was either increased or reduced. The variables were grouped depending on their interdependencies and relative importance. For instance, PRESSURE and VELOCITIES were assessed together and the relaxation changes were always applied to the whole group of variables at the same time.

Ewer designed his control system to allow for at least 10 sweeps between the neighbouring control actions. He found that the control actions performed during the time step introduced a local instability (commonly known as a "kick") which required some time to die away. Applying changes too frequently can accumulate the adverse effects of several modifications and destabilise the simulation (potentially beyond recovery). Ewer claimed that the effect of the kick usually disappeared after 5 sweeps and therefore he decided to impose a minimum gap of 10 sweeps between two adjacent control actions but used an even bigger number (20) in his dynamic control example. This value effectively determines the frequency of the control actions.

The control changes were small since Ewer believed that they were less likely to cause serious instability. Consequently, Ewer adopted a "little but often" technique as a basis of his control strategy. This decision might have been influenced by the lack of reliable divergence detection and recovery method hence a "divergence-avoidance" approach. On the other hand, Ewer did implement a simple divergence recovery policy: local trend analysis was used to detect major convergence problems and then, if any were detected, the hard-coded "safe" set of control parameters was applied. This approach, however, did not guarantee divergence recovery and it usually incurred a substantial performance penalty. Consequently, any problems with convergence had to be avoided, which was partially achieved by using only small adjustments.

The performance of the control system was demonstrated using a simple 2D case with a small fire and a partition, which is removed after the initial 30s. The results showed that Ewer's system reduced the overall number of sweeps by 50% when compared with a non-controlled simulation using default (safe) settings. The reduction was similar to the performance improvements obtained for the same case by a human expert. The results were very encouraging but unfortunately the system did not scale very well and failed to control 3D scenarios effectively. Ewer attributed these problems to more degrees of freedom and higher complexity of 3-dimensional cases. Nevertheless, he did demonstrate that considerable savings in run time could be achieved by correct and efficient control actions. It was also confirmed that the reduction in the number of sweeps did not affect the accuracy of the results. Ewer also acknowledged that further research was necessary before an automated control system could provide tangible improvements in a broad range of complex cases.

A brief summary of the benefits and limitations of Ewer's approach is presented below.

Ewer's major contributions:

- The research showed that an automated system was capable of reducing the simulation time and improving the solution stability.

- Ewer listed the problems that he encountered during the design of the control system and proposed solutions to overcome some of them.
- Ewer believed that the residual errors were the main indicators of the simulation state and that most of the information relevant to the control decisions could be retrieved from the residual graphs.
- Adverse effects of applying the changes during the time step ("kick") were documented and explained.

Main limitations of the system:

- The control architecture was not very sophisticated and used informal knowledge obtained by its author while experimenting with SMARTFIRE.
- Solution monitoring was limited to a few discrete points during the time step.
- Control decisions were based on local information extracted from the residual graph. There was no attempt to examine the whole graph or assess a full time step in the control process.
- There was no real attempt to perform any control actions between time steps. A single rule was used every time a new time steps was started: "if the last time step converged then use its control parameters; if not – use the pre-determined safe set of relaxation parameters".
- The system did not adjust the time step size. The time step size is believed to be another major factor determining the performance and stability of the simulation.
- The system did not guarantee full convergence and was focused exclusively on improving the convergence rate.

Despite problems with 3-dimensional cases the results obtained by Ewer showed that there was a potential for substantial performance improvements. Ewer's system was able to provide a reduction in execution time and autonomously control a simple CFD case. Problems with complex scenarios did not undermine its achievements, as it was only a prototype designed to test the feasibility of automated control. The author acknowledged its shortcomings and indicated the areas for improvement. It was believed that further research would lead to a more reliable version of the system, capable of controlling complex cases successfully.

4.4 Knowledge acquisition

The initial research focused on identifying deficiencies of Ewer's approach and developing advanced algorithms for better and more comprehensive assessment procedures. It must be noted that the fundamental principles of Ewer's approach remained intact – the control actions were still limited to relaxation adjustments applied repeatedly within the time step simulation. At that time such approach seemed rational, as there was insufficient knowledge available to propose an alternative architecture. A detailed analysis of Ewer's system and its set of rules served as a starting point in the process of identifying areas for improvements. Consequently, the first stage of the research was focused on knowledge acquisition and involved several interviews with three different experts. The goal was to formalise the current knowledge and document the control techniques used by experts in their work.

4.4.1 Consultations with experts

The knowledge acquisition consisted of a series of interviews with four different experts that were familiar with SMARTFIRE and had some knowledge of other numerical packages. Each expert was asked to describe his own control technique and then was asked several questions regarding various issues related to the control problem. The following main topics were investigated during the interviews:

- Definitions of convergence and divergence.
- Variable priorities (experts were asked to classify the solved variables according to their importance).
- Influence on their control decisions of different features in the residual graphs.
- Description and identification of different stages during the simulation.

The experts were also encouraged to add any comments they felt might be important. In the final stage of the interview, each expert was shown a set of residual graphs and asked to classify them as good or bad and then suggest appropriate control actions.

4.4.2 Interview results

The interviews failed to uncover a consistent set of rules for controlling a CFD simulation. Each expert seemed to be using a slightly different technique. One preferred to adjust the time step size only and used the default set of relaxation parameters in most simulations. Another one usually modified only relaxation parameters for PRESSURE and VELOCITY. The third expert analysed the residual graphs for ENTHALPY variable to determine whether the time step size should be reduced (it was reduced when ENTHALPY was diverging). Another important observation was that the experts were usually focused on divergence recovery and had little knowledge of performance-oriented modifications. As a result there was insufficient expertise available regarding the effects of the control actions and their suitability in particular cases.

There was good agreement about the convergence and divergence definitions. All experts stated that if all residual errors dropped below the tolerance level then the time step was believed to have converged. One expert added that the mass error could be an additional way to confirm the convergence. Furthermore, the divergence was defined as residual errors consistently increasing although sometimes it might be difficult to distinguish between short-term convergence problems and real divergence. The experts confirmed that the residual graphs were the main indicator of the simulation state but stressed that their control decisions were also based on the analysis of the physical processes happening within the simulated domain

All experts agreed that PRESSURE and VELOCITY were the driving forces of the simulation but there were different opinions about the relative importance of other variables. However, in most cases ENTHALPY, KINETIC_ENERGY and TEMPERATURE were also classified as important.

When the experts were presented with several example residual graphs, their assessment results (good/bad) were very similar. They agreed that the convergence rate was the most important factor but added that other features (i.e. graph smoothness, presence of oscillations) also play a part in graph assessment. However, the experts found it very difficult to recommend any control actions based on the information only available in residual graphs.

The experts stressed that the full convergence of all time steps guaranteed the accuracy of the final results. One of the experts added that the change in heat release rate was normally the most important factor determining the stability of the simulation. A big change in heat output within a single time step may result in divergence and therefore should be avoided (usually by reducing the time step size).

4.4.3 Analysis and interpretation

The interviews confirmed that the residuals were the main indicator of the simulation state but the experts also analyse residuals from previous steps and the physical conditions in the domain when making any modifications to the control parameters. The convergence rate was deemed the most important property of the residual graph. Experts also mentioned the smoothness of the residual graph and presence of outliers and oscillations as other significant features. They strongly emphasised the importance of full convergence of all time steps and its effect on accuracy.

The knowledge elicitation did not provide as much information as it was hoped for. The experts had limited knowledge about the effects of different adjustments and the suitability of particular actions to specific problems. It transpired that the performance-oriented modifications were performed very rarely. The experts had more experience with divergence but their standard recovery procedure was rather simple and involved reducing the time step size.

Unfortunately, the interviews did not uncover enough knowledge to make it possible to design a completely new control system. However, the new information helped introduce significant improvements to Ewer's control technique. It was hoped that additional research would lead to better understanding of the control problem and therefore result in further improvements.

4.5 New rule-driven control system (KBS ver 2.0)

Although the experts did not provide as much information as it was hoped for, some of their observations together with the conclusions drawn from the failure of the previous system gave rise to major enhancements to the original rule-driven approach. However, partially due to the limited success of the knowledge elicitation, the main principles of the original design remained largely intact:

- Control actions comprise of small relaxation adjustments applied during the time step.
- The adjustments are governed by a rule-driven system, which relies on a custom assessment algorithm to extract relevant data from the residual graphs.
- The residual graphs remain the exclusive source of information about the current state of the simulation and the quality of the solution.

The new system (KBS ver 2.0) contained many improvements that were believed to be able to provide tangible performance benefits and allow efficient control of 3D scenarios. The information obtained during the interviews with experts was used in the development of a new assessment algorithm for residual graphs. The new algorithm was able to extract the features that the experts deemed relevant to the control process. The assessment procedure consisted of several stages and various measures (remaining iterations to convergence, divergence detection, smoothness, average gradient, etc) were constructed to describe the quality of a particular graph.

Furthermore, a special state based approach was implemented for efficient scheduling of the control actions. Four different control phases in a single time step were identified and different control rules developed for each of them. The following sections contain a more detailed description of the major improvements.

4.5.1 Residual graph assessment

One of the very obvious problems in the original system was the inadequate trend assessment algorithm. Since only the last three values were used to determine the trend, the results were not just inaccurate but very unreliable. To address this problem, the newly developed assessment algorithm was designed to use substantially more data to perform a detailed and comprehensive analysis. During the first assessment stage, the graph was classified as 'mainly up', 'mainly down' or 'unstable'. This method was modelled on the initial graph assessment performed by human experts. Depending on the classification result, different techniques were used for further analysis. These procedures have now been superseded in the further work described in this thesis and therefore only a brief description of the assessment algorithm for a 'mainly down' graph will be presented here as an example.

If a graph is classified as 'mainly down' then the main purpose of the control action is to speed up the convergence since a downward trend indicates that there is no need for divergence recovery. Small fluctuations are irrelevant for the purpose of trend assessment and therefore the graph is usually smoothed (Ott-93) before any further analysis. Smoothing is useful as it eliminates most minor variations (always present in residual graphs) and a graph is produced with a more consistent trend that is easier to analyse. If necessary (e.g. many trend variations persisting after the initial processing), a more aggressive smoothing process can be subsequently applied. The resulting graph is then analysed in order to extract more information about the trend. If after smoothing the graph still does not exhibit a consistent downward trend then a specially developed algorithm (adapted from the computer graphics domain, see Earnshaw-85, Le Riche-69 and Reuman-74) approximates the graph with a set of lines. The resulting set of gradients is used to further analyse the trend. More examples of how the assessment procedure works are presented in Figure 4-1. It is important to note that at this point the relaxation changes are only recommended and not applied. Another set of rules is used in the next stage to analyse the recommendations from all the variables to decide what modifications would eventually be applied.

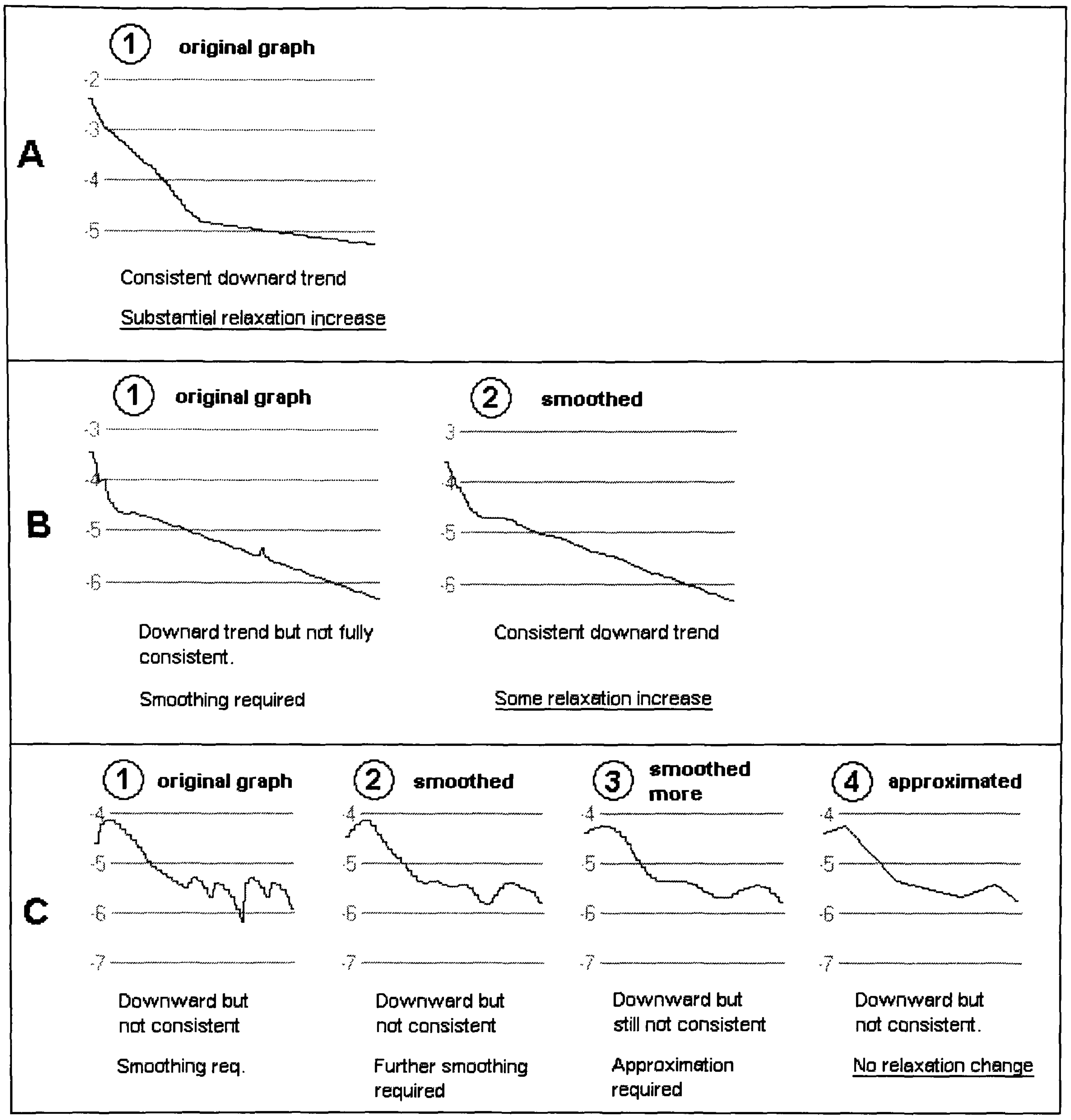


Figure 4-1 Example of assessment procedure for three different 'mainly down' residual graphs (for a single time step only)

The assessment algorithms were constantly improved, as new data emerged during the testing of the new control system. This resulted in the development of several interesting techniques for graph analysis, which proved to be very useful in further stages of the research.

4.5.2 Intelligent scheduling of control actions

Another improvement was the introduction of more sophisticated scheduling of the control actions. Ewer's system accessed the simulation every 20 sweeps while additional rules restricted the changes during special periods, i.e. the control actions could not be performed within the first few sweeps, for some period after adjustments and in the final part of the time step. The next version of the KBS adopted these principles but implemented more advanced methods for state identification.

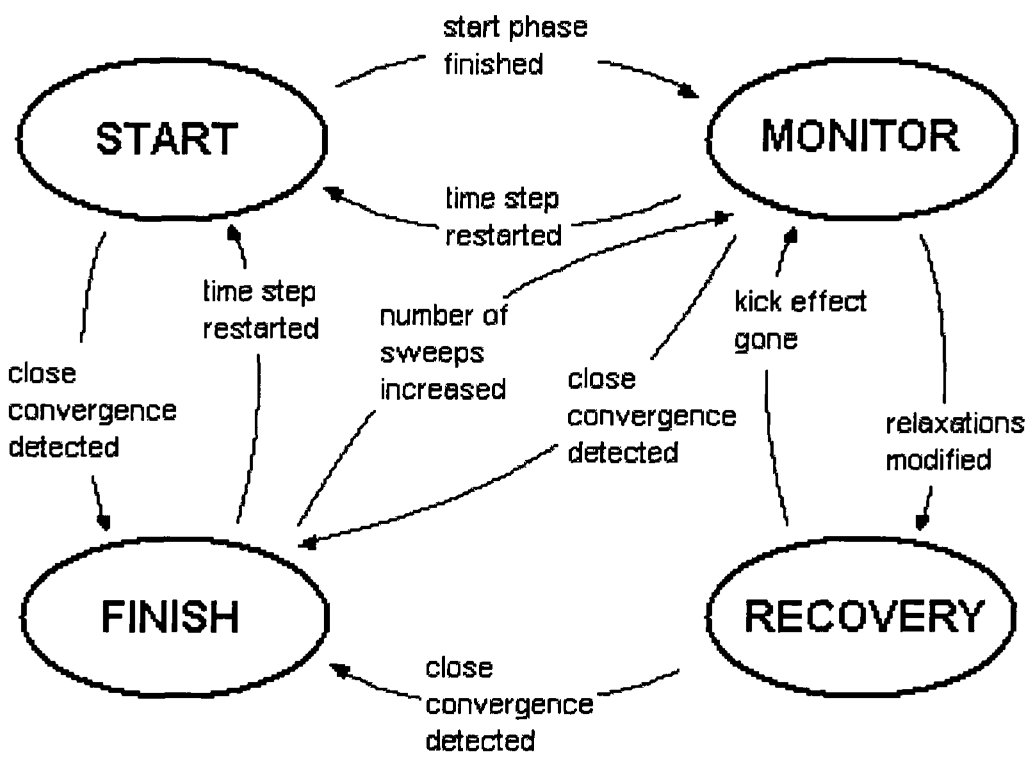


Figure 4-2 Different stages in the control system

Figure 4-2 shows all four states that can be attained within a single time step together with a brief description of the events that cause the state transition. Each stage represents a different part of the time step processing and requires significantly different control actions:

START – the solution process is in this stage during the first few sweeps. Substantial errors are expected and the residual graphs often appear unstable. Experts believe that no control actions should be applied at this stage since the residual trends are not fully developed yet and therefore do not provide sufficient information to make correct decisions. Normally, the starting phase finishes after 5-10 sweeps.

MONITOR – this state represents the 'middle' part of the time step, which is the most interesting stage of the simulation. Virtually all performance-oriented control actions are performed in this state. The relevant residual graphs are analysed and then appropriate changes introduced (if required). If any adjustments are made, the system enters a new state: RECOVERY.

RECOVERY – the recovery stage is similar to the starting period. As the name suggests, its main purpose is to allow the numerical engine to recover from a local instability introduced by the most recent changes. When the 'kick' effect dies out, the system returns to the MONITOR state.

FINISH – In the final phase of the time step no modifications are allowed. When it is determined that residuals are close to reaching convergence then no performance-driven control actions should be applied as the associated 'kick' may artificially increase the residual values and therefore defer the convergence.

Consequently, at each point of the simulation, KBS 2.0 operates in one of the four different states, which determine what control actions are allowed. The system closely monitors the simulation progress to identify the transition points between the stages. Most of the transitions are relatively straightforward to detect but some require non-trivial algorithms. For instance, the length of the START phase can be arbitrarily defined as the first 10 sweeps of the time step whereas the identification of the FINISH state is more complicated. If it is agreed that the FINISH phase occurs during the last 15 sweeps before convergence, then one must be able to detect a point in the time step simulation where there are only 15 sweeps remaining to full convergence. In order to solve this problem, a special algorithm for predicting the convergence point had to be developed. Fortunately, when the residual graphs are presented in a logarithmic scale then a simple method of the least-squares approximation gives acceptable results and is therefore used for convergence prediction. Obviously, the accuracy of the prediction varies but this fact does not invalidate its benefits. The convergence prediction method was substantially improved in the later stages of this research (see Chapter 5 and 6)

4.5.3 Discussion of initial test results

The new system incorporated many improvements that Ewer identified as essential. As a result, the control procedures became more predictable and more robust.

Unfortunately, KBS 2.0 was still unable to control 3-dimensional cases effectively. The results obtained during tests were not consistent, in some cases the new KBS reduced the number of sweeps, in others the total number of sweeps actually increased. To make matters worse, even in the best cases the observed performance improvement was lower than 7%. The lack of consistent reduction in simulation time was an early warning sign since the implemented enhancements were expected to provide substantial performance benefits. Contrary to the initial expectations, 3D cases remained very difficult to control. The analysis of the simulation progress and the automated control process did not reveal any obvious areas that could benefit from further refinements to Ewer's approach. These problems prompted a suggestion that perhaps the underlying architecture was inappropriate and did not model the human control actions correctly. Since there was no obvious solution that would promise to overcome the difficulties encountered, it became clear that a more thorough analysis of the control problem was necessary before new control architecture could be proposed.

4.6 KBS 2.0 - Analysis

The initial tests clearly demonstrated that the new control system did not provide any significant reduction in the number of sweeps and its behaviour was inconsistent and inefficient. Since most of the proposed improvements had been implemented, it became apparent that only radical changes to the system architecture might be able to produce the required performance gains. Consequently, it was decided to conduct a thorough analysis of the test results in order to establish the factors responsible for the failure of the first two systems (Ewer's and KBS 2.0, based on Ewer's approach) and perhaps devise a different architecture for the next generation of the control system. The research was therefore once again focused on the identification of the deficiencies in the rule-driven control systems.

4.6.1 Adjustments introduce instability

The first identified source of the problems was the "kick effect" occurring immediately after the changes were applied. After analysing several residual graphs, it was revealed that often the adjustments did not seem to speed up the convergence rate while the introduced instability significantly delayed the convergence. Even in the graphs where the increase in relaxation did result in a better convergence rate, the benefits were often nullified by the kick. This problem is illustrated in Figure 4-3.

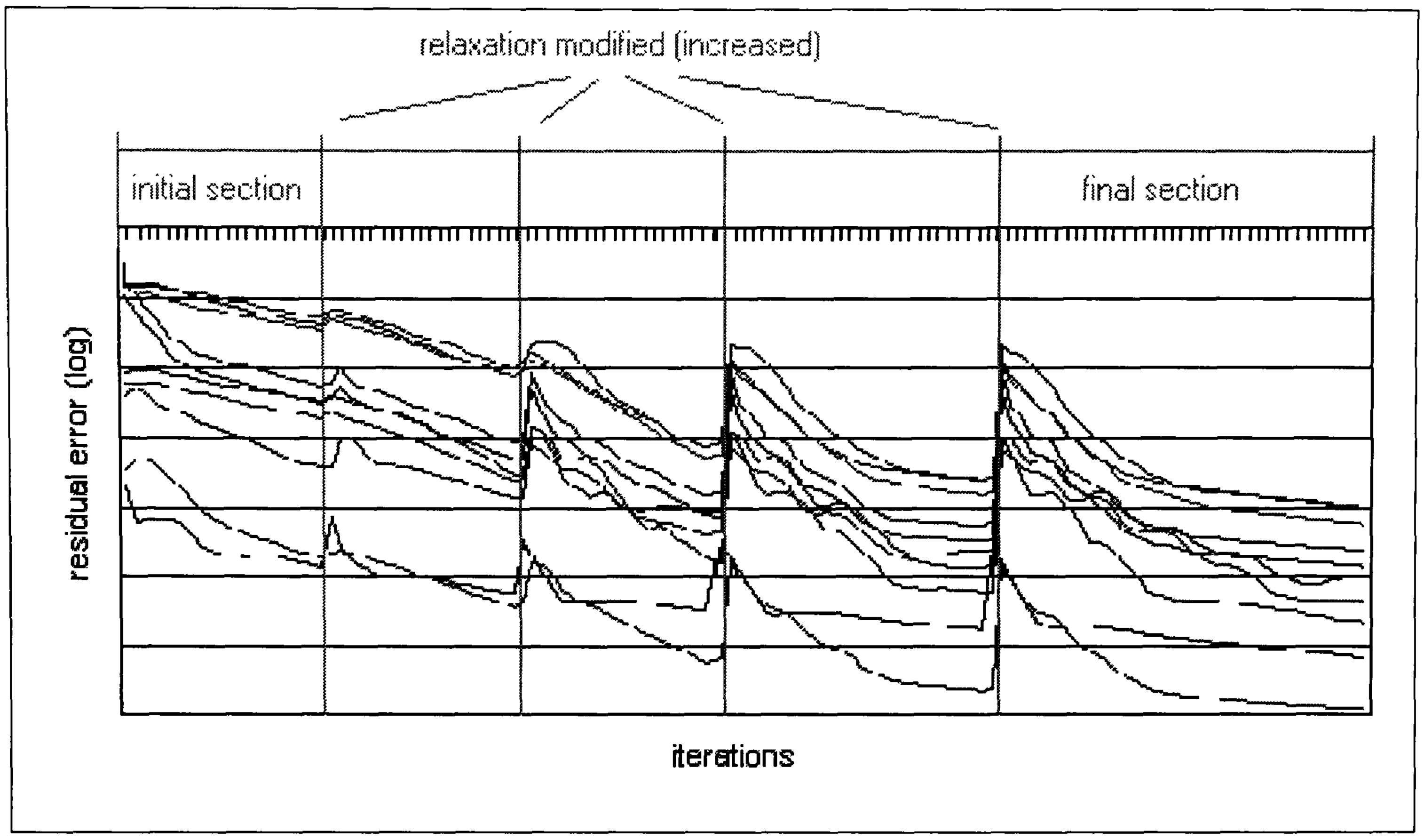


Figure 4-3 The "kick effect" in residual graphs

When the relaxation parameters are modified during the time step then the residual errors change abruptly (producing the "kick effect"). The magnitude and direction (sharp increase or decrease) of the "kick" vary and depend on a number of factors. The important conclusion is that the resulting instability in the residual values can delay the convergence, even if the purpose of the changes was to speed it up. For example, in Figure 4-3 the last modification creates a sharp increase in residual values, which cancels most of the increase in speed that was gained. This problem is compounded by the fact that the control architecture relies on frequent but small adjustments, which result in the introduction of several instabilities within a single time step. In order to minimise the adverse effect on performance it might be more efficient to make bigger changes but less frequently. However, bigger adjustments require sophisticated analysis of the current simulation state, as they are more likely to cause serious instability and subsequent divergence. Perhaps the changes should be applied between the time steps, as the "kick effect" would be merged with the instability caused by changing the current simulation time. This approach avoids introducing local instabilities during a time step but it requires further research.

4.6.2 Not enough information for control decisions

Another major flaw identified in the design of the first two control systems was the fact that the control decisions were based on very limited amount of information. Ewer's system used only three most recent residual values as the indicator of the current trend. This was clearly unsatisfactory and several attempts to improve the assessment procedure were made. The number of points analysed was substantially increased (to at least 20). Other factors like trend consistency and the change in convergence speed were also considered in the set of control rules. Nevertheless, the results suggest that the amount of information extracted from the residual graphs might have been insufficient to make purposeful control actions.

In the search for a better feature extraction procedure, the experts' assessment techniques were once again put under scrutiny. It transpired that the experts always examined the whole graph (and sometimes even several preceding graphs) to assess the quality of the simulation. The experts assess both global and local trends in the residual errors and their decisions can often be influenced by historical data, i.e. the outcome of the previous time steps. Unfortunately, the frequent adjustment strategy used by both systems makes such global assessment very difficult because using different sets of control parameters at different parts of the graph obscures its true form by introducing artificial irregularities. It became clear that correct control decision could not be based exclusively on the analysis of local trends, as the residual graphs often contain very important macro-features that are essential to the effectiveness of the assessment procedure. Relying only on local analysis can also be misleading as the residual values are inherently noisy and there is always a danger that the noise will be interpreted as a trend. The natural "noisiness" is compounded even further by the kick effect, which not only makes the local analysis more difficult but also makes the global assessment virtually impossible.

4.6.3 Incorrect control architecture

As the analysis progressed it was becoming apparent that the underlying control architecture was based on unsound principles. Despite various enhancements, the software agent failed to provide the reduction in simulation time or stability improvements that were hoped for. Furthermore, the convergence of every time step could not be fully guaranteed. A quick solution for those problems was not apparent. It was agreed that none of the improvements so far conceived seemed capable of providing the required benefits. However, it was known that the experts were able to significantly reduce the simulation time and guarantee full convergence. Therefore, it was concluded that the system architecture and the associated control actions must have been inappropriate and fundamentally different from the human control techniques. This conclusion led to the reassessment of the information extracted from the experts during the knowledge acquisition phase.

The knowledge reassessment identified distinctive differences between the control methods used by humans and the algorithms used in both control systems. Further interviews with the experts established that the global features of the residual graphs were essential in determining the best adjustments. Local trends may also contain vital information but should only be used to enhance the result of the global assessment.

4.6.4 Analysis summary

Further analysis revealed that the rules used in the KBS were brittle and arbitrary, as there was insufficient knowledge available at the time about the effects of frequent control actions. Experts tried to help but did not have the necessary knowledge, since they hardly ever modified the standard set of relaxation values. Most of the time the experts were put off by the amount of time necessary to exercise proper control and therefore their experience was limited and came from rare and non-standard cases. This further confirmed that the knowledge elicitation was not very effective and failed to identify the issues that were essential to the full understanding of the control problem. This was probably due to the combination of factors of which the main ones were:

- Lack of experience in formal knowledge elicitation techniques.

- Too much emphasis during the interviews on the control techniques used in the original Ewer's system.
- Insufficient expertise available (the experts interviewed did not have all the necessary knowledge required to build an effective rule-driven control system).

The analysis also established that the previous conclusion, stating that the frequent relaxation adjustment strategy was a natural extension of the control technique used by experts, was incorrect and the control system built around that principle did not emulate the human control actions properly.

A brief summary of the problems affecting the KBS 2.0 is presented below:

- The kick effect seriously affects the performance.
- The assessment method is focused on local features rather than full graph assessment.
- The kick effect introduces artificial irregularities hence making the full global analysis of the graph impossible.
- The system architecture does not easily allow for the time step size to be changed.
- There is no reliable divergence recovery procedure.
- Full convergence assurance is difficult to implement and guarantee.
- The set of rules used is very brittle and inflexible.
- Control actions fundamentally different from human control.
- "Little-but-frequent" control strategy proved to be ineffective and inefficient.

As a result of these problems the 3D cases could not be properly controlled. There was no significant reduction in the observed simulation time while in many cases, the automated control even led to a small increase in the total simulation time. These problems prompted a search for the new architecture for the control system.

4.7 New approach to the control problem

The major factor contributing to the failure of the previous control strategy was the presence of the "kick" and its adverse effects on performance and stability. Consequently, one of the priorities adopted for the design of the new architecture was whenever possible to avoid introducing local instabilities. It was proposed that all the modifications to the control parameters should be applied between the time steps. In this way, adjustments 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 the experts. Unfortunately, the experiments with the KBS 2.0 showed that there was insufficient knowledge available to build a robust rule-driven control system. This was partly due to the fact that the knowledge acquisition was not exhaustive enough but also because the experts were not familiar enough with the nature and the effects of various adjustments since they only performed them very rarely. It was also becoming apparent that the relaxation and the time step size should be modified simultaneously but there was no knowledge available about the effects of combining these two types of control actions. Moreover, even the effects of simple adjustments were not well known and therefore it was virtually impossible to design a reliable control system at that stage.

This analysis led to the conclusion that an effective rule-based control system could not be built using the expertise then available, as very little was known about the effects of control actions. Further research was therefore required to enhance our understanding of the problems associated with automated control of CFD simulations. Moreover, new control methods should be investigated, as the knowledge acquisition failed to identify formal rules that could be used in a classical rule-driven expert system.

4.8 Investigation of the effects of control actions

The knowledge reassessment showed that very little was known about many aspects of control actions. In standard cases experts do not usually interfere with the simulation process. The relaxation is adjusted very rarely – most of the time the default set of parameters stays intact for the whole simulation. The time step size is modified even less frequently. Generally, if the simulation went badly then a second run with a smaller time step size is considered. However, the experts generally seem to agree with the following statements:

- Adding under-relaxation slows down the simulation but at the same time makes it more stable.
- Increasing the time step can lead to divergence but may also speed up the simulation.
- Removing under-relaxation speeds up the solution but can also cause divergence.
- Decreasing the time step size stabilises the simulation but is also believed to adversely affect the performance.

There is little information about whether the changes can be combined (e.g. increasing the relaxation and reducing the time step size at the same time) and what the effects of such combinations would be. It is also not very clear what changes are best suited to a particular situation (e.g. what needs to be adjusted in case of divergence: time step size, relaxation or both?). Moreover, the experts have contradicting views about the magnitude of changes that should be applied and their impact on performance and stability.

In order to answer all those question and confirm the experts' intuition, a comprehensive set of experimental runs was devised to investigate various types of control actions and analyse their effects. The main goal was to test a broad range of changes to control parameters and store the full set of results for further analysis. Based on the analysis of KBS 2.0 it was decided that all control actions should be confined to the period between the time steps and therefore the experiments should only include this type of adjustment.

Several different cases were used for this investigation. All the scenarios were based on a Steckler case (Steckler-82) but with different heat source locations and different fire sizes. The geometry of the Steckler room remained unchanged while the heat output of 50kW and 250kW was varied (62.7kW was used in the original experiments). Steckler-type case is convenient, as it is very well documented and is often used as a benchmark case (Grandison-01). It reaches a steady state, which allowed testing the effects of the control actions in the steady state as well. Furthermore, since it is a fairly small and simple case, the processing time is relatively short. This fact was very important, as the experiments increased the normal execution time by a factor of five. The main limitation of the Steckler case is the use of a heat source with constant output. Constant heat output does not model the fire growth correctly and therefore two additional cases were run, which contained a heat source described by a growing heat release curve (with a peak at 50kW and 250kW respectively). A heat release curve is a commonly used to model the fire growth process.

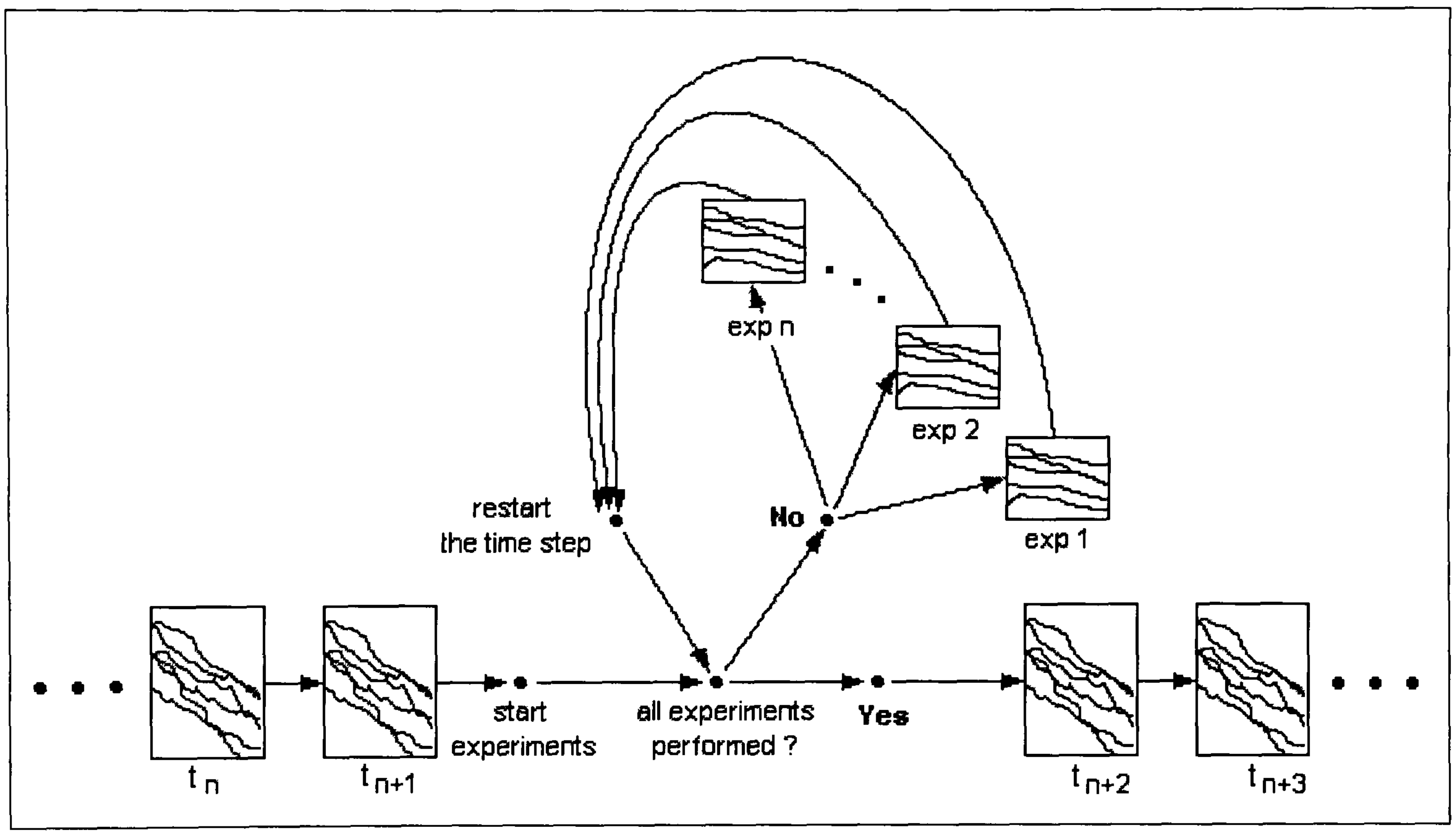


Figure 4-4 Diagram of the experiment procedure

Ten different scenarios were simulated during the tests. Each scenario consisted of 100 time steps (with a time step size either 1s or 2s) with the control parameters being modified after every 5 steps. Consequently, for each case there were 25 points where a comprehensive set of control actions was tried. Every set of experiments contained 20 different control actions that tested various types of adjustments. At each experiment point, the simulation state was first saved, and then one type of modification introduced and the time step restarted. When the time step has completed then the relevant data was saved for further analysis, the previously stored simulation state was recovered and a different type of control action tested. This procedure was repeated until the whole set of modification had been tested, in which case the initial state was retrieved one more time and from that point the simulation progressed undisturbed using the original control parameters until the next experiment point (i.e. for the next 5 time steps). Figure 4-4 presents a diagram of the experiment procedure.

As every case contained 100 time steps with the experiments performed after every 5 steps, the total of 500 experiments were performed within a single scenario, which gave a total of 5,000 experiments. A single type of control action was therefore tested 250 times, which produced sufficient data to use statistical analysis.

The details of different types of modifications that have been tested are presented in Table 4-1. The changes were relative and were limited to ±50%, ±20% for relaxation and +100%, -50% for the time step size. The relaxation adjustment was always applied uniformly to all variables. The results of the experiments were analysed using two different methods:

- **Convergence speed analysis** – every experiment was compared against the corresponding time step without any changes applied. The difference between the number of sweeps required to attain convergence determined the relative speed improvement (or deterioration). The final figure describing the convergence speed was normalised to represent a uniform measure of the improvement.
- **Visual analysis of the residual graph** – the residual graphs, which were stored during the simulation, were then visually assessed using a viewer developed specifically for this purpose. The visual inspection was necessary to compare various features that might reflect the stability of the simulation (smoothness, presence of a flat-convergence phase, major irregularities, oscillations, etc). It was a tedious task requiring considerable patience (as it involved assessing 5000 residual graphs) that was made much easier by the custom viewer, which allowed quick comparison and assessment of the relevant features.

Exp No.	linear relax. mod. factor	False time step relax. mod factor	Time step size mod. Factor	experiment description
1	+50%	+70%	0%	Substantial relaxation increase, no time step size change
2	+20%	+30%	0%	Relaxation increase, no time step size change
3	-50%	-50%	0%	Substantial relaxation reduction, no time step size change
4	-20%	-20%	0%	Relaxation reduction, no time step size change
5	0%	0%	+100%	No relaxation change, time step size doubled
6	0%	0%	+50%	No relaxation change, time step size increased by 50%
7	0%	0%	-50%	No relaxation change, time step size halved
8	+50%	+70%	+100%	Substantial relaxation increase, time step size doubled
9	+20%	+30%	+100%	Relaxation increase, time step size doubled
10	-50%	-50%	+100%	Substantial relaxation reduction, time step size doubled
11	-20%	-20%	+100%	Relaxation reduction, time step size doubled
12	+50%	+70%	+50%	Substantial relaxation increase, time step size increased by 50%
13	+20%	+30%	+50%	Relaxation increase, time step size increased by 50%
14	-50%	-50%	+50%	Substantial relaxation reduction, time step size increased by 50%
15	-20%	-20%	+50%	Relaxation reduction, time step size increased by 50%
16	+50%	+70%	-50%	Substantial relaxation increase, time step size halved
17	+20%	+30%	-50%	Relaxation increase, time step size halved
18	-50%	-50%	-50%	Substantial relaxation reduction, time step size halved
19	-20%	-20%	-50%	Relaxation reduction, time step size halved

Table 4-1. Types of control actions tested during the experiments

The results of the analysis form a substantial document but only the conclusions are relevant for this dissertation:

- The increase of relaxation speeds up the convergence. However, excessive relaxation can destabilise the simulation causing divergence.
- Removing the relaxation slows down the simulation (It is also believed that it can provide divergence recovery but there was no conclusive evidence in the experiments although the results did indicate that removing relaxation stabilised the solution).
- Smaller time steps are usually more stable but not necessarily slower (!). Therefore the time step size reduction may be a very efficient method for divergence recovery since the impact on performance is minimised.

- Bigger time steps can provide some reduction in simulation time but may also cause major convergence problems.
- The results indicate that the time step size is responsible for the stability of the simulation while the relaxation controls the convergence speed. However, the relation is not straightforward and there are many other factors involved.
- Increasing the relaxation and reducing the time step size can produce superior results in terms of improved time performance and acceptable stability.
- Increasing both the relaxation and the time step size can produce a substantial reduction in execution time but it is also likely to cause major instabilities and divergence.
- Removing the relaxation and reducing the time step size improves the stability but at the same time significantly degrades the performance.
- Removing the relaxation and increasing the time steps size does not provide any tangible benefits. The performance is seriously affected while the stability of the simulation does not improve. Consequently, this type of changes did not seem to provide any benefits regardless of the current simulation state. It neither stabilised the simulation nor accelerated the convergence.

The experiments confirmed the experts' opinions about the stabilising effects of relaxation removal and the time step size reduction. It was also confirmed that adding relaxation might speed up the convergence and consequently the whole simulation process.

But there were also a few surprises revealed during the experiments. The time step size reduction was generally believed to have an adverse effect on the performance. This belief however, was not confirmed by the experiments. In most of the cases, reducing the time step size had little effect on the performance. The convergence speed was not significantly affected however the actual result varied (from a 20% reduction to a 60% increase in convergence time). Moreover, the smaller time steps always appeared more stable and smoother than their larger counterparts. This came as a surprise to the experts since they normally assumed that a bigger time step meant shorter simulation time. The experiments showed that the time step size has more impact on the simulation stability than speed. Therefore a simulation of a scenario might take a similar amount of time regardless of whether it uses 100 steps of 1 second or 500 steps of 0.2 second. Of course, the reality is more complex and the above example is somewhat simplistic. For instance, the findings are based on the assumption that all the time steps have fully converged. In a typical non-controlled simulation many time steps do not fully converge but run for a predetermined number sweeps. In such cases, a bigger time step always means a shorter simulation time, as the processing time is defined as a number of sweeps multiplied by the number of steps and therefore is not convergence dependent. This may also explain why experts believed that a bigger time step reduces the simulation time. However, it is understood that non-converged time steps affect the accuracy of the results and such simulations should only be used for "quick and dirty" runs. Consequently, all performance comparisons presented in this dissertation are always based on fully converged simulations (whenever possible, as in some non-controlled simulations obtaining convergence of all time steps is very difficult to achieve).

The experiments identified one class of control actions (relaxation removed and time step increased) as ineffective since it never provided any tangible benefits. The experiments also helped to link some types of control actions with a particular simulation state (e.g. reducing the step size can restore convergence). Further research is required to fully understand these relationships. The experiments showed that the effects of the control changes vary depending on a case but there are some general rules that apply most of the time.

Apart from facilitating the analysis of different control actions, the experiments also provided a considerable amount of data (in the form of residual graphs) that could be easily accessed and analysed further using a specially developed viewer. These graphs proved to be invaluable in further development and quick validation of different assessment techniques.

4.9 Summary

This chapter describes initial attempts to develop a rule-driven control system for SMARTFIRE. A prototype system created by Ewer is presented and its advantages and deficiencies analysed. The subsequent attempts to improve the original Ewer's design are described together with the discussion of the initial results. The reasons behind the poor performance of the system are analysed and several different factors that contributed to the lack of performance improvements are identified. It is shown that due to insufficient expertise available, a pure rule-based approach cannot be successfully applied at this stage. Some suggestions for other types of control architecture are offered. Finally, the details of experiments performed to gain insight into the nature and effects of various control actions are presented together with a brief discussion and conclusions.

Chapter 5

Heuristic search as a control technique

5.1 Introduction

In Chapter 4 it was shown that the rule-based systems were not very successful in emulating the techniques used by human experts to control fire simulations. In this chapter we propose a new system that relies on heuristic techniques to determine the optimal control parameters. A complete heuristic evaluation function is presented together with three dedicated algorithms developed specifically for residual graph feature extraction. All components described here form the building blocks for the new heuristic control system.

This chapter also provides a reference to other relevant research projects that use heuristic techniques to solve different types of complex problems.

5.2 New architecture for the control system

After the failure of the rule-based approach, the knowledge elicited from experts was reassessed. It was known that the experts were able to significantly improve the performance but still, the automated system designed to model human control actions failed to provide any improvement. It became obvious that the initial design was inherently flawed and it was believed that there were several different factors responsible for its failure:

- Knowledge elicitation phase was not thorough enough.
- The chosen architecture (frequent rule-driven relaxation adjustments) did not model human control actions correctly.
- Experts use only limited number of stable rules and therefore their control actions are often based on subjective assumptions and trial-and-error search.

- Different control procedures are used by different experts, which made it difficult to design a system based on a consistent and exhaustive set of rules.
- Incorrect conclusions were drawn from Ewer's test results and the initial interviews with experts.

The first stage in the search for a new solution involved establishing precisely which facts were known and could be used for the purpose of automated control. Although there was insufficient expertise on control rules, there did exist knowledge about when a simulation was going well and when it was not. Based on the residual graphs, an expert was able to differentiate between acceptable and unacceptable solution states. This prompted 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 <u>results</u> of any control actions.

The new control system must therefore be able to closely emulate the experts' ability to extract solution state information from residual graphs. This fact was already well known but it was not pursued actively enough during the initial stage of knowledge acquisition.

It also became apparent that the experts never precisely knew what control actions should be applied. Some experts claimed to follow some arbitrary rules (e.g. reduce time step size if enthalpy clearly diverges) while other used a generic approach and, for instance, always reduced the time step size when the solution diverged. Furthermore, experts were never certain whether the control action they had applied would have the desired effect. It was therefore a common practice to save the solution state before making any adjustments so that the previous state could always be restored if the adjustment did not work as planned. This approach resembles a trial-and-error search although there is an element of expertise in the search.

The overall structure of the human control procedure outlined above, is that of a heuristic search with a heuristic evaluation function derived from the examination of residual graphs and residual histories. Consequently, it was decided that the new control architecture should emulate the human control technique using a simple search algorithm guided by appropriate heuristics, based on residual graph assessment.

5.3 Heuristic search algorithm

One of the main benefits obtained from the analysis of KBS ver. 2.0 failure was the conclusion that the control actions should be applied between time steps and should also include time step size modifications. Another important fact was the observation that experts always saved the solution state before making any adjustments so they could recover from change-induced divergence. The experts rarely performed performance-oriented adjustments because of the length of time required to assess the results of a single control action. Fortunately, an automated system does not suffer from tiredness and can perform a much more exhaustive search examining several different types of changes in order to find an almost-optimal set of control parameters. As a result, the new system was designed around the following main principles:

- A runtime heuristic search is used to determine appropriate control parameters.
- Heuristic evaluation function based on residual graphs is used for the assessment of both the simulation and the search result.
- Automatic divergence detection is based on similar heuristics.
- A heuristic search is also employed in divergence recovery.

The most comprehensive approach to the control problem would be to test all the possible combinations of the parameters for every time step and then select the path that provides the best results (e.g. shortest execution time). Such exhaustive search can find the optimal path to the solution but the cost of the search would be very great. Since the heuristics relies on the assessment of the residual graphs, the search would require a very large number of full simulations to be performed just to obtain the relevant data. Clearly, this method is inappropriate, especially as we are not interested in finding the shortest path to the solution but in obtaining the final simulation results in the shortest possible time. However, for the purpose of discussion it is useful to consider some of the features associated with exhaustive search. Consequently, there are two main difficulties associated with such search method:

- The parameters are continuous and therefore it is impossible to create an exhaustive and finite set of different configurations of control parameters. Consequently, the branching factor of the search tree is unlimited.
- The cost of the exploration of a single set of parameters (examination of a single node in the search tree) is of the same magnitude as the cost of a single time step. As a standard simulation usually contains at least 100 steps and runs for several hours or days, the cost of the exhaustive exploration is prohibitive.

The first problem can be easily overcome by using a discrete subset of the control parameters. This technique avoids the infinite branching problem thus making the search cost finite although still not feasible. Assuming that **20** discrete sets of control parameters are used, the equivalent branching factor is also **20**. Therefore, the cost of the search is $\mathbf{20}^{x}$ (where x is the number of time steps – for simplicity we do not consider changes to time step size). Consequently, the cost of the search for a close-to-optimal path (but not fully optimal, as the control changes are discrete) is still absolutely unacceptable. Fortunately, a numerical simulation contains special properties that can be exploited in order to reduce the cost of the search.

Firstly, we are not interested in finding the optimal path to the solution. The two main goals:

- reducing the simulation time,
- ensuring the validity of the results

can be achieved without finding the shortest possible path to the solution. Since the search time forms part of the simulation time, it must be kept to the absolute minimum. On the other hand, we have assumed that a heuristic search is the right solution to our control problem. Consequently, an appropriate search procedure must be devised, which is capable of minimising the search time while maximising the performance improvements obtained by the control parameters found during the search.

Secondly, experts consider a time step to be valid if the residuals have converged. Consequently, time step correctness does not directly depend on the set of parameters used. The sub-optimal path can provide better performance if the search cost is limited. Generally, there is little point in trying to find a better set of parameters for the time step that has already produced correct results (a different situation arises if the time step diverges – then the correct parameters must be found). This property is very important as it indicates that we never need to backtrack further than the last fully converged time step.

Thirdly, if the final solution exists then a path to the solution can be found from any point between START and END, providing that all time steps between START and the chosen point have converged. This fact has profound implications as it guarantees that if a particular time step diverges then a single-step backtracking will always be sufficient to resolve the problem and return to the path that leads to the solution. In other words – there are no hidden dead-ends, i.e. divergence can always be resolved by backtracking by one step and trying again with a different set of parameters.

These conclusions resulted in the development of a simple search algorithm that virtually guarantees finding a correct solution if one exists. **The algorithm uses a best-first search strategy (a greedy search) with single step backtracking and employs a heuristic function, which evaluates different control parameters by performing partial time step simulation.**

It was decided that the simulation process would be treated as a search for the solution. The goal is to obtain accurate results for all time steps in the shortest possible time. It is very important to differentiate between the following two goal definitions:

- finding the shortest path to the goal,
- reaching the goal in the shortest possible time.

In the first case, the path is the required solution to the problem. A classical example of such search is the 8-puzzle (Pearl-84) where the objective is to rearrange a given initial configuration of eight numbered tiles arranged on a 3x3 board into a given final configuration (usually an ordered sequence). Since the final state is given, the main task is to find the shortest sequence of actions that lead to this state.

In the second situation one is primarily concerned with reaching the target in the shortest possible time. Therefore, there are two distinctive goals: reaching the solution and minimising the search time. Again, a number of examples are available in literature. A typical one is the 8-Queens problem (Floyd-67) where the goal is to place 8 queens on a chessboard such that no queen attacks each other. One is only interested in the final solution and therefore the search path is of little interest. However, since no one wants to wait several hours for the solution, the search time should be kept to minimum.

A slightly more sophisticated example is the problem of real-time robot navigation in unknown environment. In this problem the final state is known (the destination) but the path is not. However, the search is not focused on finding the shortest path but on reaching the destination in the shortest possible time. Of course, the path has significant impact on the time required to reach the destination but there are further factors that have to be considered, like the cost of environment exploration (since it is initially unknown). We will describe this problem in more detail as it shares many characteristics with the control architecture proposed for SMARTFIRE.

The autonomous robot navigation is an intensively researched subject in AI. Several examples of different heuristic navigation systems were already presented in Chapter 3. Here we will concentrate on a single case and use it to highlight the similarities with a CFD control system. It is a classic problem of automatic navigation through an obstacle course from A to B and it is largely based on the research into Mobile Robot Obstacle Avoidance by Borenstein and his team (Borenstein-91, Shoval-94 and Ulrich-00). However, the example presented here has been modified to emphasise the issues related to the control algorithm for CFD simulations – our robot uses a very slow route-finding algorithm and therefore cannot use it in real-time.

In our example a robot (agent) has to move from the point A to the point B without any information about the topography of the terrain. The agent knows its position and the position of the goal (e.g. using GPS). Primitive sensors enable the agent to detect obstacles when it bumps into one of them. It also has a custom vision system that can see at some distance and recognise obstacles but analysing the input is energy consuming and takes long time. Therefore it is often better to choose a longer path and consult the vision system infrequently rather than spend a lot of resources on the search for the shortest route since the goal is to reach the point B as quickly as possible. Blind walk is not especially effective since every time an obstacle is detected the vision system must be engaged in order to find the way around the obstacle (we assume that the agent is unable to go around the obstruction without the feedback from the vision system).

A heuristic search algorithm can be used to help the robot navigate. A simple example will help describe how the algorithm works. Figure 5-1 shows an environment with the starting point (A) and the target (B). Dark shapes represent obstructions that the robot has to avoid while dark lines and shaded regions represent areas that take long time to cross and therefore it may sometimes be quicker to go around them.

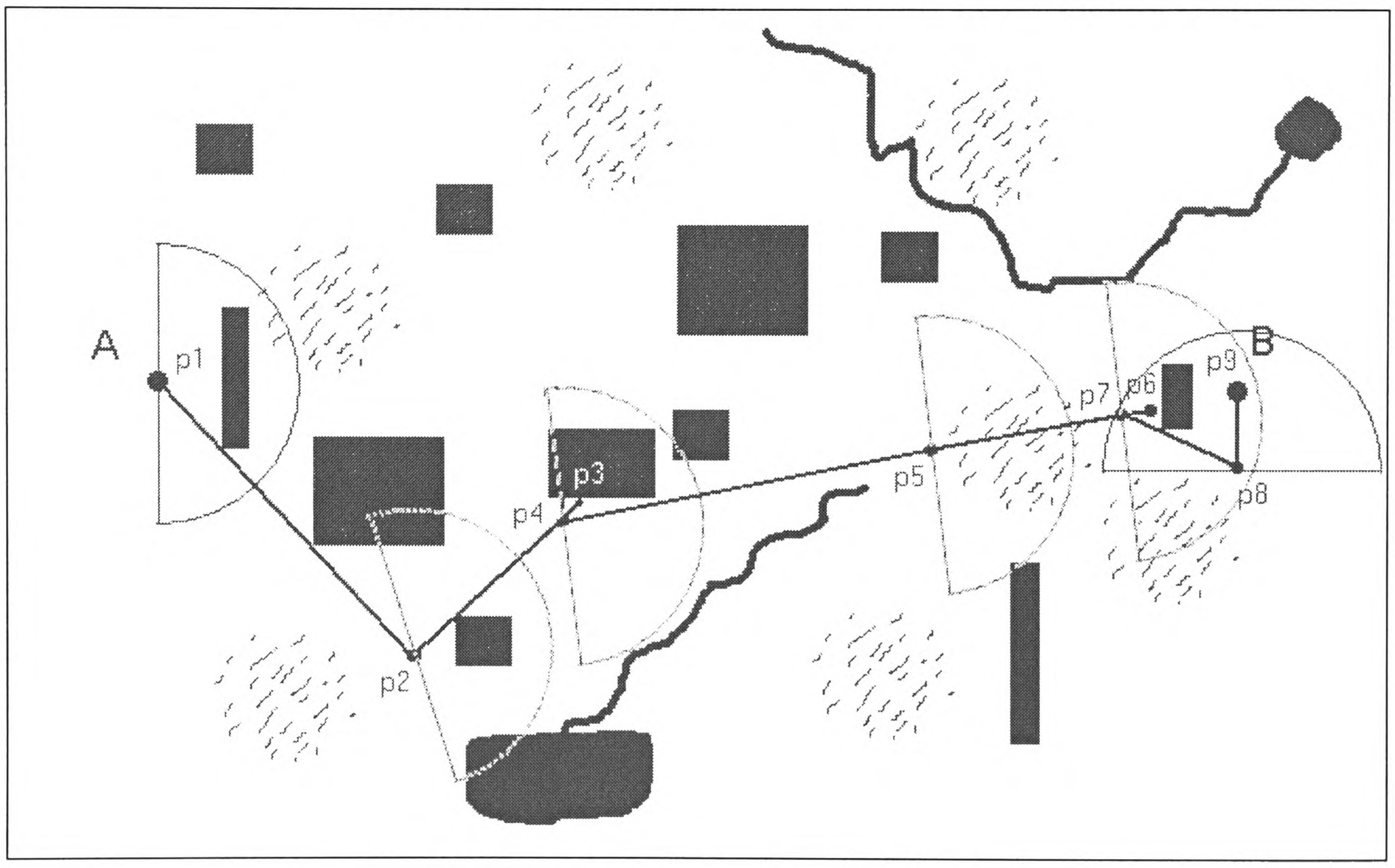


Figure 5-1 Navigating robot example

The consecutive stages in the robot's progress are discussed below:

Starting point (p1)

The agent (robot) uses its vision system to determine in which direction it should start moving. The semicircle represents the area covered by the optical sensors. The obstruction in front is detected and the agent chooses the direction that avoids the obstacle but gets him to the goal at the fastest rate. Robot turns off his vision system and moves forward for a specific distance, The distance at which the robot intends to travel is not fully covered by its vision system and therefore the process of choosing the direction is inherently heuristic. The robot does not encounter any obstacles and arrives at the point p2.

Next decision point (p2). The agent engages the vision system (always looking in the direction of the goal). The seemingly best route is chosen and robot moves forward in this direction.

Obstacle detected (p3). The agent bumps into an obstruction and has to revise its route. It goes back a short distance and arrives at point p4.

Obstacle avoidance (p4). The agent determines a new direction that avoids the obstruction in front and starts moving forward.

Next decision point (p5). The agent has covered the pre-set distance and arrives in point p5. The vision system is used and an internal reasoning engine determines that crossing the rough area in front will be quicker then going around it. The robot moves in the chosen direction.

Obstacle detected (p6). The agent bumps into an obstruction and has to revise its direction again. It goes back a short distance and arrives in point p7.

Obstacle avoidance (p7). The robot determines a new direction that avoids the obstruction in front and starts moving forward.

Moving away from the goal detected (p8). The agent detects that it has started to move away from the goal. It stops and uses the vision system to find a new direction. It then moves in the selected direction.

Goal reached (p9). The agent reaches the goal.

This algorithm is neither optimal nor infallible but it is interesting because of its similarity to the search algorithm proposed for the CFD control system. Complex features of the CFD simulations could obscure the description of the algorithm while a simple route-finding problem contains a sufficient number of details but is easy to understand and provides a suitable vehicle to demonstrate the benefits and drawbacks of the algorithm.

Unfortunately this algorithm is not guaranteed to find the solution even if one exist. This is the consequence of the following properties:

- No advanced backtracking capability – the agent cannot recover if it gets stuck in the dead end.
- Possibility of failure to find the path to the target when confronted with complex and/or large obstructions.

Both of these problems can be addressed by certain improvements to the algorithm but this is not relevant for our purposes. Instead, we will focus on the similarities between this algorithm and the control technique for the CFD system and explain later why these problems do not apply in CFD simulations.

The algorithm used by the robot can be formalised as follows:

```
1. Use the vision system to find the most promising direction
   to advance.
2. If no direction was found – unable to solve the problem.
   Terminate.
3. Move in the direction selected.
4. If goal reached – success.
5. If obstruction detected then go back and then go to step 1.
6. If the pre-determined distance covered then go to step 1.
7. If moving away from the goal go to step 1
```

The vision system is used to obtain the heuristics that determine the next move. Even if the vision system were on constantly it would still provide only heuristic information, as it can see for a limited distance only. The environment is not fully accessible and therefore only the backtracking techniques can guarantee finding the shortest path to the destination. The important point is though, that finding the shortest path is not necessary, as the goal is to reach the destination in the shortest possible time. Finding the shortest path often requires an extensive exploration of the domain, which can take a substantial amount of time and therefore can be very inefficient.

The heuristics used (the logic behind the vision system) is fairly complex as it must eliminate the paths leading to obstruction, assess whether it is more efficient to choose a shorter but slower path through a rough region or walk around it. It should also estimate which direction leads to it reaching the goal in the shortest time. There is another important feature – although the agent moves with its vision system switched off it can still detect arising problems (i.e. bumping into the obstruction) and initiate corrective actions.

Now a similar algorithm is presented but applied to a CFD simulation.

1. ***Use the look-ahead system to find the most promising set of control parameters for subsequent time steps.***
2. ***If no suitable parameters were found – terminate.***
3. ***Execute the next time step.***
4. ***If problems detected (divergence, oscillations) restart the time step and go to step 1***
5. ***If n time steps executed since the last look-ahead go to step 1.***
6. ***If goal reached – success.***
7. ***Go to step 3.***

The algorithm is virtually identical to the one used in the route finding system. It guarantees finding the solution providing that the following criteria are met:

- The goal is theoretically accessible.
- The goal can be reached from every point of the simulation providing the preceding time steps completed successfully (i.e. there are no dead-ends).
- The agent never moves away from the goal.

To reduce the complexity we presume that it is up to the expert to decide whether the goal is accessible and that the agent assumes that it can always find the solution. The third postulate is always true, as the time step size has to be a positive number and therefore every successfully completed time step brings the agent closer to the goal. The only remaining issue is the absence of dead-ends, which experts believe to be true and therefore it will not be investigate further (as this dissertation is focused on the emulation of expert control actions).

The complete algorithm incorporates the ideas that have been presented so far. The search cannot be performed after every time step because of the excessive computational cost incurred. Consequently, the search is only performed at specific points, determined by a dedicated scheduling method. Because the control parameters are continuous and therefore the potential search space is infinite, only a limited number of modifications can be tested in reasonable time. In the first version of the system, 15 different control actions are tried at each search point. The results are stored and then assessed. The control parameters that provided the best results are applied and used in the subsequent time steps. The evaluation function that identifies the best improvement forms the most complex part of the system and is essential to its effectiveness.

5.4 Evaluation function

A heuristic search cannot be performed without an appropriate evaluation function. The available paths (i.e. sets of control parameters) that may lead to the solution have to be assessed in order to select the best one. It was determined that experts did have some knowledge that could assist in the development of such function. Extensive interviews with experts were conducted, which provided valuable information, however the final conclusions remained ambiguous. The experts agree that most of the information that is important to the control system can be extracted from the residual error graphs. It is also accepted that convergence speed (the number of sweeps required to attain convergence) determines the performance. Generally – the fewer sweeps are required to complete a unit of the simulated time (e.g. 1s) the better. Unfortunately there was much ambiguity about the influence of various other features present in the graph on the assessment result. Experts could not fully agree on the importance of oscillations and irregularities. Consequently, a special procedure was devised, whose only purpose was to formalise the evaluation algorithm by analysing the experts' assessment procedure. The extensive set of graphs acquired during the earlier experiments was analysed and the results discussed with the experts. The experts were asked to order a group of graphs according to their quality (or rather, the quality of the corresponding simulation state) and then to describe the features that influenced their decisions. Initially, the assessment was expressed verbally but as the interview progressed, it started becoming more formal. The findings were summarised and then confirmed with experts during final interviews. As a result three major features, believed to reflect the underlying simulation quality and speed, were identified. The experts acknowledged the findings but were unable to determine the necessary priorities and therefore could not fully assist in the development of a combined evaluation function (a function that produces a final assessment based on the sub-assessments of each feature separately).

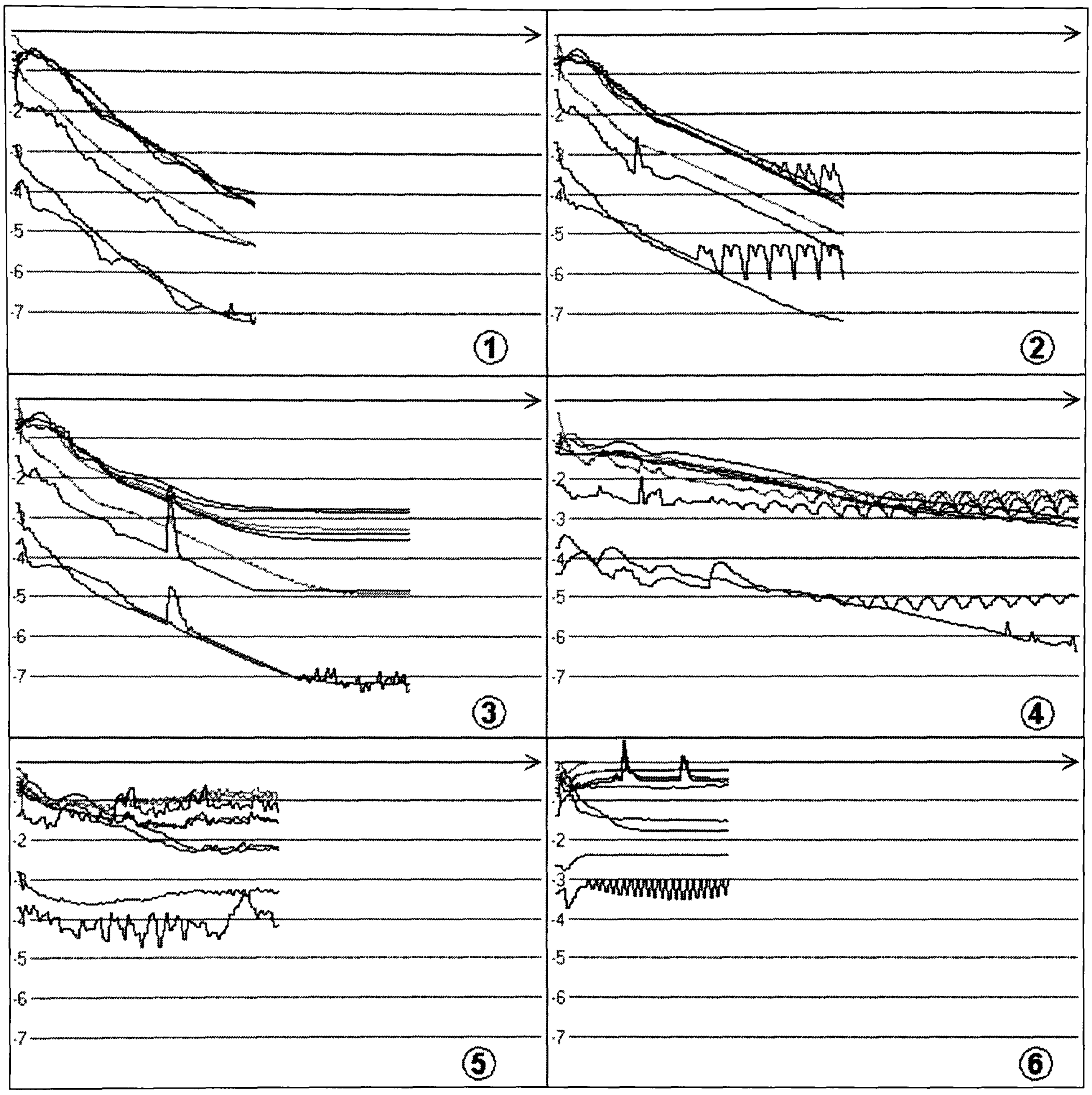


Figure 5-2 Residual graphs from various time steps in single simulation (250kW fire with fine mesh - see Section 6.4.2. for set-up details)

An example set of graphs that were used in the interviews is shown in Figure 5-2. All of these graphs were produced at various points of one simulation and are shown on a logarithmic scale (e.g. value -2 represents 0.01 residual error). The first graph (1) was assessed as the best one – all the variables quickly converged to the required tolerance (0.0001). A consistent downward trend was present throughout the whole time step. The next graph (2) also converged but contained some oscillations that seemed to have delayed convergence. However, the trend was mainly down. The following graph (3) failed to converge despite the fact that it was smooth and the initial convergence rate looked very promising. At one point however, the residuals flattened out and remained constant above the required tolerance. The fourth graph (4) contained substantial oscillations, a few residuals actually increased and there were also further irregularities (spikes and bumps). The graph did not converge. Graph (5) displayed more serious problems. It contained serious irregularities that could not be classified as oscillations. There seemed to be no consistent trend in the residuals as they stayed at approximately the same level but with substantial fluctuations. Consequently, convergence did not occur in that time step. The last graph (6) had small irregularities and oscillations but the residuals remained at a high level after very brief period of increase.

Similar analyses were performed on several sets of graphs available from the earlier experiments. This round of interviews was much more successful than the first one as it was eventually identified that the three most important factors influencing experts' assessment are:

- **Convergence speed** (number of sweeps required to attain convergence)
- **Presence and magnitude of irregularities** (sections of graph where the residuals differ significantly from the global trend)
- **Presence and magnitude of oscillations** (sections of graph that exhibit strong periodic variations in residual values)

The experts agreed that this list represented a comprehensive set of features, which they normally use in their assessment of the residual graphs. One issue remained unresolved – determining how much influence each of these features had on the assessment result. The experts were not able to answer this, as each one of them used their own informal assessment. Nevertheless, the identification of these features was a big step forward, as it provided the building blocks for a combined evaluation function. The feature extraction algorithms developed as a result of the interviews are presented in the next section.

5.4.1 Convergence speed

The convergence speed can be defined by the following formula:

$$conv\_speed = \frac{sweeps}{sim\_time} \quad \left[s^{-1}\right] \qquad (5.1)$$

The value represents the average number of sweeps required to solve a single second of the simulated time (for the lack of a better name, the term 'convergence speed' is used throughout this thesis although it is counterintuitive as its value decreases when the performance improves). The average convergence speed can be calculated for the whole simulation, a part of it or for a single time step only. Figure 5-3 presents residual graphs from two time steps, both starting at the same point in the simulation but using different control parameters. This example describes the method for convergence speed calculation.

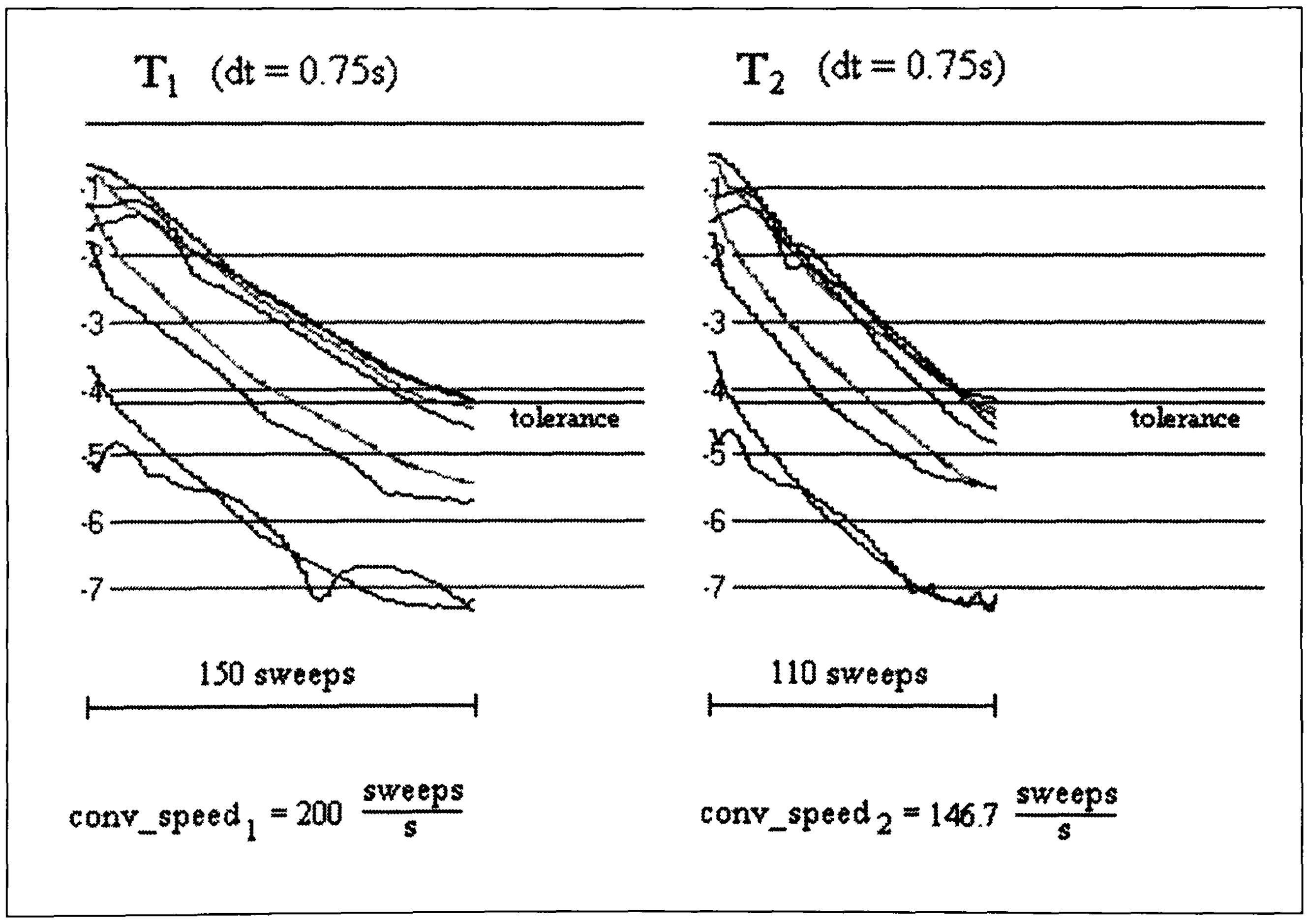


Figure 5-3 Convergence speed comparison

The number of sweeps to convergence can be obtained directly from the fully converged graph, as the actual convergence point already exists. However, this is not very useful since we usually want to estimate the number of sweeps to convergence before the required tolerance level is attained. An accurate early assessment is essential to the efficiency of the control system since it helps identify and terminate diverging time steps quickly. If an early experiment assessment determines that the required improvement could not be provided then it can be stopped immediately. This leads to big savings in execution time, which is a desirable feature in an efficient assessment procedure. Consequently, the following question must be answered: How to estimate the number of sweeps required to convergence for a not converged graph?

The search for an algorithm capable of estimating the convergence point started with the visual examination of many different graphs. The residual graphs are normally presented using a logarithmic scale as it makes trend assessment much easier while a linear scale obscures most of the interesting features. The initial analysis focused on stable and quasi-stable graphs only, since even the experts have difficulty predicting the behaviour of unstable or diverging graphs. The stable graphs, on the other hand, contain consistent trends and their behaviour can usually be predicted with reasonable accuracy.

Firstly, the experts were asked to extrapolate several residual graphs manually. Some of the examples (together with the corresponding full real-simulation graphs) are presented in Figure 5-4. The comparison suggests that the experts usually focus on the global trend while predicting the convergence point. They normally do not attempt to predict any irregularities or radical trend changes. It was therefore confirmed that the experts' technique is predominantly based on the linear extrapolation of the most recent trend. Most of the fluctuations are filtered out as they are considered to be irrelevant noise. The examples show that, although it is difficult to make a very accurate prediction due to frequent unexpected trend changes, the intuitive method used by experts provides a satisfactory approximation of the real convergence.

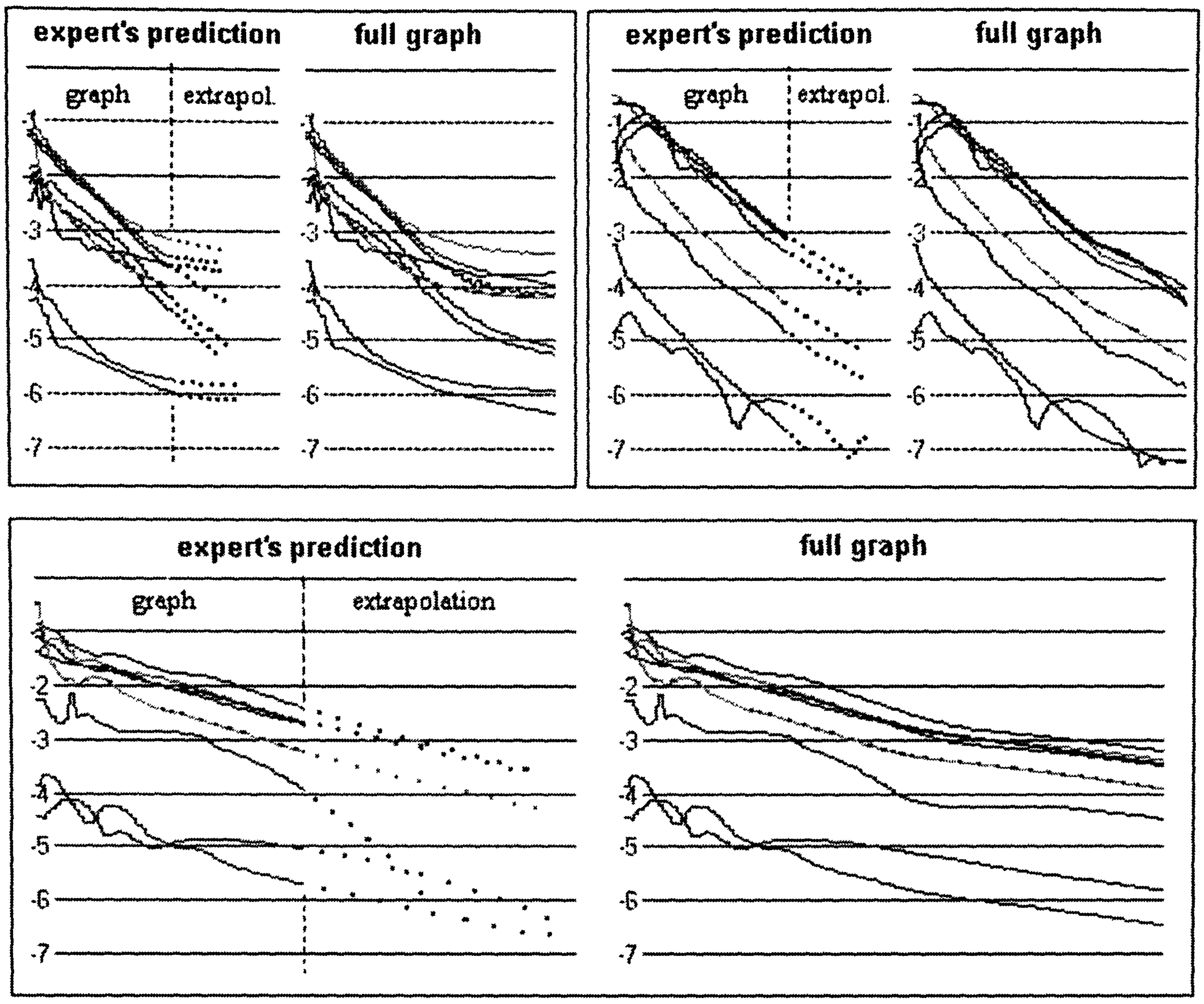


Figure 5-4 Comparison of experts' extrapolations and real graphs

This analysis led to the development of a formal approximation technique. As the approximation was linear, the first ideas involved extrapolating the residual graph using the least-squares method. An example result of such extrapolation is presented in Figure 5-5. It is apparent that the results achieved are extremely inaccurate. The linear approximation puts too much weight on the old trends that have little influence on the final convergence rate. The residual graphs are rarely linear and therefore only small segments can be treated as approximately linear. Experts normally use the most recent consistent trend as an indication of the current convergence rate. Consequently, a successful algorithm for convergence prediction should put more emphasis on the most recent trend(s).

A very simple technique (used by Ewer in his rule-driven system) would be to make the prediction using only a few of the most recent residual values. However, this method is not very accurate and more importantly, it is extremely sensitive to fluctuations in the final part of the graph. A single spike of a short duration but substantially different from the global trend can completely change the prediction result. Consequently, this method was also rejected as unsatisfactory.

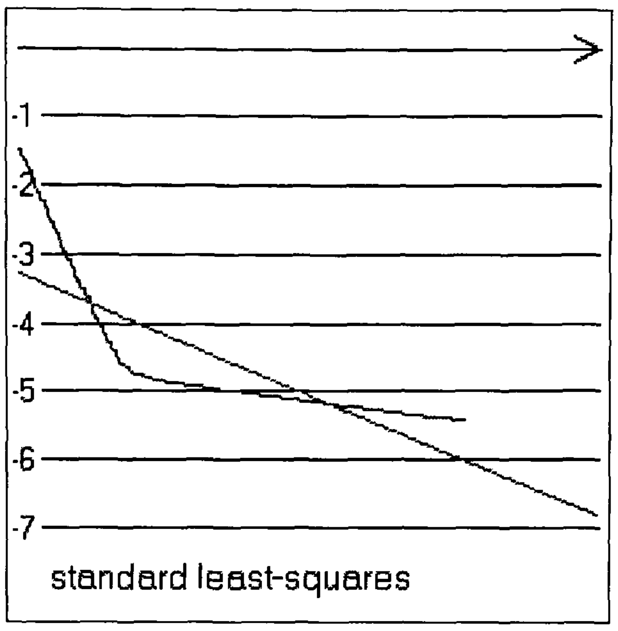


Figure 5-5 The least-squares approximation method

A more accurate prediction method should progressively add more weight to the points at the end of the graph and therefore closely track the final trend while still retaining some of the global trend characteristics. Such a method would be less sensitive to irregularities or oscillation in the final section of the graph. Fortunately, one can obtain the required feature by introducing a small modification to the least-squares method.

The classic least-squares method (Ott-93) obtains the linear approximation coefficients by minimising the sum of squared errors:

$$S = \sum (y - y')^2 ; \qquad where \;\; y' = ax + b \qquad (5.2)$$

The modified method allows fine-tuning of how much different groups of sample points influence the approximation. It can be achieved using a special weighting function **w(x)**. The function is applied to the errors and the approximation coefficients are computed by finding a minimum of the following formula:

$$S = \sum w(x) \cdot (y - y')^2 ; \qquad where \;\; y' = ax + b \qquad (5.3)$$

where **w(x)** represents the weight and can be virtually any function with real values. The coefficients are found using exactly the same method as in the derivation of standard least-squares formula but the results are slightly different:

$$a = \frac{n\sum w(x)\cdot xy - \sum w(x)\cdot y \cdot \sum w(x)\cdot x}{n\sum w(x)\cdot x^2 + \left(\sum w(x)\cdot x\right)^2}$$

$$b = \frac{\sum w(x)\cdot y - a\sum w(x)\cdot x}{n} \tag{5.4}$$

Using different functions one can obtain approximations that have various weight distributions. In our case it is necessary to track the final trend more closely while the early trends are less relevant. Several different weighting function were considered and finally two simple functions were selected for further assessment: ***sin(x)+1*** and x^2. Only short sections of these functions are actually used (Figure 5-6).

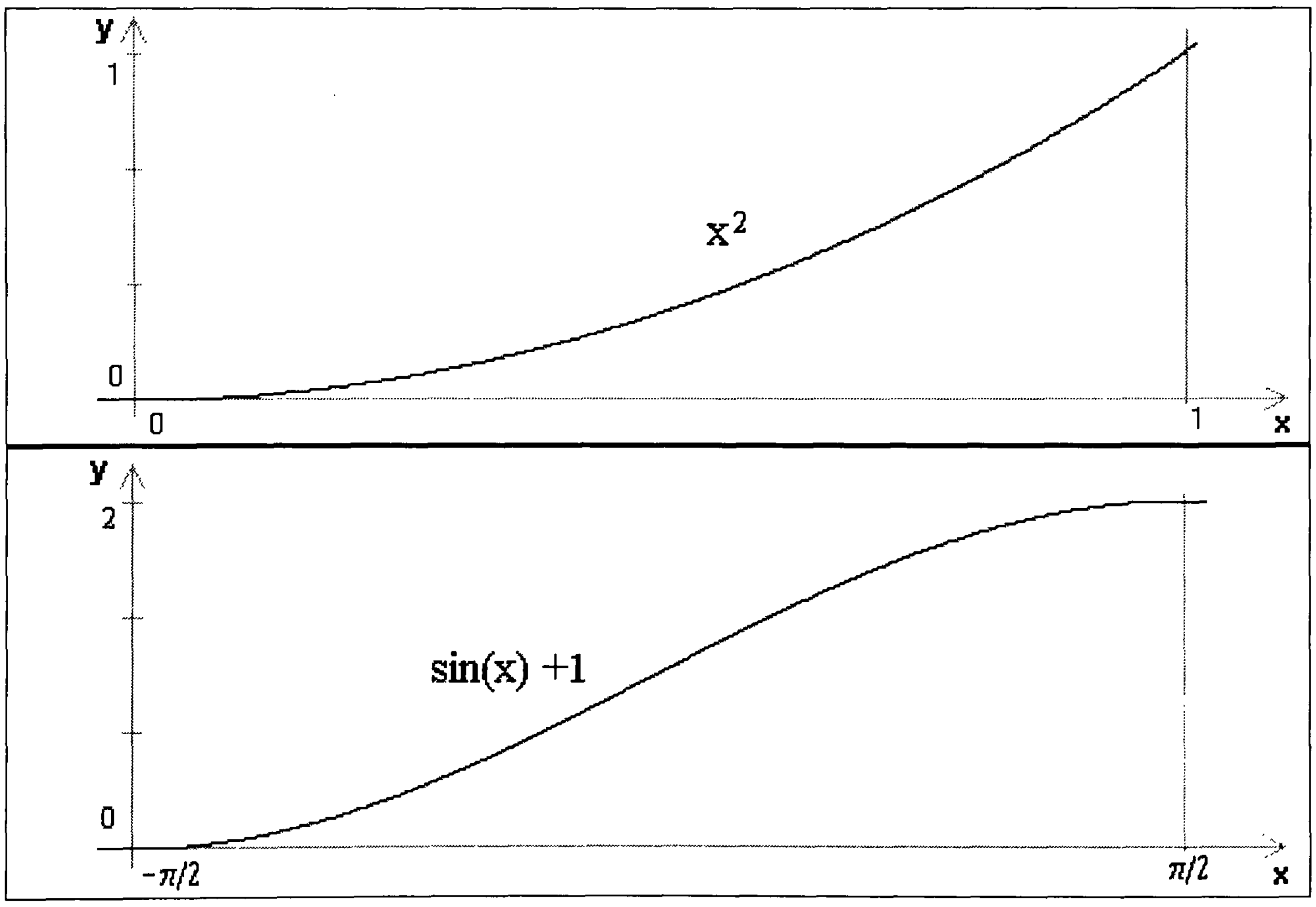


Figure 5-6 Different weighting functions considered

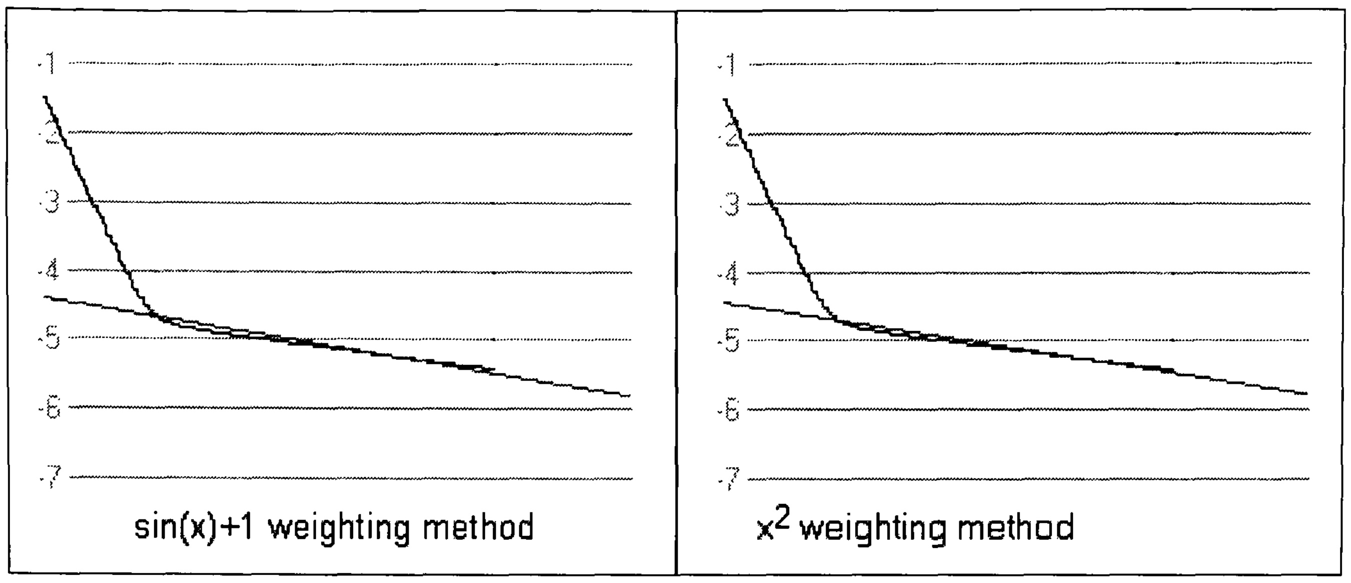


Figure 5-7 Comparison of different weighting functions

The weighting function is normalised before each approximation to ensure that it covers all sample points. The results of using these functions as modifiers to least-squares approximations are shown in Figure 5-7. There is little difference between the two predictions but after analysing several different graphs and the corresponding approximations, the x^2 function was selected as more appropriate. Each function provided a slightly different ratio of stability vs. accuracy, however the final choice was largely influenced by personal preference.

This completed the development of the convergence metric. The convergence speed is measured as a number of sweeps required to complete one second of the simulated time. The number of sweeps is acquired from the residual graphs directly or, if this is not possible, from the extrapolation using the weighted least-squares method.

5.4.2 Irregularities

It is difficult to define precisely what is an irregularity. There are many diverse features in the residual graphs that are broadly classified by experts as *irregularities*. Virtually any part of the graph that does not exhibit a consistent and downward trend can be described as irregular, which means that designing a universal assessment technique is rather difficult. Fortunately, most of the irregularities share a common trait – they contain a section with an upward trend. In a perfect residual graph all the residuals are steadily decreasing and therefore it is reasonable to assume that anything different would indicate convergence problems and therefore can be classified as an irregularity.

Due to the diverse set of irregularities encountered, any attempts to make the definition more precise while still keeping it generic would fail. Nevertheless, it is generally agreed that the upward trend is an intrinsic part of almost all irregularities and a technique that could assess the duration and the magnitude of such trend would be an adequate estimator of the graph's irregularities. Consequently, the resulting method relies heavily on this particular property. The amount of irregularities in the graph is measured with a single value that reflects both the magnitude and the duration of the upward trend. This algorithm also has a very useful property of discarding small fluctuations that do not constitute an irregularity while at the same time preserving all significant changes in the monitored value.

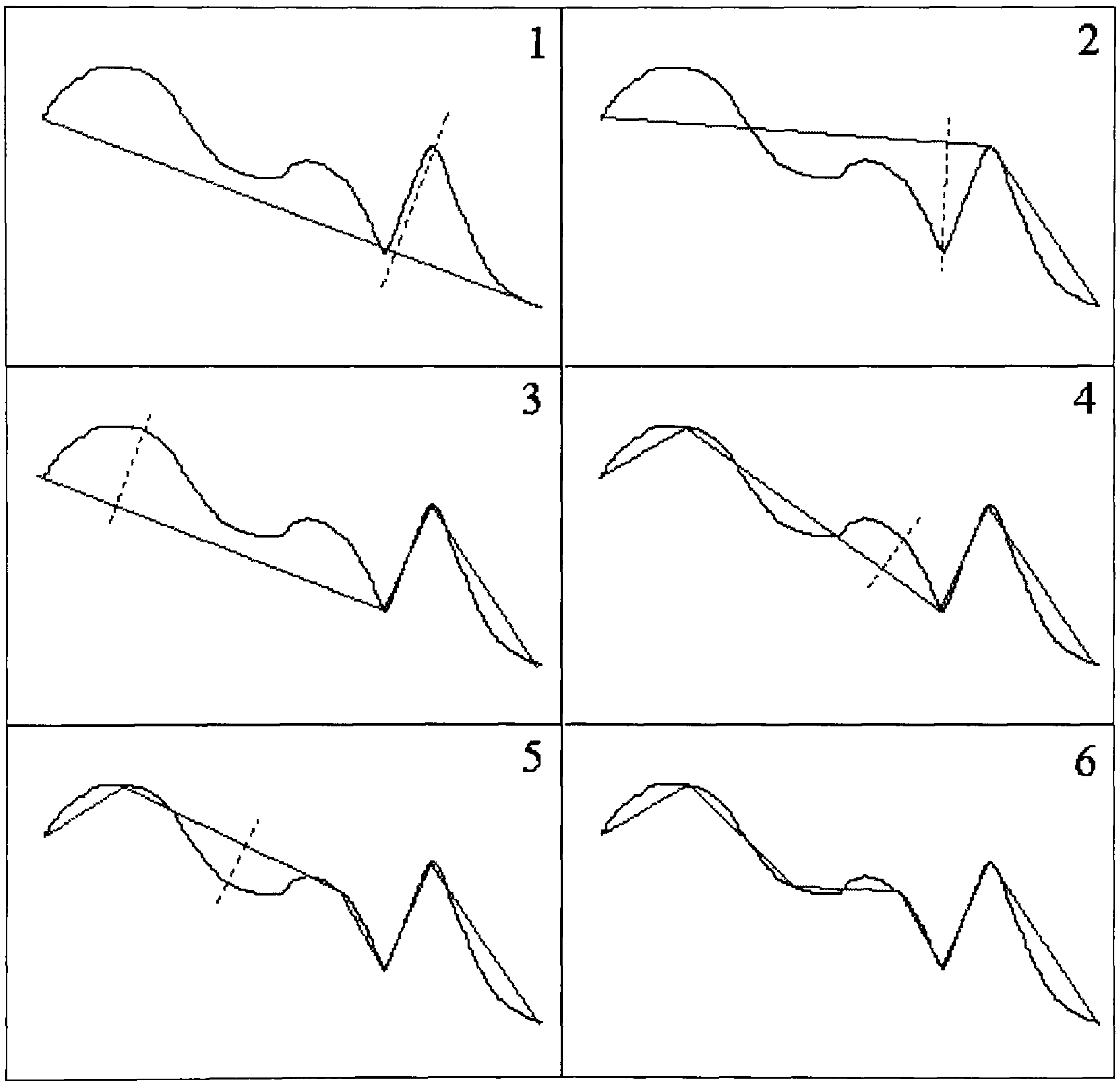


Figure 5-8 Method of approximation with line segments

The assessment method employs a special algorithm commonly used in computer graphics for curve vectorisation (Le Riche-69, Reuman-74). Depending on the required accuracy, this algorithm can either retain only the most significant features of the curve or produce an approximation, which is virtually indistinguishable from the original. It employs an iterative procedure that gradually reduces the errors present in the approximation until the required precision is achieved. Figure 5-8 shows an example curve and the first 6 consecutive steps of the approximation algorithm.

In the first step of the algorithm the two ends of the curve are joined by a single line segment (we are assuming that the shape approximated is a continuous curve with separate ends). Next, the curve is analysed in order to find the point that lies the furthest from the approximating line (or lines). The line is then split into two segments to include the point with the biggest approximation error. This procedure is performed repeatedly, producing more line segments and better approximations, until the maximum error (distance from the curve to the set of approximation lines) drops below the predetermined tolerance. In Figure 5-8 the dashed lines show the points with the maximum approximation error. Unlike many other techniques, this algorithm tracks all major changes in the monitored values and does not flatten steep spikes, which makes it very well suited for irregularities detection.

This procedure produces a set of lines approximating the graph but does not explicitly assess any irregularities. The resulting lines have to be analysed further in order to obtain the required assessment. The method devised for this purpose is based on the assumption that both the magnitude and the duration of the irregularity are equally important. The procedure estimates the irregularities by computing the area underneath the segments with the upward trend. Consequently, the complete assessment algorithm consists of the following steps:

- Approximate the graph with line segments up to the required tolerance (the tolerance determines the threshold below which the irregularities are considered irrelevant).
- Identify all continuous sections in the approximation (consisting of one or more line segments) that contain upward trend only.
- Calculate the area underneath the sections with upward trend.

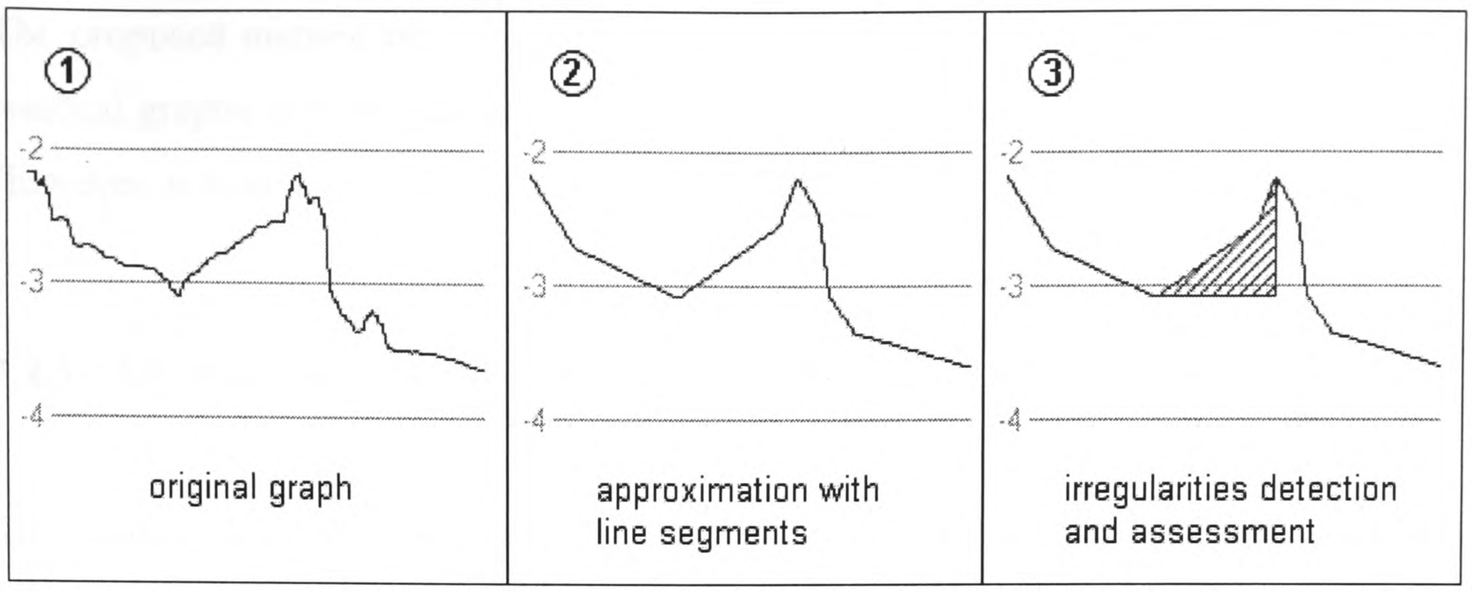


Figure 5-9 Irregularities assessment

The area underneath the relevant segments is an estimator for the amount of irregularities present in the graph. Usually, the total area is averaged over the whole graph to represent a uniform measure of irregularities per single sweep. The algorithm is demonstrated on an example residual graph in Figure 5-9.

There are a few instances where the assessment result does not reflect the magnitude or duration of irregularities correctly. Figure 5-10 presents two examples where the assessment is not very precise. The spike in Graph A is quite substantial but due to the very steep upward slope its assessment will not fully reflect this. The irregularity from graph B will also be under-emphasised, as it will be evaluated as a number of small irregularities rather than a major single one. On the other hand, the saw-teeth irregularity can be classified as oscillation and consequently, will be assessed by a different algorithm.

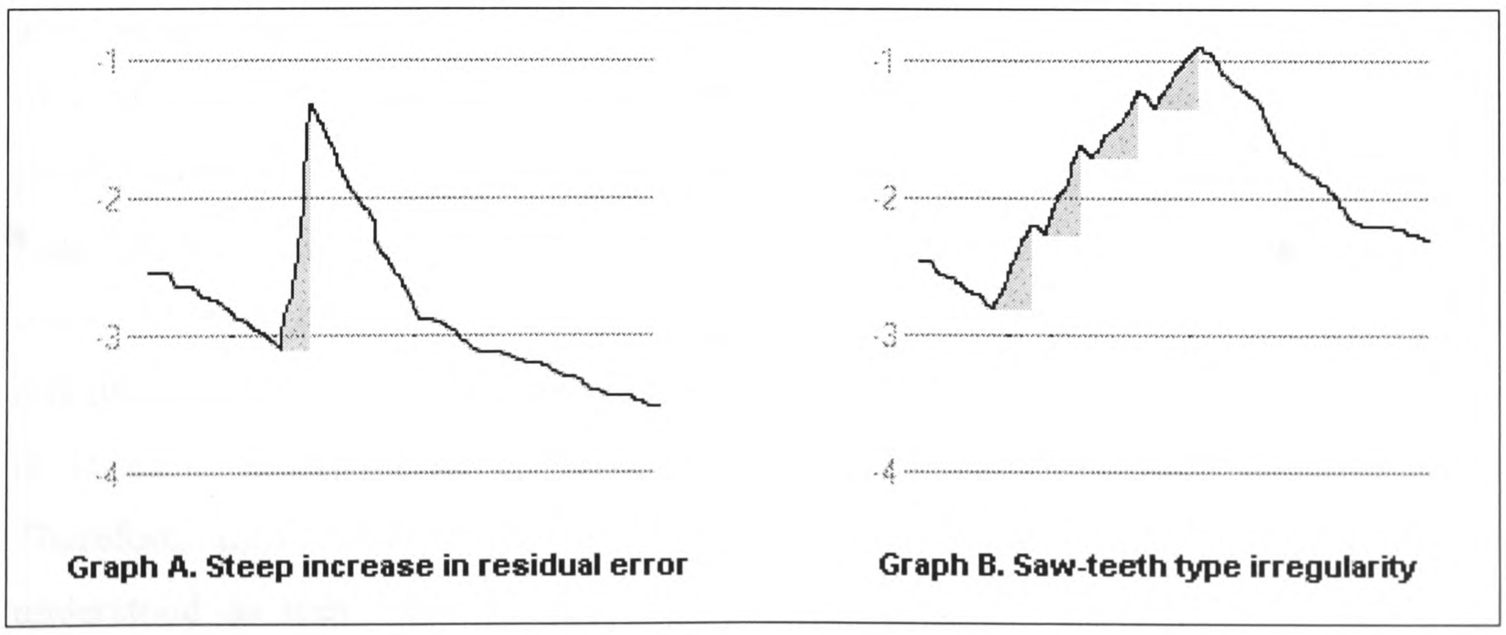


Figure 5-10 Graphs with difficult to assess irregularities

The proposed method for irregularity assessment was validated on an extensive set of residual graphs and, despite the limitations, its accuracy was confirmed as satisfactory. However, it is believed that this technique could benefit from further research.

5.4.3 Oscillations

The next feature that was deemed important by the experts was oscillation. The residuals always contain various fluctuations but only some of them are relevant to the overall assessment. Special care was taken to ensure that only real oscillations were picked up by assessment algorithm. Consequently the following criteria for identifying significant oscillations were proposed:

- The amplitude of the oscillation must be larger than a predetermined tolerance.
- The amplitude must remain above the tolerance for at least two full oscillation cycles.

The amplitude constraint is very important since oscillations are very common in residual graphs but only the ones with substantial amplitude convey important information about the simulation state. The purpose of the limit imposed on the minimum duration is to avoid classifying major but isolated irregularities as short-duration oscillations. Consequently, an effective method for oscillation extraction and assessment should detect periodic fluctuations with stable frequency and with duration and amplitude bigger than pre-determined tolerance.

The resulting algorithm for oscillation assessment is based on the Fourier Transform and spectral analysis. This mathematical tool is very effective in analysing various types of oscillations. Furthermore, the problem lends itself nicely to Fourier analysis since the graphs are made up of discrete, evenly spaced points, which facilitates the use of the Fast Fourier Transform algorithm. The Fourier Transform plays an important part in the theory of many branches of science but even a brief overview is well out of scope of this dissertation. Interestingly enough, many scientists know the Fourier Transform not in terms of mathematics, but as a set of propositions about physical phenomena. Therefore, most terms, which are common in Fourier analysis, are often easily understood as they have a clear physical representation. Consequently, no formal introduction will be presented here and the discussion will rely on intuitive understanding of its features. There exists a very extensive literature about the Fourier Transform, which should be consulted if any theoretical details are required. (65-Bracewell) provides approachable theoretical overview with strong emphasis on applications. Masters (Masters-95) presents a very basic but intuitive description of digital filters and spectral analysis for time series predictions. Several concepts from that book serve as an inspiration in the design of the oscillation detection algorithm. A somehow more detailed treatment of similar topics can be found in (Masters-94). A very popular book, 'Numerical Recipes in C' (Press-92), describes the Fast Fourier Transform algorithm and the associated C++ library that was used as an underlying FFT engine in the ICS implementation.

In brief, the oscillation detection algorithm uses spectral analysis to identify the frequencies that may correspond to significant oscillations of the residuals. Then, the phase-shifting filters are applied for each potentially relevant frequency to investigate whether the amplitude and duration constraints are satisfied. The algorithm produces a number, which represents the length of the graph section that contained significant oscillations (zero if none was found).

The first step of the algorithm performs de-trending. The main purpose of this procedure is to remove the low frequency components that would dominate the frequency spectrum since most of the graphs contain a steady downward trend. This problem is well illustrated in Figure 5-11 where the power spectrum contains a very prominent peak at low frequencies, which obscures higher frequency oscillations.

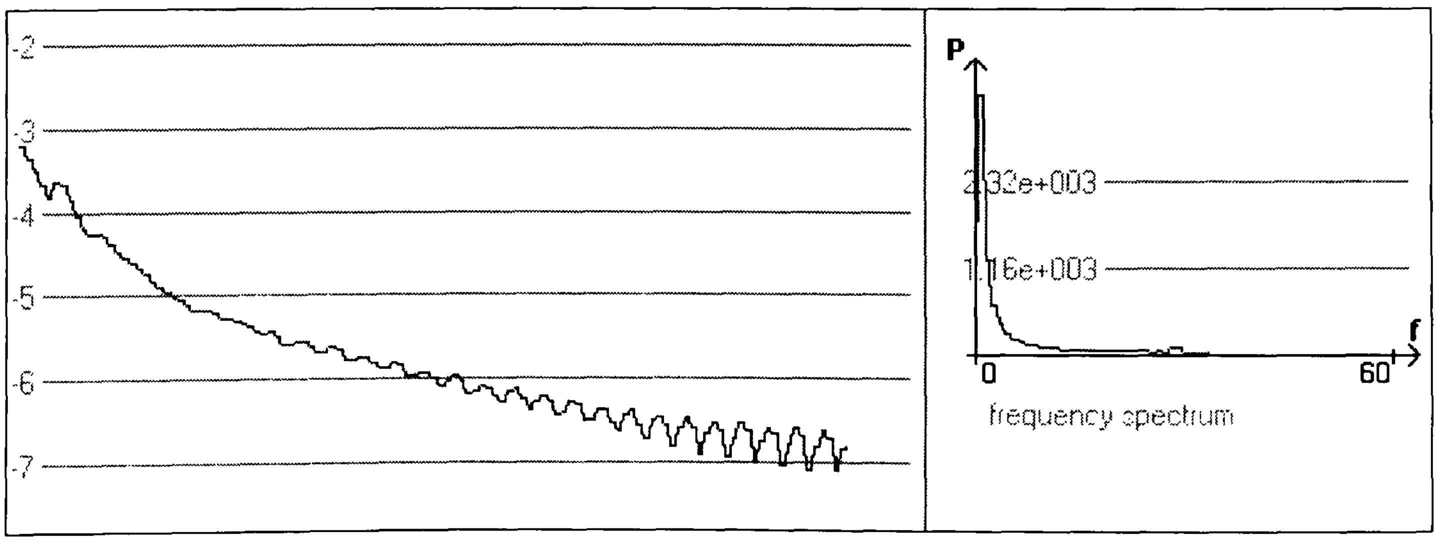


Figure 5-11 Original residual graph and its frequency spectrum

The graph does contain important medium-frequency oscillations but since most of the energy in the spectrum is grouped around the very-low frequencies the higher frequencies cannot be easily identified from the spectrum. The energy of the global downward trend is substantially higher than the energy of the relevant oscillation and therefore the peaks representing the higher frequencies are not visible. Consequently, the graph must be de-trended before it can be further analysed. Several simple methods for de-trending were examined. A method that involves subtracting a fitted polynomial trend function from the graph was initially considered but the accuracy of the approximation varied and most of the time the de-trended graph retained a large portion of low-frequency components. Further tests confirmed that the best de-trending procedure involved digital filtering. The original graph is filtered using a broad high-pass filter that removes only the very low frequency oscillations preserving the relevant medium and high frequency components.

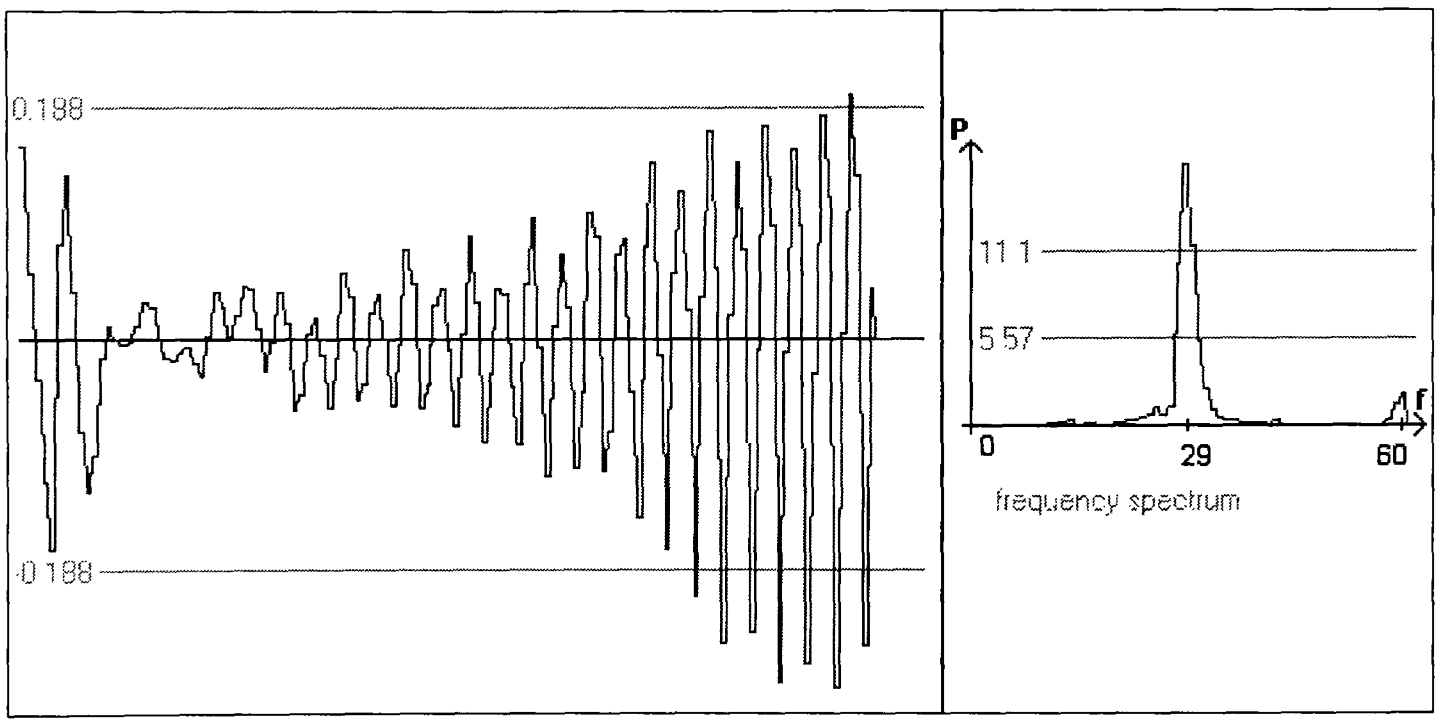


Figure 5-12 Results of applying a high-pass filter

After applying the filter the graph is centred to make its average equal 0. This helps remove the remains of the global trend that might have been spared by the high-pass filter. The final results are shown in Figure 5-12. With the global trend removed, the medium-frequency oscillation is very clear and the corresponding power spectrum also shows a very strong peak around the relevant frequency. The spectrum also shows another, substantially smaller peak, which represents twice the frequency of the main oscillation. This reflects the fact that the oscillation consists of several different sinusoidal components.

After de-trending, the next step of the algorithm is spectral analysis. It is easy enough to informally describe how to detect the significant frequencies (all the peaks that "stand out" in the power spectrum may indicate important frequency) but formalisation poses some problems. Firstly, the peak only indicates that the particular frequency contributes substantial amount to the entire energy of the graph. Such peak can also be the result of a strong irregularity present in the graph, which does not necessarily amount to oscillation. It can also represent some part of the global trend, which could not be successfully removed by de-trending. This situation is very common if there is no oscillation present in the graph since in that case the energy is usually grouped around the low-frequency band. Consequently, strong peaks only identify potential frequencies but do not resolve whether the particular peak represents an oscillation, which satisfies other requirements (amplitude and duration constraints). Further analysis is therefore required to confirm whether a particular frequency is linked to a significant oscillation in the graph.

The strong peaks in the spectrum are identified using a very simple technique. First, an average for the whole spectrum is calculated and the peaks that are significantly larger than the average indicate a possible strong oscillation. Only real peaks are considered, i.e. local maxima. An example spectrum is presented in Figure 5-13. There are three peaks that clearly stand out in the spectrum and therefore are likely to represent strong oscillations of long duration. Still, the corresponding frequencies must be further examined before any conclusions can be drawn.

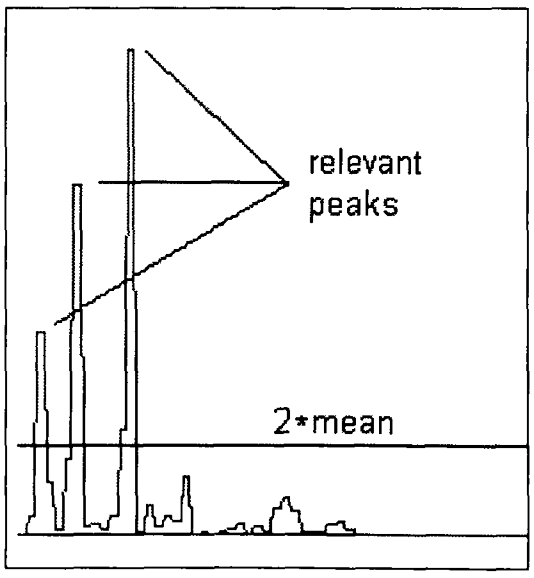


Figure 5-13 Result of spectrum analysis

Let us return now to the initial example from Figure 5-11. The graph was de-trended, the frequency spectrum analysed and it was established that there were peaks at f=29 and f=60. Having identified the peaks, the next step is to determine whether the related oscillations satisfy the constraints of minimum amplitude and duration. The amplitude and duration is estimated using phase-shifting filters. Again, an overview of the phase-shifting filters is beyond the scope of this dissertation and therefore we will only concentrate on the features that are essential to our application. More details about phase-shifting filters can be found in (Masters-95) or (Masters-94).

The algorithm that examines the potentially significant frequencies relies on a very useful feature of a certain type of digital filters – the ability to detect periodic events. Two band-pass filters are applied to two separate instances of the same graph (a phase-shifting filter and a standard in-phase filter). The filters are specifically constructed to focus on a specific frequency (such a pair of filters is commonly known as quadrature-mirror or QM filter pair). This creates two outputs: in-phase and in-quadrature (In-quadrature means that the filtered output is shifted by $\pi/2$ with respect to the input). The outputs are then combined by calculating at every point a square root of the sum of the squares of each output. The result is an estimate of the amplitude for the chosen frequency at each point of the graph.

This procedure is applied to every frequency identified as potentially important during spectral analysis in order to estimate its amplitude and duration. These two factors determine whether the graph does contain the oscillation. The example assessment is shown in Figure 5-14 and Figure 5-15 (the minimum amplitude threshold was set to 0.03). It is clear that these two frequencies are components of the same oscillation but again, this is not important to the assessment.

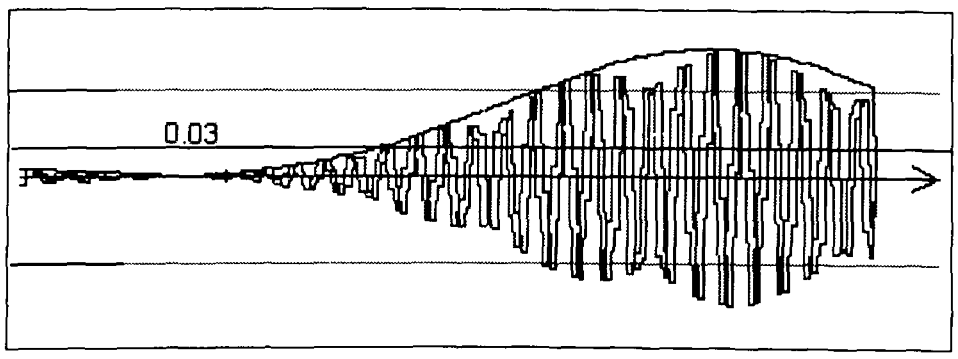


Figure 5-14 Oscillation assessment for f=29

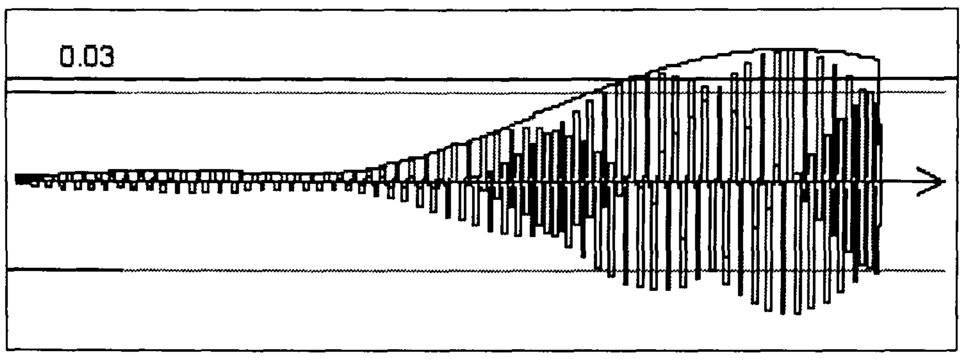


Figure 5-15 Oscillation assessment for f=60

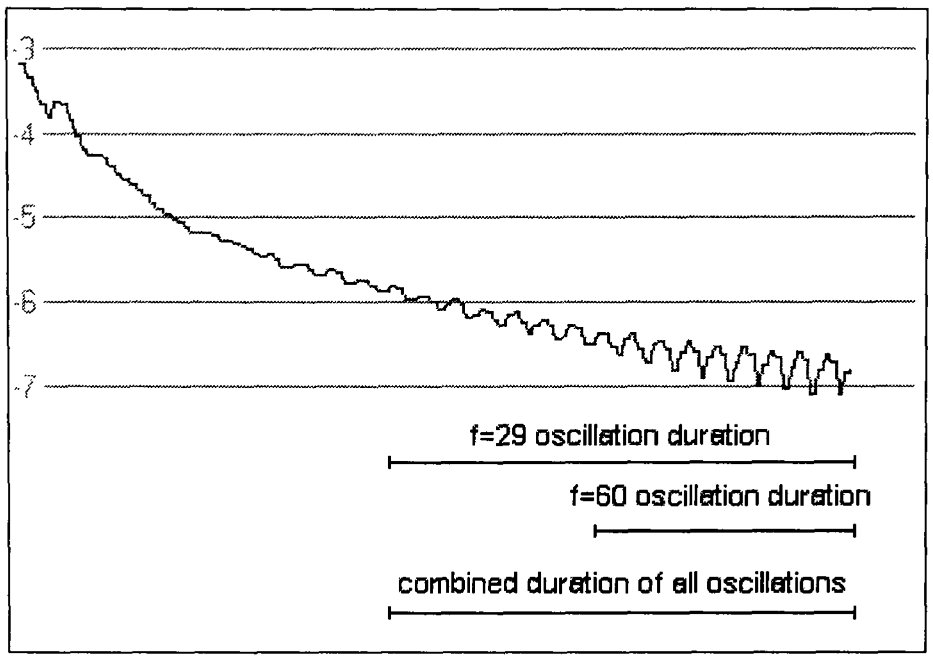


Figure 5-16 Final results of the oscillation extraction

The final result of the assessment is presented in Figure 5-16. The duration of each frequency is shown but (as such distinction is not necessary for out purposes) the final result only includes the combined duration of all the detected frequencies.

The complete algorithm for oscillation detection is presented here:

1. ***Apply the high-pass filter to discard low-frequency components (global trends).***
2. ***Estimate the frequency power spectrum using FFT.***
3. ***Analyse the spectrum to detect distinctive peaks that indicate the presence of significant oscillations.***
4. ***Apply QM filters to isolate potential frequencies (one at a time). The filter produces two outputs – in-phase and in-quadrature.***
5. ***Combine the outputs from QM filters to estimate the amplitude.***
6. ***Check if the amplitude is greater than the predefined threshold.***
7. ***Check if the amplitude stays above the threshold for at least two full periods of the analysed frequency.***
8. ***If both conditions are true then the graph is classified as containing this particular oscillation.***

The above algorithm was tested on a diverse set of graphs and was proven to provide sufficiently accurate assessments.

5.4.4 Compound evaluation function

Each of the algorithms described so far was designed to assess a specific feature of the residual graph that is believed to be important in the overall assessment of the simulation state. However, we have not yet established how to combine the three separate measures into a single final assessment. Unfortunately the consultations with experts did not result in the development of such a comprehensive evaluation function although some important suggestions were obtained. It is generally assumed that the irregularities and oscillations are indicative of the simulation stability. Usually, the more unstable the simulation becomes, the more irregularities and oscillations it contains. It is also understood that the convergence speed is the most important metric for measuring both speed and stability of the simulation. If the solution converges quickly then the simulation is believed to be stable. There are two major factors that can adversely affect the convergence speed:

- Using a conservative set of control parameters (the simulation is stable but the convergence rate is slow; reason: too much under-relaxation applied).
- The control parameters are too relaxed (simulation unstable; reason: too much relaxation and/or too big time step size).

The optimum set of relaxation parameters should provide the best convergence speed while maintaining adequate simulation stability. It initially seemed reasonable to use convergence speed as the only component of the compound evaluation function. However, this approach was deemed too simplistic. The convergence speed is primarily an estimate of the execution time and therefore the resulting evaluation function would primarily be concerned with reducing the simulation time. This may seem perfectly fine but one has to remember that the architecture adopted for the ICS relies on performing the search infrequently as it is very costly. Maximising the time performance always increases the danger of divergence and divergence recovery is computationally expensive. The recovery also has an adverse effect on solver performance, which lasts for several time steps. Frequent searches use substantial resources and can therefore delay the simulation. A very aggressive, performance-oriented control system may recommend control parameters that are suitable only for a single time step and which would trigger divergence (and consequently a recover procedure) in the next step. Hence, an assessment based exclusively on the convergence speed could have the opposite effect to the one intended. This is a very common problem associated with simplistic evaluation function. Most researchers tend to use heuristics that provides a balance between a "goal-attraction" and "problem-avoidance". The importance of choosing a balanced heuristic can be well illustrated using the previously introduced 'navigating robot example'.

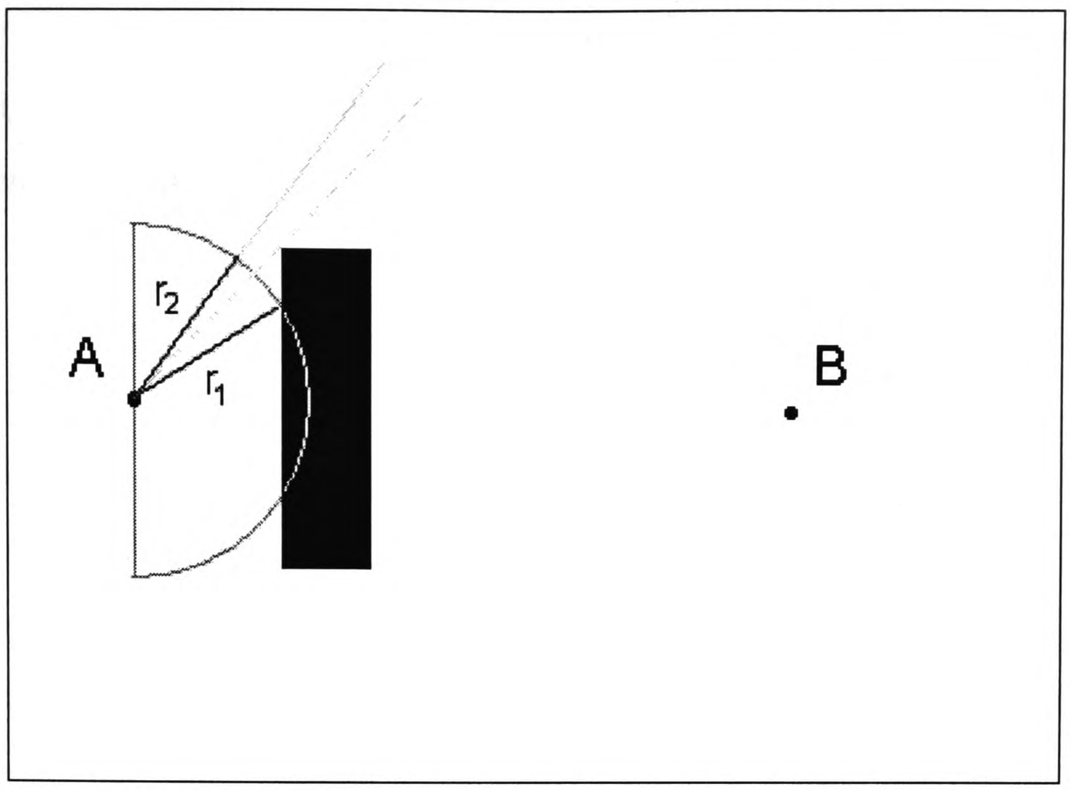


Figure 5-17 Avoiding a large obstruction

In Figure 5-17 there is our familiar robot trying to get from A to B. However, there is a big obstruction in front of it and the agent has to determine the best route forward to avoid it. The robot should employ its vision system infrequently since it uses a lot of time and energy. Its field of vision is presented as a semicircle. Now, if the assessment function only considered the speed at which the robot approaches the goal (point B) then the route chosen would be r_1. Having seen the whole picture, this decision is obviously wrong since it leads to bumping into the obstacle. However, the robot only sees as far as its sensors allow and therefore this path may look sensible to the robot. This example demonstrates the dangers associated with an overly simplistic evaluation function. The agent would be much more successful if it analysed the input from the vision system more thoroughly and determined that the obstruction was likely to extend beyond its vision range and that it might be safer to take a longer route but with more chances of success. The route r_2 is clearly a much better choice. It is not optimal (the optimal path is shown with a dashed line) but the overhead is small compared with the cost of bumping in the obstacle, having to engage the vision system and performing an obstacle avoidance manoeuvre. This is a very basic example but it serves its purpose of demonstrating dangers associated with choosing a very simple evaluation function. A more complex case of a flying robot can be found in (Elnagar-95) where the authors proposed a heuristics that was based on a sum of obstacle-repulsion, goal attraction and level attraction (a certain altitude was optimal for robot operation). In (Inoue-91) a collision avoidance algorithm for a robot manipulator used heuristics that favoured the middle plane when navigating between two obstacles. This strategy not only allowed some room for inaccuracy in robot's movement but also minimised the probability of a collision when any of the obstacles moved.

Returning to our problem, in a CFD simulation the presence of irregularities and oscillations is believed to indicate the proximity of the simulation to serious problems (i.e. divergence). Consequently, a successful evaluation function should provide a balanced assessment of the speed and stability of the simulation. Unfortunately, initial attempts to design such function failed. Several functions were created and while some of them produced acceptable results, their design was based on a guess rather than real understanding of the underlying priorities and therefore it was difficult to predict how they will behave in real simulations with a broad range of complex graphs.

Nonetheless, the work on the evaluation function produced some very interesting conclusions that enhanced our understanding of the assessment method. The most important one states that an attempt to perform an absolute assessment of the graph quality (i.e. outside of the simulation context) is bound to fail. The evaluation function must be based on a comparison of graphs from the same simulation. A single residual graph assessed as very bad in one simulation can be acceptable in another. Fortunately, a relative assessment function is perfectly sufficient to create an effective control system.

Unfortunately, at that stage it became obvious that the development of the suitable compound evaluation function required further research. It was therefore decided to defer finding such a function and to build a control system using one of the prototype compound function already developed. The chosen function, which was not modelled on a human assessment technique, was expected to produce non-optimal evaluations but it facilitated building a complete control system. It was hoped that such a system would provide more experimental data and would therefore assist in the development of a better and more accurate technique for compound assessment.

The prototype function became quickly superseded in further work and therefore a very brief overview is provided here. Only converging graphs were evaluated (diverging graphs were immediately rejected). Each type of the three important graph measures (convergence speed, irregularities and oscillations) was assessed separately and a total score for each graph was calculated using a rather convoluted algorithm. The graph with the highest compound score was assessed as the best one. The assessment process is presented in Figure 5-18.

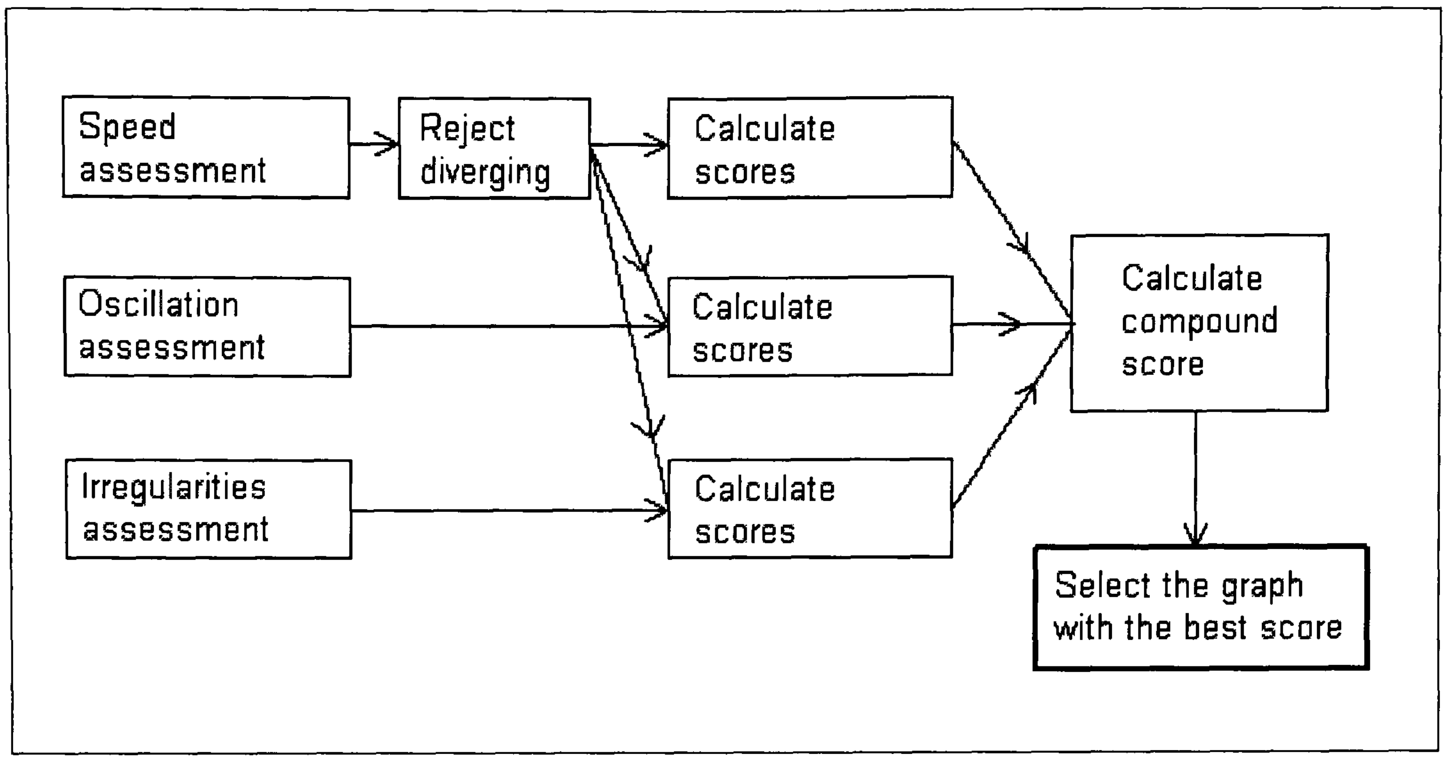


Figure 5-18 Algorithm for calculating the compound evaluation function

5.5 Fault detection

In the proposed control architecture the heuristic search is performed either at regular intervals or when serious problems develop. Consequently, the problems have to be properly detected in order to initiate the search at the correct time. Before the design of the automatic fault detection algorithm could begin, the nature of simulation faults has to be properly understood and classified. Fortunately there was little ambiguity in this matter and the following list of common problems emerged:

- **Divergence** – this problem occurs when the numerical solvers fail and the subsequent approximations proceed in the wrong direction and therefore do not get closer to the correct solution. Divergence is usually caused by too big a time step size or too much relaxation. A non-continuous event (e.g. removing a partition or the second fire igniting) occurring within the simulated domain can also produce divergence. Very often it is difficult to identify the exact cause of the problems and therefore the divergence is easier to detect than to predict or prevent. If the solution diverges then the simulation results are assumed to be incorrect.

- **Major convergence problems** – similar to divergence but less serious. The result from the simulation experiencing such problems can be either wrong or close to the correct solution but it is usually difficult to determine the actual magnitude of the error. It is therefore safer to assume that the results are wrong.
- **Slow convergence** – the solver takes a very long time to achieve convergence (compared to the preceding time steps). The results are correct or close to being correct. Slow convergence can be caused by physical events occurring in the domain but it can also indicate problems with the problem set-up: unrealistic convergence limit or bad quality mesh.
- **Oscillations** – the simulation results are not necessarily affected but the presence of oscillations often indicates some underlying problems (e.g. bad mesh quality) that can lead to inaccurate results or performance deterioration.

The problems listed above should be detected automatically in order to trigger corrective actions. It was determined that there was no need to frequently monitor the progress of a time step since these problems persist and can be detected at the end of a time step. The design of the fault detection algorithm was strongly influenced by this assumption which made the whole process relatively easy. The resulting procedures are straightforward and based on the feature extraction algorithms described earlier.

Divergence can be detected using the extrapolation technique developed for the convergence speed assessment. The extrapolated section of the graph is analysed and if there is a steady upward trend and the residuals continue to increase then the time step is clearly diverging. This assessment is performed when the number of sweeps allocated for the time step is exceeded and the residuals fail to converge. Most convergence problems manifest themselves in a very similar manner and therefore the divergence detection technique is also used for identifying convergence problems. Even the corrective actions are very similar in both cases. In fact, some experts do not distinguish between divergence and major convergence problems. However, separating these two helps emphasise that divergence always produces wrong results while a time step with convergence problems can sometimes be correct.

Slow convergence can also be conveniently detected when the allocated number of sweeps is exhausted. If the time step does not converge then the remaining number of sweeps to convergence is predicted using the extrapolation method. If the number of sweeps required is small then additional sweeps are allowed and (providing that the time step subsequently converges) everything proceeds normally. However, if the estimated number of sweeps is excessive then some corrective action is necessary to improve the performance.

Oscillations are not considered a threat to the result accuracy but they usually indicate some underlying problems and can seriously impair performance, therefore it is often beneficial to apply some adjustments. The method for oscillation detection has already been developed for the evaluation function and is also used here. The oscillations are assessed at the end of the time step when it is determined whether an excessive amount of oscillations was present in the graph, in which case the heuristic search is normally initiated.

Example graphs that experience the problems mentioned in this section are presented in Figure 5-19.

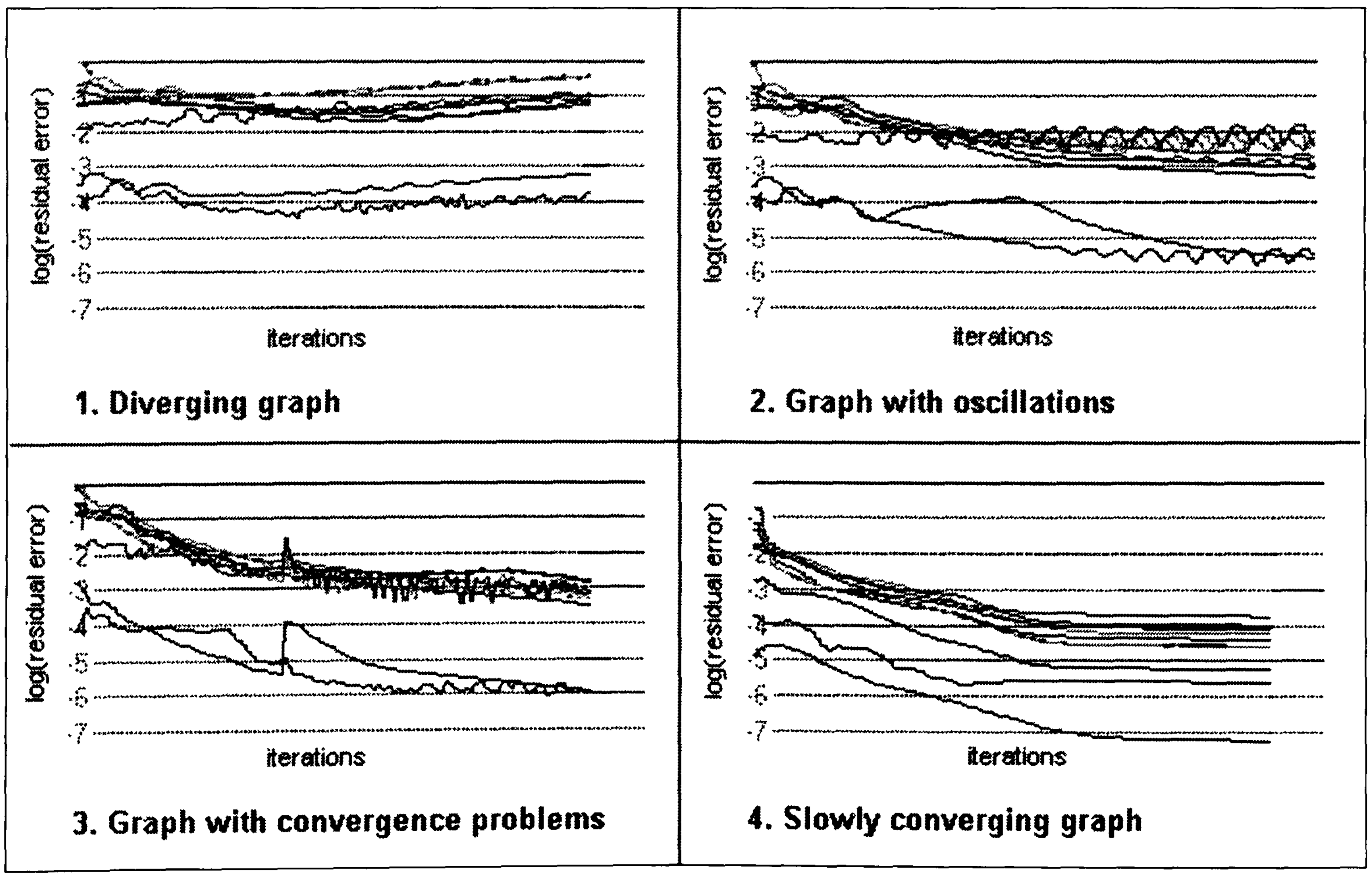


Figure 5-19 Example residual graphs reflecting specific simulation problems

5.6 Summary

This chapter describes the research process that led to the development of a new control system. A new control architecture based on a heuristic search is proposed and fully explained. It is complemented by the development of the heuristic evaluation function, which employs three specialised feature extraction algorithms. New algorithms for convergence prediction, irregularity detection and oscillation assessment are proposed. The benefits and potential shortcomings of these algorithms are briefly discussed. This chapter also explains why it was difficult to design the compound evaluation function. Finally, a technique for automatic fault detection was introduced together with an overview of common simulation problems.

Chapter 6

Tests of the prototype system and analysis of the results

6.1 Introduction

The extensive analysis and knowledge acquisition described in two previous chapters made it possible to develop the next prototype control system. This chapter presents the first version of Intelligent Control System (ICS) that uses the evaluation function and the algorithms described in Chapter 5. Several significant enhancements to the initial concept are proposed and incorporated into the new system. Subsequently the improved ICS 1.0 is used to control three different fire simulations and the results are analysed and compared against non-controlled runs. We identify areas that need further improvement and present initial conclusions.

6.2 Prototype control system

The system works in two separate modes and therefore consists of two main modules. The first module is responsible for the continuous monitoring of the simulation progress, detecting problems and the overall supervision of the simulation. This module (*Monitor*) is engaged when the simulation is progressing normally and the solvers are producing the required results. *Monitor* does not apply any modifications but performs a continuous assessment of the simulation. Adjustments to the control parameters are made by the second module (*Search*), which also performs the heuristic search. While in this mode, the simulation is not progressing but the same time step is repeatedly executed with different sets of parameters in order to determine the optimum control parameters. The *Search* module finds and applies the best control parameters and then returns the control to the *Monitor* module. Figure 6-1 shows different stages in a simulation controlled by the ICS. Initially the monitoring module is engaged and several consecutive time steps are executed. After each step its results are assessed and a decision is made whether to initiate the search module. After the step t_n the search is not performed (1) but after t_{n+1} (3) the monitoring module decides (4) to conduct experiments to find a new set of parameters. The search can be triggered either by a special scheduling algorithm or because problems were detected in the current time step. If the search is scheduled then the next experimental step is initiated but if there were problems then the current time step is restarted and used in the search. At point (5) the search for better control parameters is started. A comprehensive set of experiments is conducted and the resulting residual graphs are stored for further analysis. In the next stage (6) the results of all the experiments are assessed using the heuristic evaluation function (described in the previous chapter) and then the control parameters from the best experiment are applied to the simulation and are used in the subsequent time steps. At that point the search finishes, the monitoring module is engaged again and the simulation proceeds with a new set of control parameters (7, 8, 9, etc).

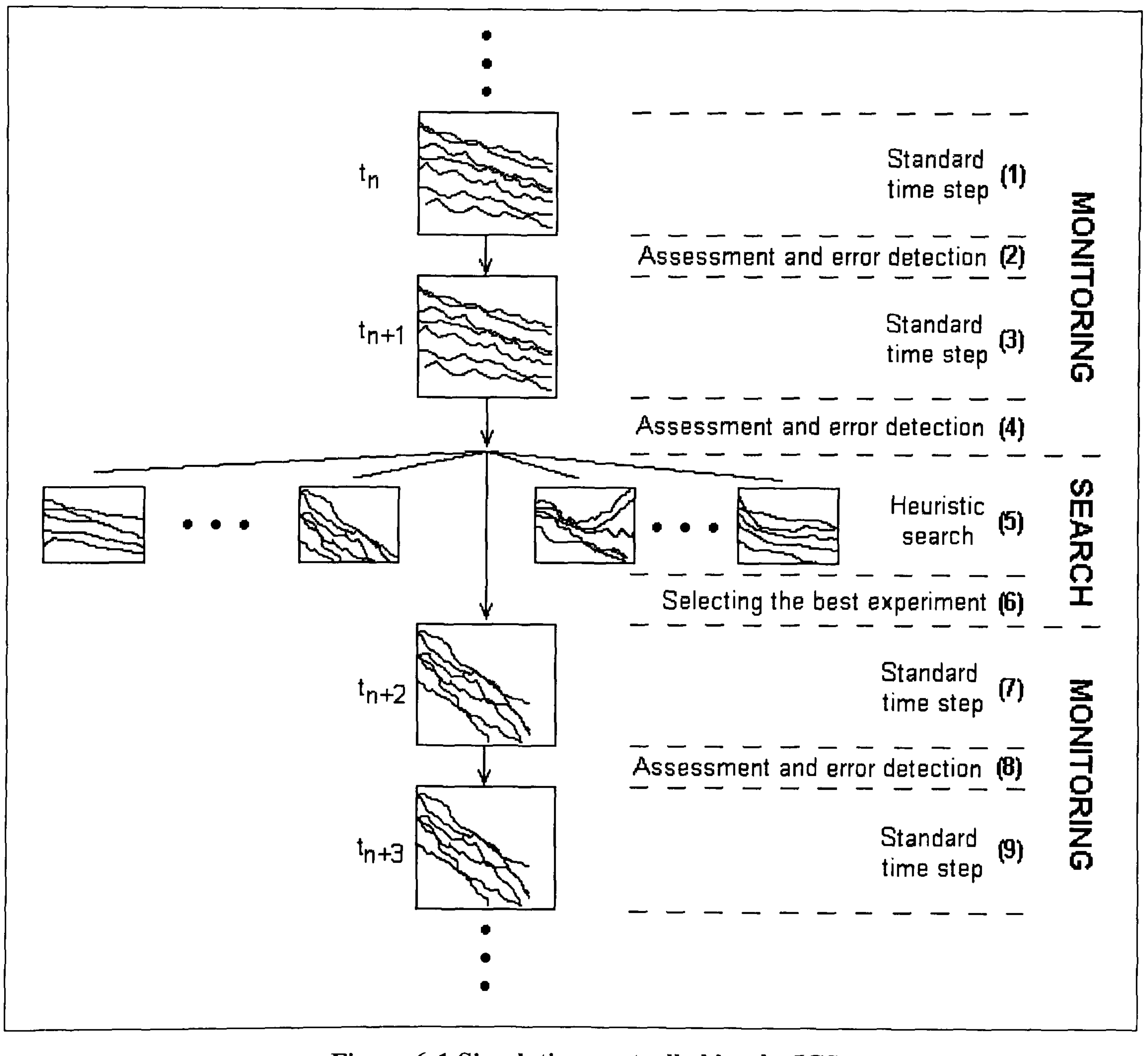


Figure 6-1 Simulation controlled by the ICS

It is worth noting that while the search is performed the real simulation is effectively stopped. During the search, the results of a time step are only calculated to produce residual graphs for heuristic assessment. Since up to 20 experiments are performed in every search, the time spent searching can be substantially longer than the time used on the actual simulation. The computational cost of a single experiment is comparable to the cost of a single time step and therefore it can be roughly estimated that if the search was performed every 20th time step then the search time would be equal the time spent for real simulation. This highlights the main problem associated with the search – a substantial increase in execution time. In the first version of the system the experiments were performed every 5th time step which resulted in significantly longer execution time. This was initially acceptable since the main purpose of the first version of the ICS was to provide a better understanding of the control techniques but it highlighted an important issue – unless the search cost is radically reduced, the control system would be impractical to use.

The actual search procedure was very straightforward. Each time it was initiated, it executed the current time step 19 times, each time using different control parameters. The changes to the parameters included reducing (by 50%) and increasing (by 50% and 100%) the time step size combined with various relaxation changes. The relaxation changes were applied uniformly to all the variables and were defined as a relative change in relaxation value (±20% and ±50%). The solution was never over-relaxed (the maximum linear relaxation was equal 1). The previous time step (not the experiment) was always used as a source of the initial parameters.

After a few tests of the ICS, it became apparent that some improvements were necessary in order to make testing of the new system feasible and practical.

6.3 Initial improvements to the original design

The improvements introduced in the first version of ICS included modifications to the feature assessment functions and some enhancement to the overall architecture aimed at improving time performance. The main changes are presented below.

6.3.1 Segment detection in convergence prediction method

It was quickly noted that the extrapolation method used did not produce very accurate results. The predicted number of sweeps to convergence was usually close to the real value but often not close enough. It was believed that a better agreement was possible and therefore an investigation into the cause of the inaccuracies was conducted. The analysis of the data revealed that the biggest differences between the predicted and actual convergence occurred in graphs with distinctive segments containing different trends. Further visual examination confirmed that although the approximation method focuses on the recent trend, old trends still significantly influence the approximation result. The problem is well presented in the left graph in Figure 6-2.

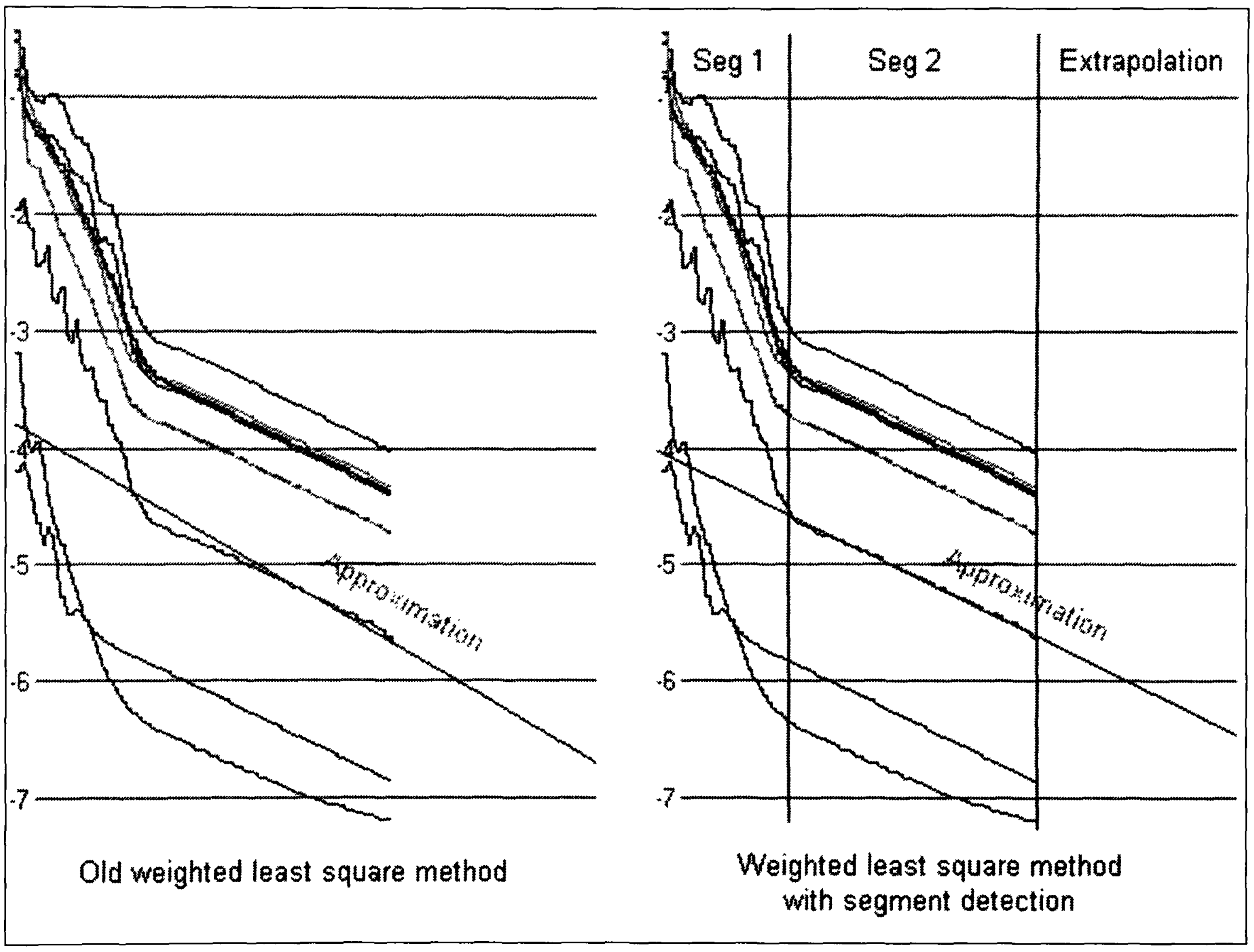


Figure 6-2 Benefits of segment detection

The residual graph displayed in the picture consists of two distinctive sections with different trends. During the time step the average trend has changed significantly and only the most recent trend determines the convergence speed in the final stage. But since the approximation method still includes the section of the graph with the old trend the resulting convergence prediction is inaccurate. The weighting method used to modify the least squares method removed most of the old trend's influence but some remained and affects the accuracy. Experts' predictions do not suffer from such problem because they are able to identify the global segments and adjust their extrapolation method accordingly. A more accurate assessment is shown in the right graph in Figure 6-2. The global segments have been identified and only the data from the most recent stage is used in the approximation. The results are clearly superior to the ones obtained with the simpler method. For a human, such an assessment is automatic and easy while a computer emulation of the same process can be quite complex. Fortunately, the appropriate algorithm was successfully developed and tested. It did not replace the old method (weighted least-squares approximation) but enhanced it by performing global segment detection prior to the approximation. Providing that the length of the last segment is sufficient then only the points from that segment are used in the subsequent approximation. If there are no distinctive segments, or the final segment is too short then the whole graph is used in approximation (exactly like in the old method).

The algorithm for segment detection is based on the analysis of changes in the average gradient over the whole residual graph. Figure 6-3 shows all stages of segment detection. Picture no. 1 shows the example residual graph. One can clearly see that the graph can be divided into two distinctive parts defined by two different trends. In the first segment the average gradient is quite steep while in the second section the graph flattens out and the gradient becomes very stable but less steep. This holds the clue to an effective segment detection algorithm – the analysis of the changes in average gradient throughout the whole graph. It must be stressed that we are only interested in global segments (present in all the variables) since single variables can contain various irregularities that must not be identified as separate sections. Only the global trend, affecting all the variables, indicates a long-term change in the convergence speed and therefore provides essential information for accurate convergence prediction.

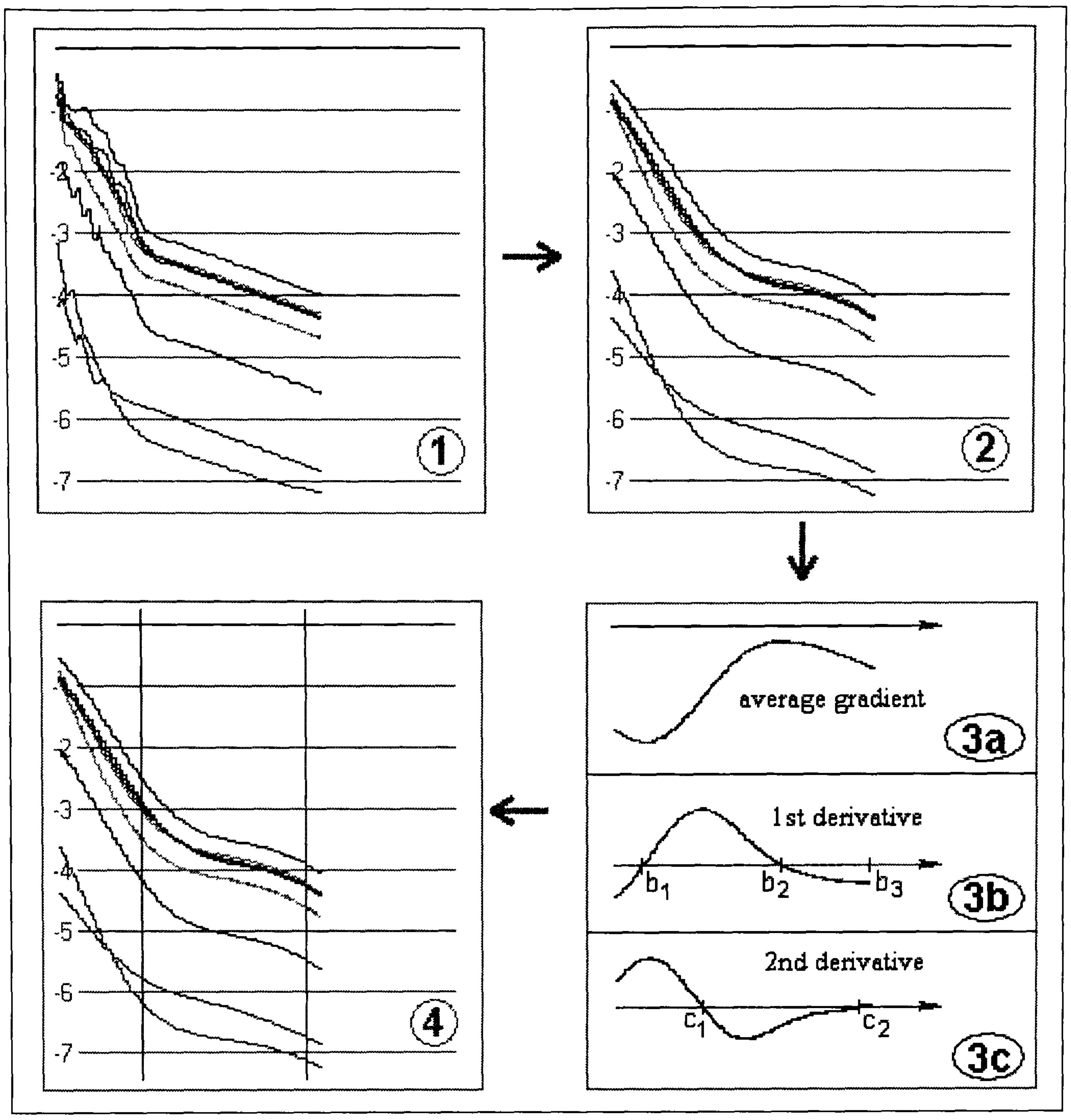


Figure 6-3 Algorithm for segment detection

Before the gradient can be effectively analysed the graph is pre-processed to eliminate all small fluctuations. A low-pass filter with very low frequency cut-off value is used for smoothing (Masters-95). This process guarantees a very simple gradient curve but introduces distortions to the residual graphs. The distortion is an acceptable trade-off as the resulting graph is very stable and the appropriate segments easy to spot. Picture no. 2 shows the graph after applying the filter, with the deformations (especially in the final section of the graph) clearly visible.

The next algorithmic step involves calculating the average gradient curve for the whole graph. This produces a curve showed in the picture 3a. It is reasonable to assume that the average gradient changes at the fastest rate when a new global trend emerges in the residual graph. Consequently, the transition point between segments was defined as the point where the gradient is changing at the fastest rate. Since the first derivative (picture 3b) represents the change of the gradient then the extrema of this function determine potential transition points between segments. Therefore, the 2nd derivative (picture 3c) is computed to find the extrema of the 1st derivative. For the function in Figure 6-3 these are found at points c_1 and c_2. The final step in segment detection is the calculation of whether the overall change in the gradient during the potential transition phase is sufficiently large to be assessed as the global transition. In our example, the overall gradient change is computed between points b_1 and b_2 (for transition point c_1) and b_2 and b_3 (for transition point c_2). If the difference is large enough than the relevant point is recognised as a valid transition point.

Picture no. 4 shows the graph divided into segments. Due to the distortions introduced by the low-pass filter an additional segment is detected in the final part of the graph. However, such segments are normally very short and therefore are easy to eliminate. In subsequent tests the prediction algorithm supported by the segment detection proved to be very accurate and robust.

6.3.2 Better divergence detection method

In some cases the procedure used for divergence detection can lead to erroneous results. The problem occurs in a certain situation when one of the residuals is steadily increasing but stays well below the tolerance level. The early algorithm classified such graphs as diverging since one of the residuals was increasing. However, it is believed that such graph should only be treated as diverging if the increasing residual is predicted to rise above the convergence level before other residuals reach it. Consequently, the algorithm for the convergence point prediction was enhanced to allow for a proper assessment of similar graphs.

The graph in Figure 6-4 is stable and can be expected to convergence within the next few sweeps as shown. However, since one of the residuals is steadily increasing, the older algorithm would assess this graph as diverging. The new improved technique can deal with such problems correctly by calculating a divergence prediction for residuals that increase but stay below the tolerance level. Both predictions are compared and if the divergence point occurs sooner then the graph is assessed as diverging. But when the convergence point is predicted to occur before the divergence then the graph is classified as converging. The experts approved this approach and described it as reflecting their own intuition.

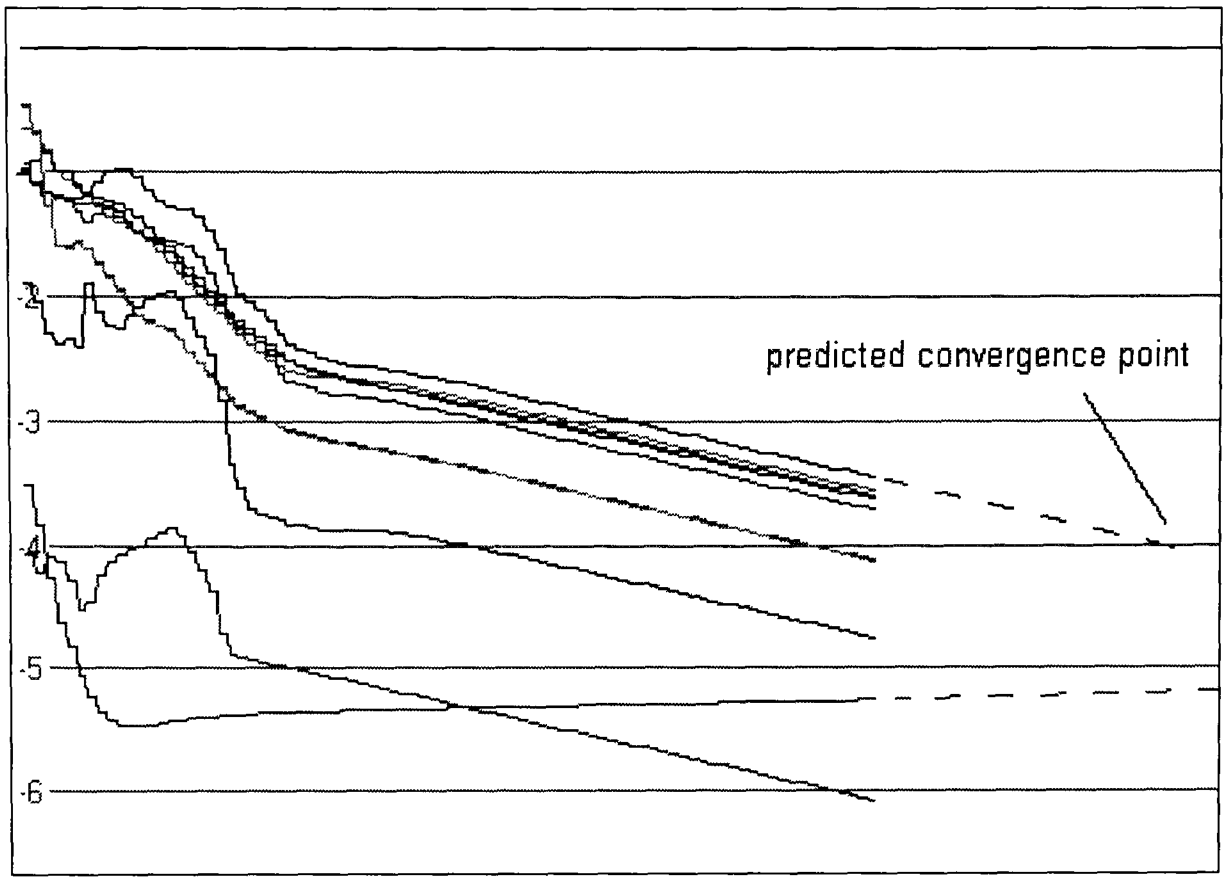


Figure 6-4 Problems with convergence prediction

6.3.3 Early detection and termination of diverging experiments

Since the computational cost of the search is significant, any technique that promises to reduce this cost is very desirable. A simple method that helps minimise the search time relies on early detection and termination of diverging experiments. In the standard search method each experiment has an allocated number of sweeps, which depends on the number required to complete the previous time step. Consequently, unless the experiment converges quickly the full number of allocated sweeps is always used. If the experiment diverges then all sweeps are still used since the results are only assessed after the time step has completed. In real simulations, the time step often diverges within the first 20-50 sweeps and then incorrect results continue to be produced. Quick divergence detection can substantially reduce the execution time and therefore the appropriate algorithm was designed and implemented.

This idea is simple – instead of a single final assessment, during execution every time step is also assessed at regular intervals. If the results of the assessment indicate that the time step is diverging then the experiment is terminated and immediately eliminated from further assessment. The assessment has to be fairly conservative to avoid terminating a converging experiment, which experiences temporary problems. In order to assure that only really diverging experiments are eliminated, at least 25% of all the variables have to diverge and this assessment must be confirmed at two consecutive assessment points. For instance, assuming that the assessment period is 25 sweeps, an experiment will be terminated only if it is assessed as diverging at two different points 25 sweeps apart (e.g. after 25th and 50th sweep). The assessment is quick and can therefore provide savings in execution time with virtually no computational overhead.

6.3.4 Enhanced search scheduling strategy

In the original architecture the heuristic search was triggered after the completion of a pre-set number of time steps. It was initially believed to be a reasonable strategy but it quickly became apparent that this method for experiment scheduling has a major flaw. The scheduling procedure did not allow for the fact that the time step size could change very substantially during the simulation. Therefore, since the scheduling was tied to the number of steps, when the time step size was halved then the experiments were conducted twice as frequently as before. For a simulation that was set up to contain 100 time steps of 1s and experiments performed every 10 steps, the expected number of searches was 10. However, if the ICS decided to reduce the time step size by 75% then the number of steps required to complete the simulation would equal 400 and the total number of searches performed would be close to 40. As a result the execution time can be significantly longer than expected. To avoid such problems, it was decided to make the search frequency dependent on the simulated time. For instance, when a period of 100 seconds was simulated then the experiments would be triggered after the completion of each 10s of the simulated time. This subtle change guarantees that the frequency of the searches performed within the simulation will be close to the pre-determined number (not necessarily equal, as the search can also be triggered by divergence). However, there is another problem associated with this approach – a substantial increase of the time step size can lead to the search being conducted after every time step. Such frequent search is believed to produce various adverse effects and should be avoided. This was accomplished by specifying a further constraint on the minimum number of time steps required between experiments (this constraint does not apply to the searches triggered by divergence). Consequently, a typical scheduling rule states: *Perform the search after each 10s of the simulated time but not more frequently than every 5 time steps.*

6.4 Speed comparison

The first version of the ICS was tested with a limited number of cases since the main goal was to perform an initial validation of the whole architecture and provide additional data to assist in further development. The results of these tests are outlined in this section.

6.4.1 54kW case with coarse mesh

The first case used in the tests was a small room (2.8m x 2.8m x 2.18m – the same size as the Steckler case (Steckler-82)) with a fire in the centre of the room, directly on the floor. The fire was defined by a growing heat release curve with the peak heat output of 54.3kW (after 50s). The room had a single vertical vent (door). The mesh was very coarse and consisted of approximately 6,000 cells. Since the mesh was very simple the overall execution time was relatively short, which made it a very convenient case for initial testing. The simulated time was 100s and the simulation used the default set of relaxation parameters at the start point.

In order to obtain data for comparison, first a non-controlled simulation was conducted and the full history of the number of sweeps required to complete each time step was stored. The final simulation state (the solution) was also saved for comparison with the results produced by a controlled simulation. The sweep history is presented in Figure 6-5. All the time steps converged apart from a single one at t=70s.

It was observed that the number of sweeps required to complete each time step was closely related to the heat release rate (or more accurately, to the change in heat release rate). Up to t=54s we can see steady growth in the number of sweeps per time step, which coincides with progressively more dynamic growth in the heat release. The heat output peaks at t =50s and does not change for the rest of the simulation. Shortly after the peak is reached, a steady decrease in the number of sweeps per time step is observed. This clearly demonstrates that the rate of change in the heat release function has significant impact on convergence speed.

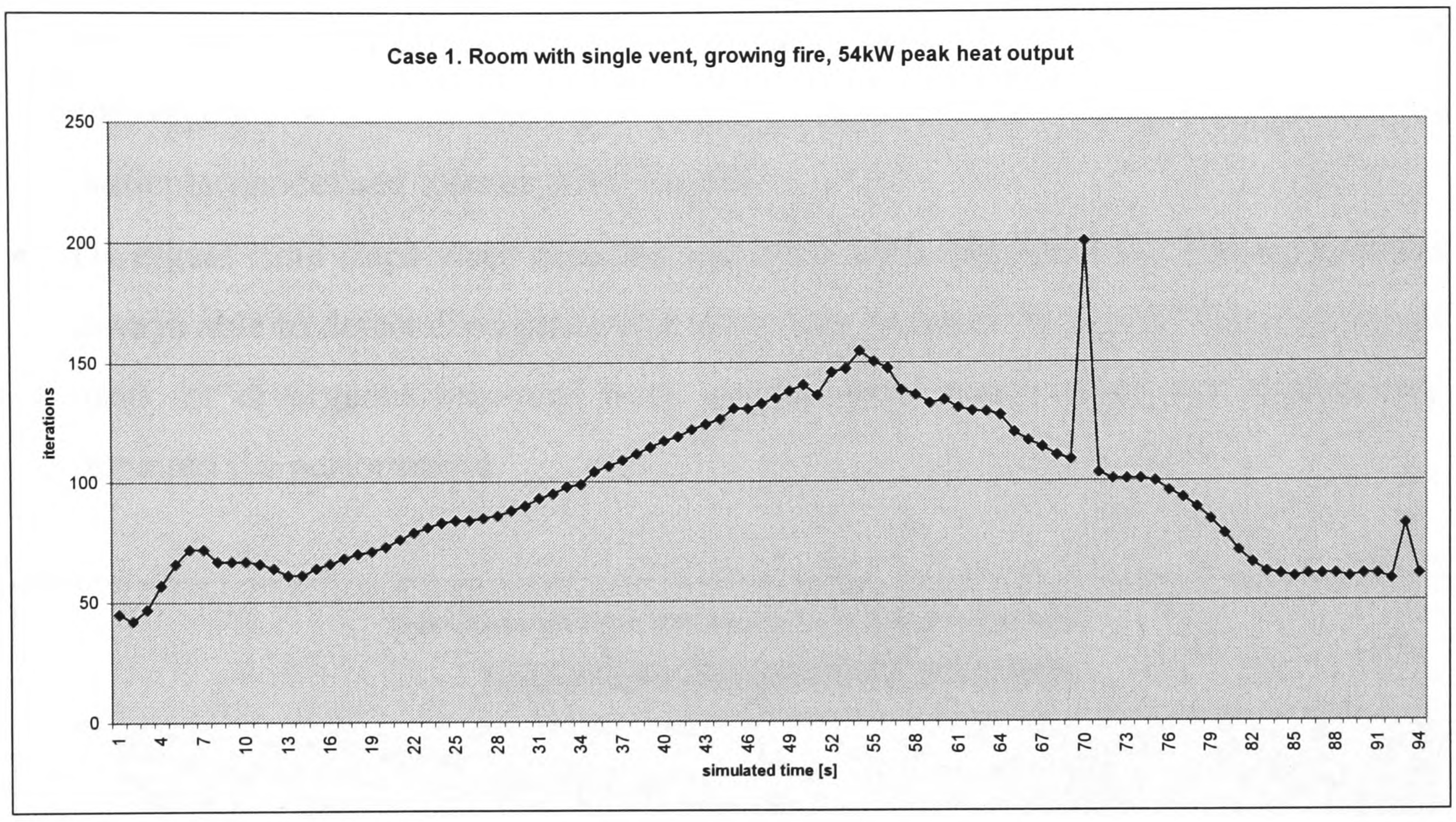


Figure 6-5 Case1, non-controlled simulation

The same case was also run as an ICS-controlled simulation. The initial set-up was the same as in the non-controlled simulation (time step size 1s, default relaxation parameters). The search was scheduled after every 5 time steps (the improved scheduling method was not implemented yet). In each search a total of 16 experiments were performed. The convergence history of the simulation is shown in Figure 6-6. At this stage a detailed analysis is not necessary and therefore only main observations are provided here:

- The path to the solution found by the ICS was significantly better than in the non-controlled simulation. The reduction in number of sweeps was 55.6%. However, the figure only represents the sweeps used during the real simulation and not the sweeps performed during the search. The total number of sweeps executed (and consequently the execution time) in the ICS-controlled simulation was substantially larger than in the standard simulation. Therefore, the 55.6% reduction represents only the theoretical improvement – the reduction that would be achieved if the agent could always apply the appropriate set of control parameters without having to search for them. Nonetheless, the results are very encouraging as they show that proper control actions can provide significant performance gains.

- The system was able to assure full convergence – all the time steps reached the required accuracy while in the non-controlled simulation one of the time steps did not converge. This fact is very important since it is believed that the full convergence of all time steps guarantees the most accurate final results (within the particular model and tolerance level used).
- Divergent time steps were detected and dealt with appropriately. The system was always able to detect divergence and to recover properly. However, the adjustments used for divergence recovery were usually quite conservative and significantly impaired the performance.

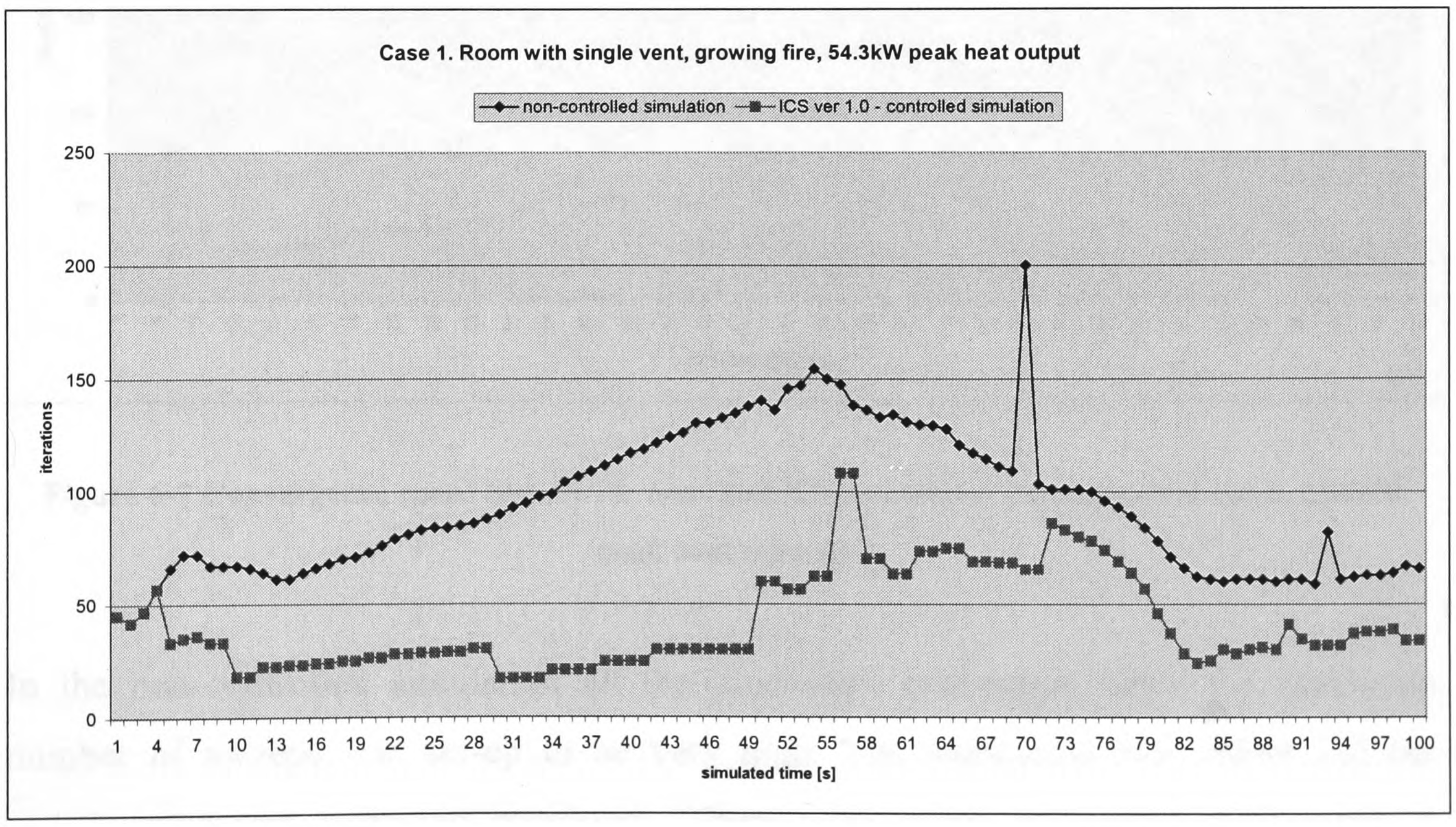


Figure 6-6 Convergence speed history for non- and ICS-controlled simulations (Case 1, 54kW peak heat output)

6.4.2 250kW case with fine mesh

The second case used the same room size (2.8m x 2.8m x 2.18m) with the same geometry (a fire in the centre of the room, directly on the floor, single vertical vent in the room – door). However, this time the fire was defined by a fast growing heat release curve with the peak heat output of 250kW (at 73s). The mesh was also improved and contained approximately 20,000 cells. Such a mesh provides more accurate results but requires a substantially longer processing time. At the starting point both simulations used the default set of relaxation parameters and the simulated period was 100s. The results from both non- and ICS-controlled runs are shown in Figure 6-7.

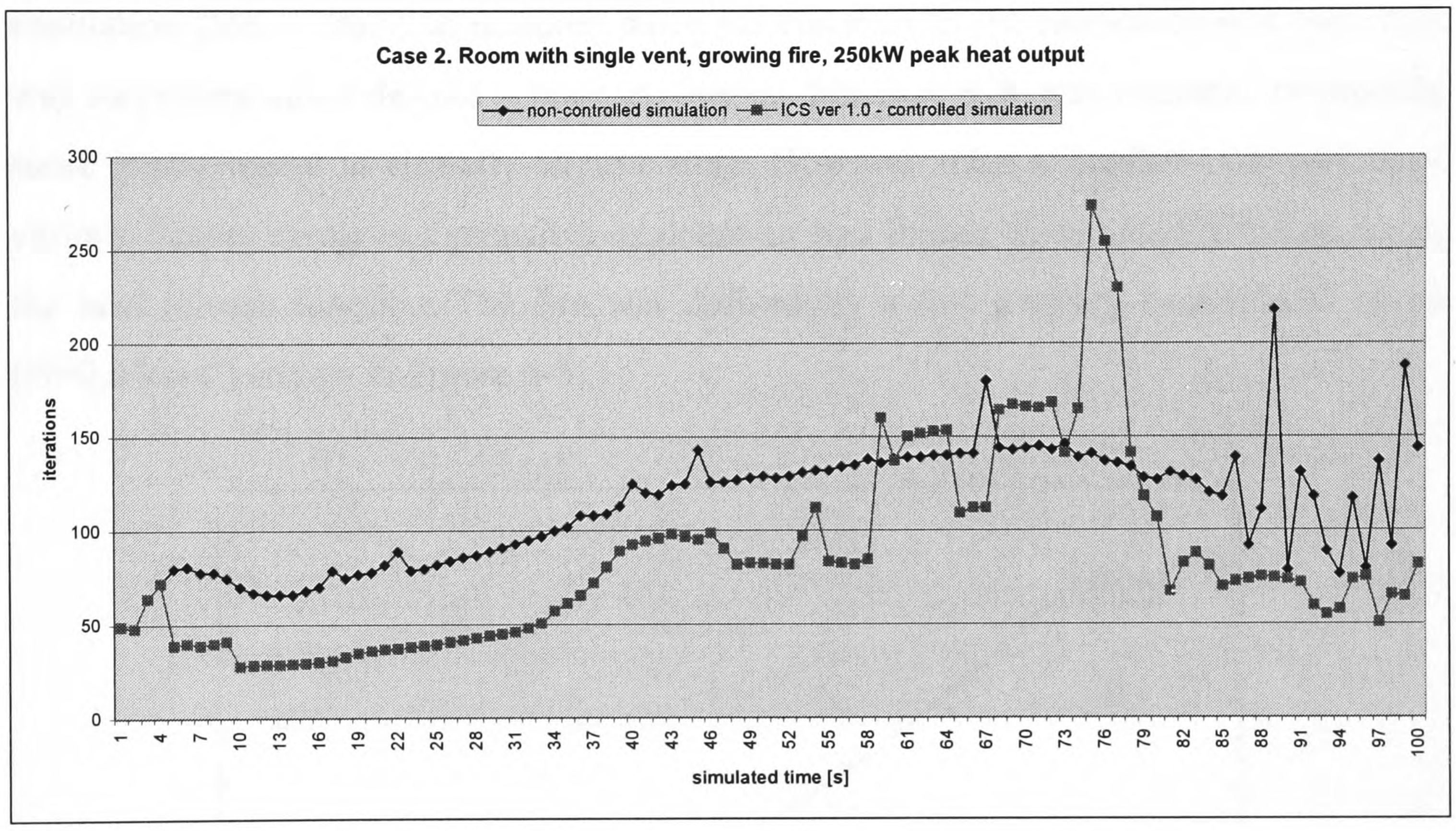


Figure 6-7 Convergence speed history for non- and ICS-controlled simulations (Case 2, 250kW peak heat output)

In the non-controlled simulation all the time steps converged, since the maximum number of sweeps was set-up to be very high. The simulation was stable and the convergence rate remained consistent. There were some substantial differences in convergence speed during the last 15s of the simulation but none of them prevented the full convergence. During the first 75 steps the number of sweeps per time step was steadily increasing due to the gradually accelerating heat output. Then the number of sweeps began to decrease but this was quickly obscured by significant fluctuations in convergence speed. The cause of these variations was unknown.

In the ICS-controlled simulation the search was scheduled every 5 steps (the improved scheduling method was not yet implemented). Again, all the time steps converged and every diverging time step was properly detected and corrected. The overall theoretical reduction in the number of sweeps was 25%. This improvement was small compared to the previous test but this was probably due to the poor evaluation function and the fact that a 250kW fire case was more difficult to control than a 50kW case.

This case proved to be interesting, as there was a period in the ICS-controlled simulation (59s – 78s) that required more sweeps than in the non-controlled one. This was surprising since despite a poor evaluation function ICS was expected to provide some improvement in virtually all time steps. However, after a detailed examination of various factors involved a plausible explanation was found. Once again, the clue lay in the heat release function. The fire was defined by a fast growing heat release curve ($H=0.0469 \cdot t^2$) shown in Figure 6-8.

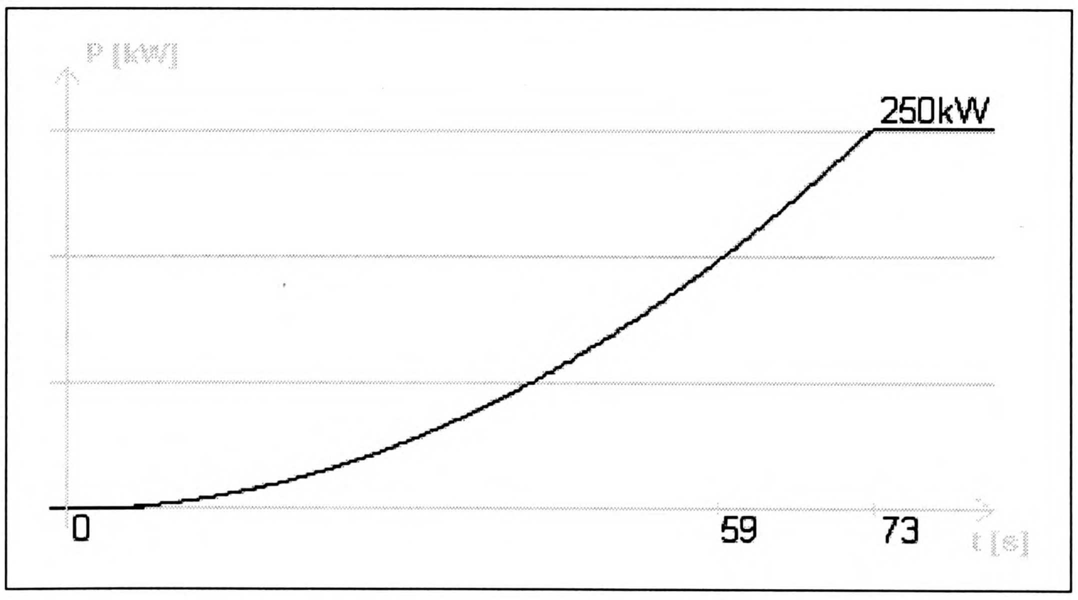


Figure 6-8 Heat release curve for a 250kW fire

By comparing the convergence history graph and the heat release curve one can see that the biggest performance deterioration occurred where the heat release was growing at the fastest rate (59s-73s). At that point the control parameters used by ICS in the first part of the simulation were no longer appropriate and caused divergence. This resulted in the divergence recovery procedure, which seriously affected performance. Some of the divergence points can be clearly identified from Figure 6-7 as the points where the number of sweeps to convergence increases steeply (58s, 68s and 74s). The convergence history also suggests that the performance-oriented adjustments might have been too aggressive and therefore increased the danger of divergence. All these problems were attributed to the poor evaluation function and it was hoped that better heuristics should be able to overcome similar problems in the future.

The following list present all the factors that were believed to contribute and compound to the performance problems:

- **Inappropriate control actions.** Partly due to the poor evaluation function, the control actions were often either too aggressive or too conservative. The divergence recovery caused substantial decrease in speed while some performance-oriented control actions resulted in frequent divergence. Analysis of all the experiments suggested that at least some of those problems could have been avoided through the use of a better evaluation function.
- **Incorrectly scheduled control actions.** In the first part of the simulation when the heat output was increasing slowly the heuristic search produced a very relaxed set of control parameters, aimed at maximising time performance. However, as the heat output started to sharply increase, those control parameters quickly caused divergence. The subsequent divergence recovery resulted in lower simulation speed, which was not improved until the next set of experiments.
- **ICS had no knowledge about the processes happening inside the simulated domain.** The search could be executed much more efficiently and effectively if the agent was aware of the physical conditions developing in the domain. The heat release rate provided the most useful information, and could be used to improve the experiment scheduling.
- **250kW fire in Steckler-sized geometry is hard to control.** The 250kW fire in a small room is difficult to control, which further compounded the problems encountered during automatic control.

Solutions to some of these problems have been implemented in the next version of ICS and are described in Chapter 7.

6.4.3 2.1MW case with fine mesh

The third case consisted of a big room (6m x 4m x 3.3m) with a fire on the floor in the centre of the room. The fire was defined by a growing heat release curve with the peak heat output of 2.1MW (after 450s). The room had a single vertical vent (door) measuring 1m x 1.8m. The mesh contained approximately 20000 cells. The simulation used the default set of relaxation parameters and ran for a period of 550s. This case was significantly different from the previous ones as it contained a very big fire (2.1MW) and used a radiation model. It was chosen to help establish whether the control architecture was flexible and generic enough to control different cases and physical models.

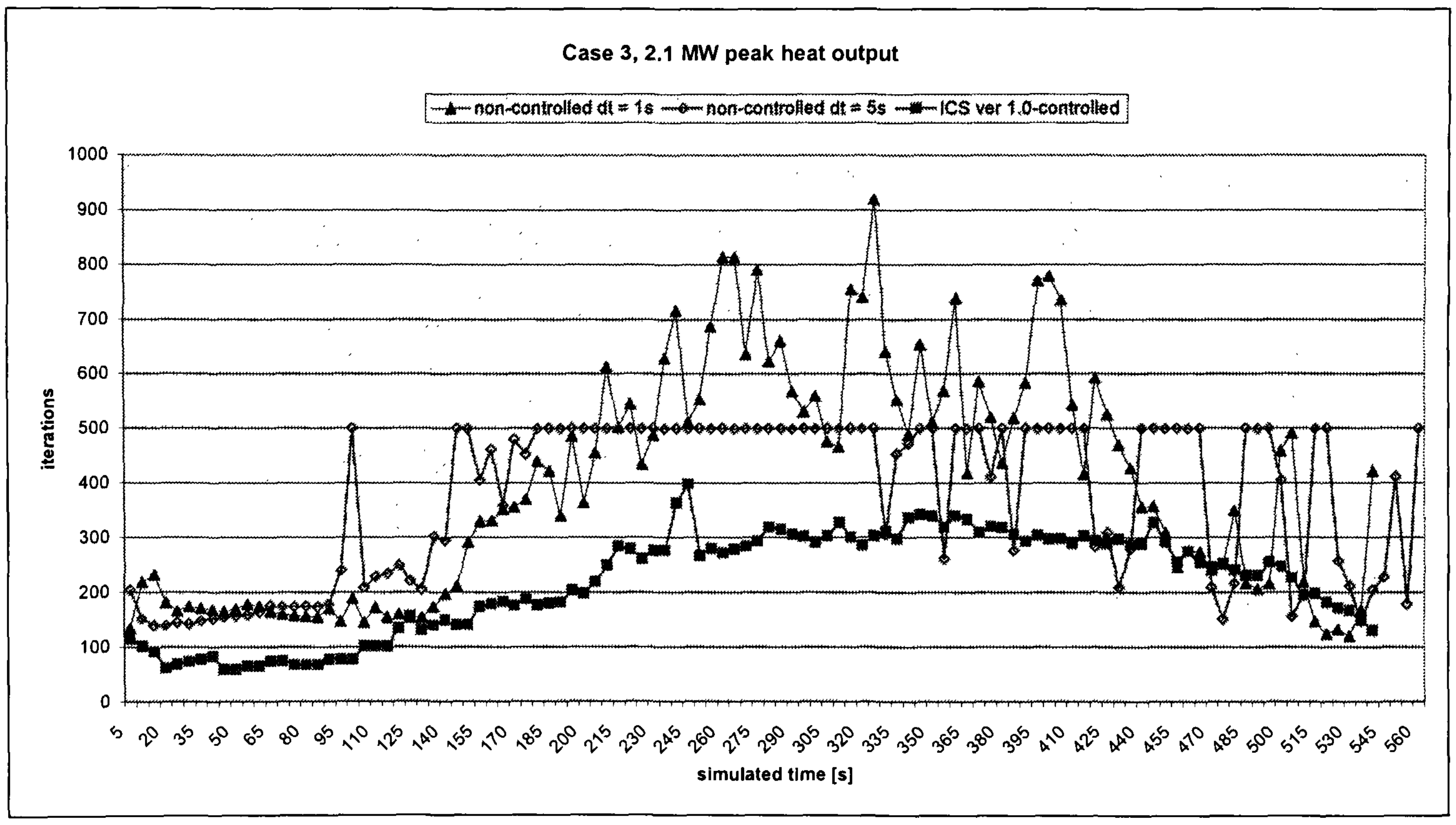


Figure 6-9 Convergence speed history for non- and ICS-controlled simulations (Case 3, 2.1MW peak heat output)

The first non-controlled simulation ran with 5s time step size but did not provide the necessary information since, despite setting the maximum number of sweeps at 500, over 50% of time steps did not converge. Consequently, any comparison with the results obtained from an ICS-controlled simulation would not be very reliable. The next non-controlled simulation used 1s time steps and produced acceptable results. Although several time steps still failed to converge, the number of non-converged steps was much lower (47 from the total of 550). This was deemed acceptable and for the purpose of speed comparison the non-converged time steps were treated as if they converged with the maximum number of sweeps allowed in the initial set-up (200). The results from both non-controlled simulations and the ICS-controlled simulation are presented in Figure 6-9.

This case is a perfect example of the effectiveness of the ICS. The first and the most important observation was that all time steps from the ICS-controlled simulation fully converged. Despite two runs, full convergence proved to be impossible to achieve in the non-controlled simulation. Furthermore, the ICS obtained considerable reduction in the number of sweeps. It needed only 24,248 sweeps for the whole simulation compared to 43,844 in non-controlled simulation, which represents 44.7% theoretical improvement. The convergence speed obtained by the ICS was superior for virtually all time steps. Moreover, even when the cost of the experiments was taken into account, the execution time of the ICS-controlled simulation was comparable to the non-controlled one (although still greater). This came as a surprise, since it was generally expected that in the early version of the control system the search cost would always be excessive. Consequently, the results from this case clearly indicate that, providing that the search cost is further reduced, an improved version of the ICS could be routinely used for simulation control by both experts and novice users.

6.5 Fault recovery

The ICS proved to be perfectly capable of detecting faults in the solution process and restoring convergence. The convergence problems were always properly detected and the search for a better set of control parameters initiated. The heuristic search led to divergence recovery but due to the poor evaluation function, the recovery actions often seriously impaired the performance. The analysis of the control decisions made by the ICS confirmed the experts' belief that convergence was best restored by removing relaxation and/or reducing the time step size. In a majority of cases, the actions chosen by the ICS for divergence recovery were similar to the actions proposed by experts. However, the control system did not base its decisions on any arbitrary knowledge about the results of particular control actions but on the real-time assessment of the experiments performed. Figures 6-10 – 6-12 show examples from ICS-controlled simulations, which illustrate various types of control actions performed by the ICS to restore convergence.

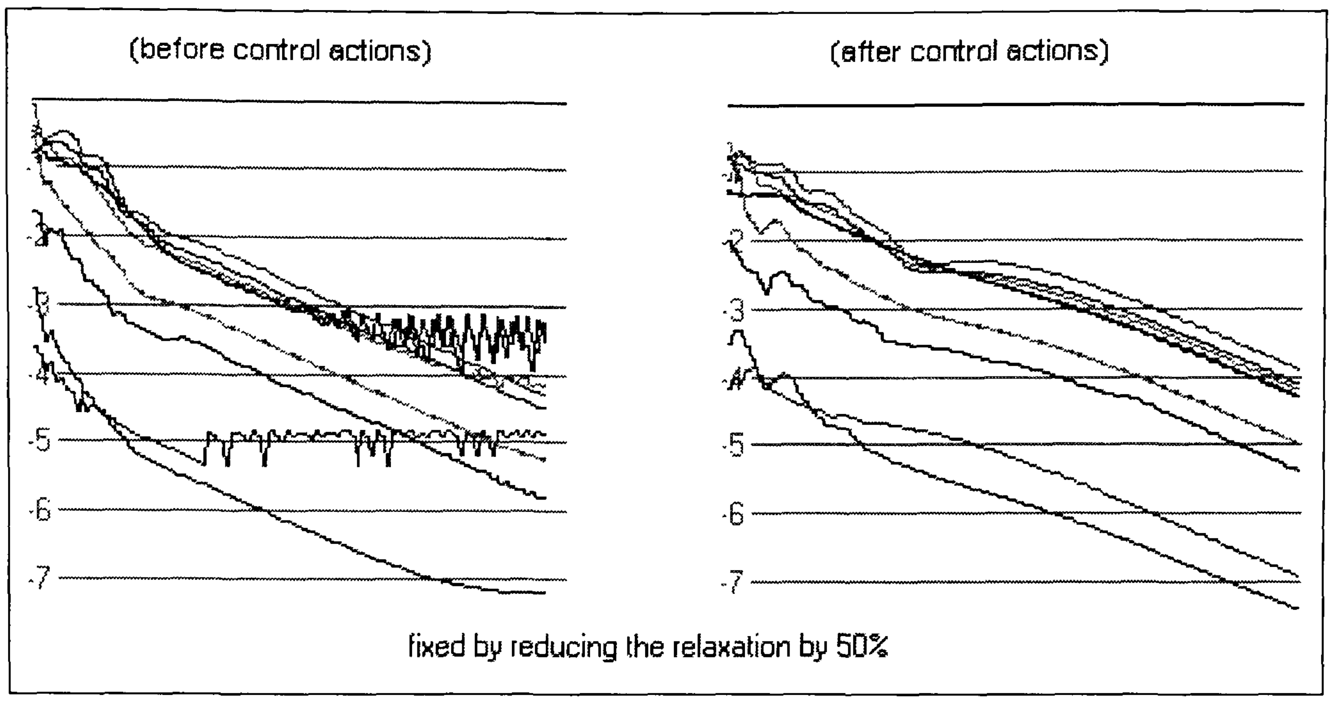


Figure 6-10 Divergence recovery by relaxation reduction

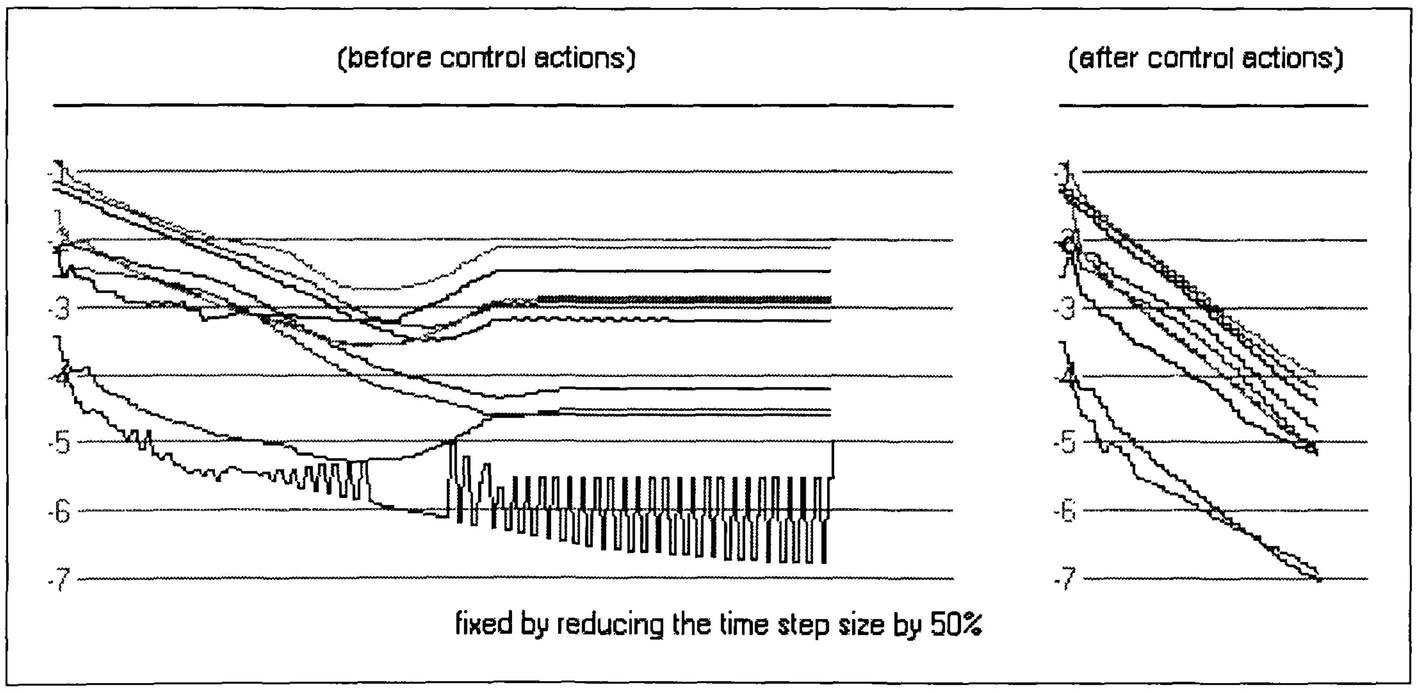


Figure 6-11 Divergence recovery by time step size reduction

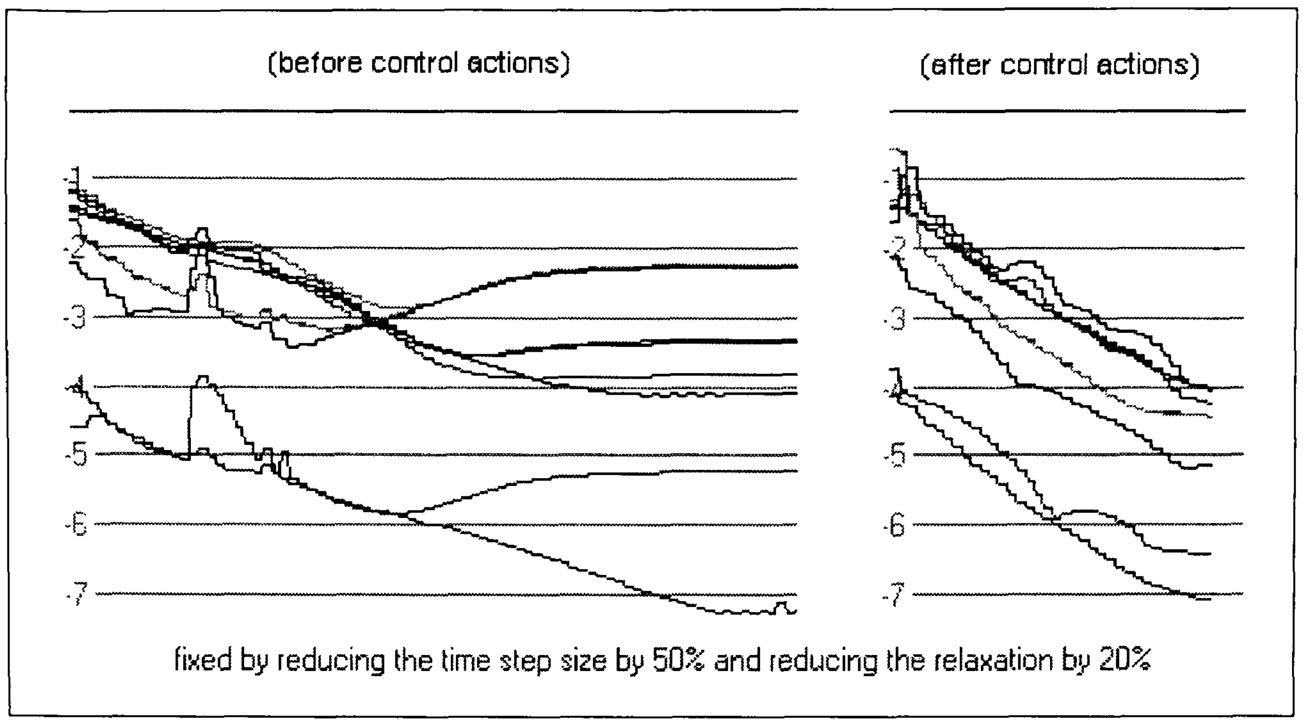


Figure 6-12 Divergence recovery by relaxation and time step size reduction

6.6 Full convergence assurance

The full convergence of all time steps can only be guaranteed if it is possible to perform robust divergence recovery. Convergence assurance is an important issue, as it guarantees that the final results are correct. Although a simulation with several not fully converged time steps can still produce acceptable results, only by ensuring that every time step converges one can be confident that the final results are the most accurate that can possibly be obtained with a particular model (within the tolerance specified). Such results can be safely used for comparison with real life data and for model validation. In a simulation where some time steps did not converge it is difficult to determine whether the differences between empirical data and the numerical solution are caused by an incorrect model or by convergence problems during the simulation.

6.7 Initial conclusions

The analysis of the results substantially enhanced our understanding of the control techniques. The experience and the data obtained during the tests helped build an improved version of ICS. The full description of the new system is presented in Chapter 7 and therefore only the most important conclusions that emerged as a result of the first ICS-controlled runs will be outlined here.

It was established that the system worked correctly and effectively. Despite a poor evaluation function, very significant improvements were observed. In all test runs the control system was always able to recover from divergence. This is a very important feature since serious divergence in non-controlled runs often requires the whole simulation to be restarted. Automating the divergence recovery process was the first step towards an autonomously controlled simulation. Thanks to the effective convergence assurance algorithm, the ICS could guarantee the accuracy of the final results.

The ICS demonstrated that savings in execution time could potentially be very substantial. Even with the poor evaluation function the theoretical reduction in simulation time reached 45%. However, there is a long way from theoretical to the real performance improvement. Currently, the shortest path to the solution is only found as a result of extensive and frequent search. In practice, for most cases the cost of this search was substantially greater then the reduction in execution time and therefore the duration of the ICS-controlled simulations was longer than the corresponding non-controlled ones. Consequently, further research should focus on reducing the cost of the search and improving the heuristics used.

6.8 Summary

This chapter presents the initial tests of the first version of the heuristic control system. Several minor improvements that were implemented during the tests are briefly explained. The results of three different scenarios are presented together with the benefits obtained and problems encountered. A full analysis of ICS-controlled cases is performed and, as a result, a number of solutions to the observed deficiencies are proposed. A poor evaluation function and excessive search cost are identified as the most significant factors that affect ICS performance. The following list contains a brief summary of the test results:

- ICS was always able to recover from divergence and other convergence-related problems.

- Problems were properly detected and recovery actions triggered.
- Convergence of every time step was assured.
- The system demonstrated that a substantial reduction in number of sweeps is possible by performing correct control actions.
- The system did not reduce the execution time because of the excessive cost associated with the heuristic search.

The next generation of the control system (ICS ver. 2.0) is presented in Chapter 7.

Chapter 7

Second version of the control system

7.1 Introduction

In the previous chapter we described the development process and the initial tests of the first version of the ICS. The test results were thoroughly analysed in order to identify weak points in the design and to assist with further improvements. The ICS 1.0 was not an efficient control system but it was extremely useful as a test vehicle for the new control architecture and helped identify and correct many initial design problems. However, it was also apparent that further improvements were necessary as the evaluation function was poor and the search cost excessive.

In this chapter we show how the system evolved in response to the deficiencies of the first prototype. Several major enhancements are presented:

- Creation of dedicated search plans for specific simulation states.
- Introduction of goal-driven search plans.
- Improvements to the compound evaluation function.
- Dynamic pruning of the search space.

The modifications were fully implemented in the next version of the control system (ICS 2.0) and are expected to significantly reduce the cost of the heuristic search.

7.2 Identification of different states during simulations

The results obtained during the tests clearly indicated that only a small subset of experiments could provide improvements in specific situations. For instance, if the time step diverges then there is no need to try aggressive control actions, aimed primarily at performance improvement, as the goal is to restore convergence. It was observed that only some types of the control actions could actually restore convergence. This observation led to the first important change in the ICS design.

It was clear that different types of graph required different modifications to the control parameters. The four main types of graphs were identified and each group had to be assigned an appropriate set of control actions:

- Diverging,
- Slowly converging,
- Containing excessive oscillations,
- Normal.

All the available residual graphs produced during heuristic search were analysed. This analysis produced very interesting conclusions. The main observation was that for each type of the residual graph there seemed to be a specific set of adjustments that were likely to provide the required solution. The conclusions for each type of a residual graph are outlined below.

7.2.1 Diverging graphs

A dedicated set of adjustments is required for diverging graphs (DIVERGENCE_RECOVERY). From the comprehensive set of control actions, only a small subset proved to be able to restore convergence:

- Reducing the time step size.
- Removing some relaxation.
- Removing some relaxation and reducing the time step size.

Consequently, only 5 types of control actions from the initial 20 were suitable for divergence recovery:

1. Reducing the time step size by 50%.
2. Removing 20% of the relaxation.
3. Reducing the time step size by 50% and removing 20% of the relaxation.
4. Removing 50% of the relaxation.
5. Reducing the time step size by 50% and removing 50% of the relaxation.

7.2.2 Normal graphs

The next group of graphs was the NORMAL_GRAPH type. If the graph converged quickly and the simulation was running smoothly then the main purpose of the adjustments should be performance improvement – there is no need to investigate any calming measures. Therefore, the following control actions were identified as having a potential speed-up effect:

- Adding more relaxation.
- Increasing the time step size.
- Reducing the time step size (!).
- Increasing the time step size and increasing the relaxation.
- Reducing the time step size and increasing the relaxation.

It is interesting (and rather surprising) that in some cases, the reduction of the time step size can result in the speed improvement. It goes against the intuition of many experts but the results from the completed simulations clearly indicated that reducing the time step size could in some cases significantly improve the convergence speed. These claims were backed by solid empirical data and the experts subsequently agreed with the conclusions and acknowledged that since a smaller time step implies a more stable simulation then it may also mean faster convergence.

There were 11 different experiments identified, which were able to provide performance improvements. This was a smaller reduction than in the case of DIVERGENCE_RECOVERY but it still amounted to almost 50% fewer experiments than in the ICS 1.0.

7.2.3 Oscillations and slow convergence

In case of excessive oscillations and slow convergence the situation was less clearly defined. It was difficult to establish which changes were most likely to remove the oscillations. Things were even more complex for cases involving slowly diverging graphs. While the oscillations could be easily detected using the algorithm described in

Chapter 5, the standard method for convergence prediction was not very effective in detecting slow convergence. However, even with a very accurate prediction function the assessment would have been difficult, as the reasoning process would have to analyse historical information from the preceding time steps. It was therefore decided that in the next version of ICS there would be no specific control actions for slowly converging graphs. If the residuals manage to converge within the allocated number of sweeps than the graph is always treated as fully converged. If, on the other hand, the graph does not converge than it is classified as diverging and the relevant recovery procedure is initiated. This might not be the best solution available but it was understood that there would be no substantial adverse effects on performance. This issue should be a subject of further research, especially as even the experts were often not able to agree whether a particular graph was normal or converging too slowly.

In the case of a graph with excessive oscillations, a set of experiments was eventually selected. The analysis suggested that a variety of changes were effective in removing the oscillations. Very often several different adjustments were successful in suppressing the oscillation in the same time step. As a result, there seemed to be little point in performing a comprehensive search (due to the cost) and therefore a small number of diverse adjustments were selected as most effective in oscillation removal (6 different types of adjustments were used in all the cases presented in this dissertation).

7.3 Major heuristic function improvements

As a result of the tests with ICS ver 1.0, there was now sufficient data to perform a thorough search for better heuristics. Firstly, it was decided that only properly converging graphs would be comprehensively assessed. There is no need to assess diverging graphs since the corresponding time steps do not produce correct results and there is little point in determining which one is "more diverging". Therefore such graphs are immediately discarded and not used in the more detailed assessment. Both the diverging graphs and the ones with excessive oscillations are considered faulty and are not assessed. As a result the evaluation function operates only on converging graphs with limited amount of oscillations.

The next major modification to the assessment algorithm involved eliminating oscillations from the compound assessment factor. Further consultations with the experts led to the conclusion that moderate oscillations were inherent part of the simulation process and did not significantly affect the convergence speed or result accuracy. The experts still maintained that excessive oscillations were dangerous and should be avoided but conceded that minor oscillations were virtually unavoidable and did not affect the solution stability. The experts presented an example, where minor fluctuations in the neutral plane height resulted in residual oscillations, which were acceptable and expected. These oscillations did not affect the accuracy and were caused by the mesh configuration combined with physical factors. Consequently, after excluding the oscillations from the compound assessment function, only two features remained: **Convergence Speed** and **Irregularities**.

The assessment function must be able to compare the experiment results and choose the best one (i.e. the one that offers the optimum balance of speed and stability). Consequently, a function that compares two different graphs and produces a number that reflects the relative difference between them should serve as adequate heuristics. Such an evaluation function would analyse two graphs (A and B) and produce a positive number if A was better than B or a negative number if A was worse then B. The absolute value of the assessment result should reflect the relative difference between the graphs and should be consistent regardless of the order in which the graphs were presented to the function. Therefore if graph A were 30% better than graph B then the graph B would also be 30% worse than graph A (although it is not how the percentage differences normally work). These considerations led to the development of an evaluation function that was believed to perform sufficiently accurate graph assessment.

The proposed algorithm introduces a new numerical value, which represents an *improvement in convergence speed*. The improvement is calculated in a special way to ensure the commutative property described above. A very simple way to compare the speed would be to use the relative change as the measure. However, this method is inconvenient since it is not commutative. For example, if graph A used 25 sweeps/s and graph B used 100 sweeps/s then graph A was 75% quicker than graph B, while graph B was 300% slower than graph A. This is confusing and therefore a more convenient comparison algorithm was devised. The new method uses a normalised 'improvement in speed' rather than raw difference in speed. The concept of 'improvement in speed' is based on an intuition shared by the experts that 50% reduction in number of sweeps constitutes 100% improvement while 75% reduction makes 300% improvement. However, when it comes to deterioration, the scale remains linear, i.e. 100% increase in number of sweeps produces 100% deterioration while 300% increase amounts to 300% deterioration (or correspondingly, -100% and -300% improvement). The complete speed improvement function is shown in Figure 7-1.

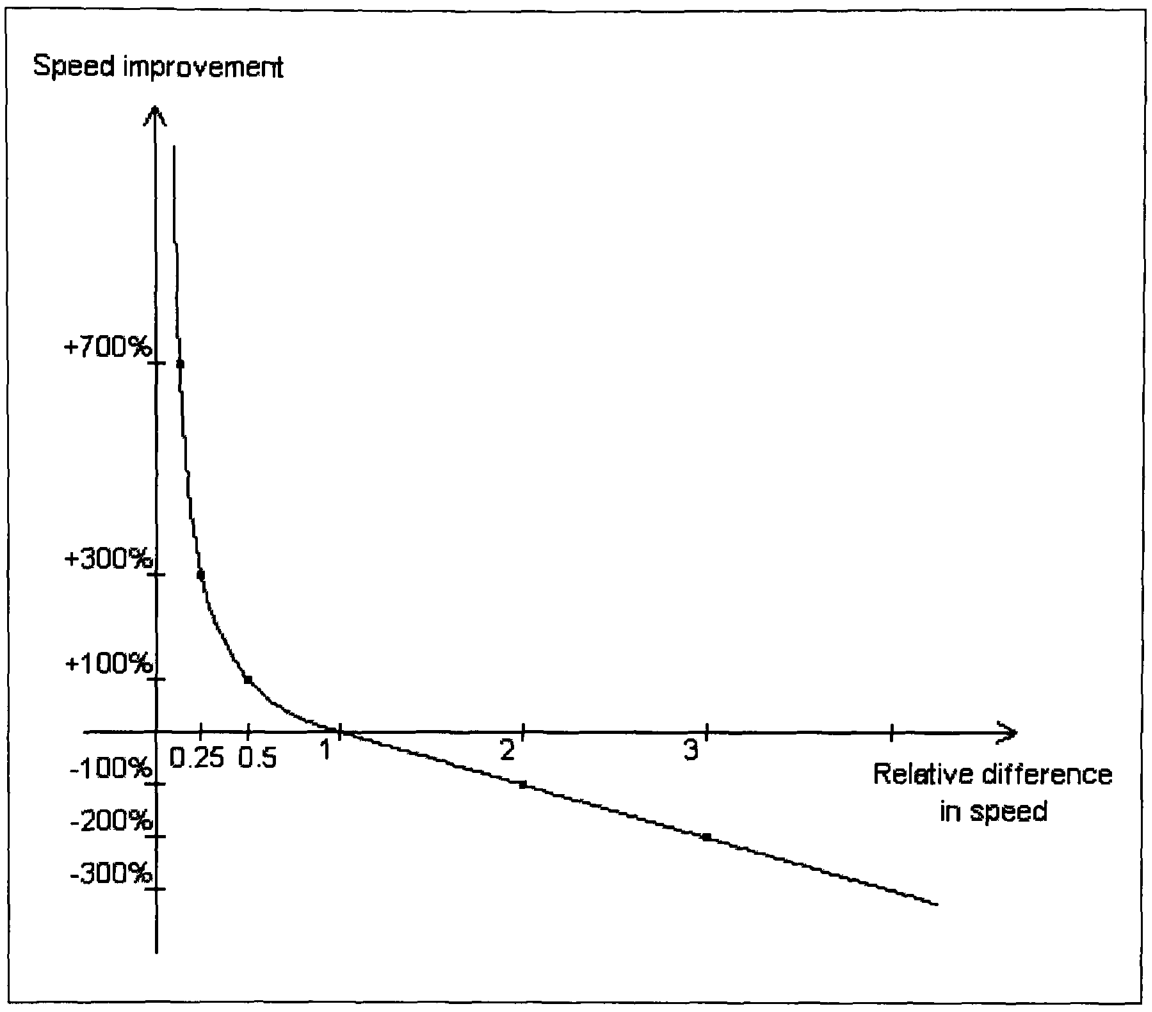


Figure 7-1 The speed improvement function

This function provides an intuitive assessment of the difference in speed between two graphs and therefore is much easier to work with than raw speed differences. It proved to be an important building block for the compound evaluation function.

Some modifications were also introduced to the irregularities assessment. After analysing a number of graphs, a "safe" level of irregularities was determined. This "safe" level is defined as an amount of irregularities below which they should be considered irrelevant. Consequently, if both graphs contain only small amount of irregularities then they are considered irrelevant and the graph assessment is only based on convergence speed. This prevents scenarios (common in the ICS ver 1.0) where big relative differences between otherwise small irregularities had substantial impact on the final evaluation result. Further provisions were also made to ensure that the algorithm for comparing irregularities had the same commutative properties as convergence speed assessment.

Consequently, the resulting *relative evaluation function* took the following form:

$$E = S + \delta \cdot I \qquad (7.1)$$

Where:

$E =$	*evaluation result describing relative difference in quality between two graphs*
$S =$	*speed improvment*
$I =$	*irregularities assessment*
$\delta =$	*weight of irregularities assessment*

It was believed that the influence of the irregularities on the overall evaluation result should be small as the time steps, which diverged or contained excessive oscillation were already excluded from the assessment process. Consequently, the weight of irregularities assessment (δ) was initially set to 0.2. After a number of experiments, this value was reduced even further to 0.1.

This relative evaluation function was used in the next version of the control system (ICS ver 2.0). The compound assessment consisted of the following steps

- Reject all experiments that diverged or contained excessive oscillation.
- Compare each experiment with a single residual graph (a graph produced by a time step with no modification to control parameters):
    - Determine relative improvement in convergence speed.
    - Determine relative improvement in the amount of irregularities.
    - Calculate the overall improvement factor.
- Chose the experiment that provided the best improvement (i.e. had the best improvement factor).

7.4 Search tree pruning

The cost of the search is one of the most important factors determining the execution time and substantial research effort was directed towards reducing the number of experiments performed. As a result, two enhancements to the original search algorithm were developed and provided a substantial reduction in the number of experiments required.

7.4.1 Goal-driven search plans

The identification of various states within a simulation led to the development of goal-driven search plans for certain problems commonly occurring in simulations. The research was primarily focused at OSCILLATION_REMOVAL and DIVERGENCE_RECOVERY procedures. In the previous system the assessment was always performed after all experiments were completed. Consequently, even if the first set of adjustments obtained the required effect, the remaining experiments would still be performed. This approach was chosen because there was little available information about the effects of control actions and therefore all modifications had to be tried in order to choose the one that provided the best performance. However, after a careful examination of the data produced by the first control system new knowledge emerged, which made it possible to optimise the search process.

Firstly, various types of control actions can now be ordered according to the speed improvement they normally provide. For example, reducing the relaxation by 20% will almost certainly result in a smaller performance deterioration than 50% relaxation reduction (assuming that both of the experiments converge). Therefore, if reducing the relaxation by 20% provides the desired effect (e.g. divergence recovery) then it is absolutely unnecessary to try to apply a 50% reduction. Consequently, by careful ordering of the experiments and early result assessment, the cost of the heuristic search can be drastically reduced in cases where there is a clear goal for the control actions (i.e. recovery from a solution fault). The following example should make the whole process completely clear.

There are 3 different types of control actions that are known to be effective in divergence recovery. In order to minimise the adverse effect on performance the experiments that have the smallest impact on the convergence speed are performed first. If a satisfactory recovery is obtained then the remaining experiments are not executed, if there is no recovery – further experiments are conducted. At the beginning only two experiments are performed, one reducing the time step size (by 50%) and the other removing some of the relaxation (20%). These two experiments are always performed (even if the first one provided divergence recovery) since they use completely different control methods and may have different effects on performance. If both of them produce satisfactory results then the one that provided better performance is selected. If neither of them was able to recover then the next two experiments are performed. If those also fail, then one more adjustment is tried. Different methods (user interaction or perhaps a substantial reduction of the time step size) can be used if none of the experiments provides the necessary recovery. The complete search plan is illustrated in Figure 7-2.

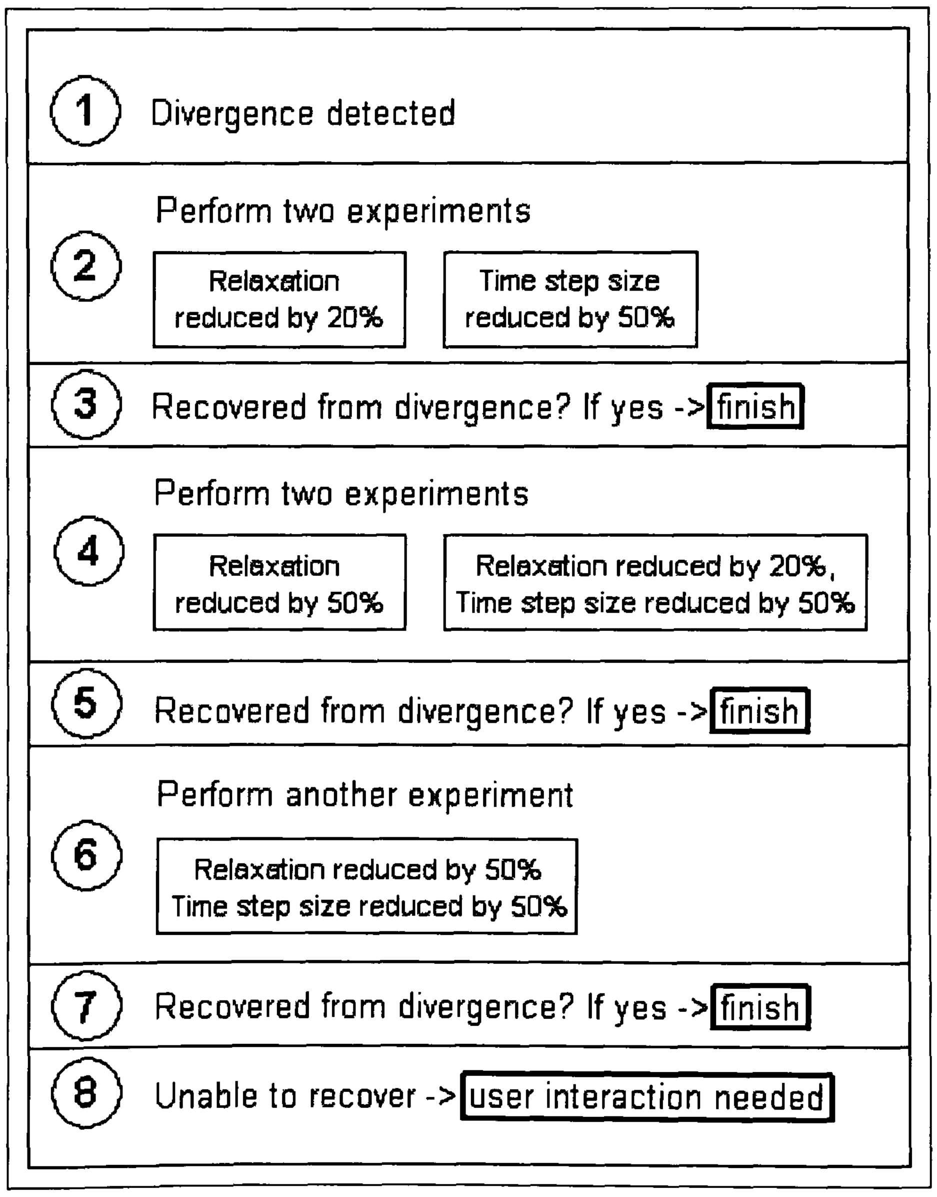

1 Divergence detected
2 Perform two experiments
Relaxation reduced by 20%
Time step size reduced by 50%
3 Recovered from divergence? If yes -> finish
4 Perform two experiments
Relaxation reduced by 50%
Relaxation reduced by 20%, Time step size reduced by 50%
5 Recovered from divergence? If yes -> finish
6 Perform another experiment
Relaxation reduced by 50% Time step size reduced by 50%
7 Recovered from divergence? If yes -> finish
8 Unable to recover -> user interaction needed

Figure 7-2 Divergence recovery algorithm

This technique was made possible because of the extensive analysis of all the data available from the earlier tests, which determined the average impact on the performance for each type of adjustment.

A similar technique is used for oscillation recovery but the oscillation removal plan is less reliable since there is no clear-cut set of control actions known to be effective in this particular situation. Consequently, the adjustments tested are more diverse which makes the ordering according to the performance impact difficult. Nevertheless, a significant time saving is also obtained for oscillation recovery.

7.4.2 Dynamic modification of the search plan

Another method for reducing the search cost involves the modification of the search plan as it progresses. One of the reasons why experts perform a very limited search is that they are usually able to eliminate most types of control actions using the information from previously completed experiments. For example, if an expert increases the time step size by 50% trying to obtain speed improvement and the experiment diverges, then he knows immediately that increasing the time steps size by 100% will also cause divergence and therefore there is no need to try this. Experts achieve impressive results using these simple rules and therefore it is very desirable to incorporate this type of reasoning into the control system. However, the interviews with the experts were only able to provide a limited number of rules and therefore a statistical analysis of all the convergence data obtained during the earlier tests was performed.

The experts confirmed that there were two main types of relationship between experiments:

- If experiment A led to divergence then experiment B would also cause divergence.
- If experiment A provided improvements in convergence speed then experiment B would not provide a better improvement.

Of course, there might be other rules but only these two could be easily identified and applied to improve the search tree pruning.

The method of discovering such rules for specific experiments is similar for both types of relationship and therefore only the process of obtaining divergence rules is described here. The data from over 300 experimental time steps performed in several different simulations was analysed in the search for consistent patterns. For each type of adjustment there are 18 potential rules that might determine whether this experiment would diverge (since there were 19 types of control actions tested). For example, for the experiment no 1 (substantial relaxation increase) there are 18 other experiments whose result might predict the outcome of that experiment. This gives a total of 342 hypothetical rules that have to be separately assessed and verified. Fortunately, in practice a smaller number of rules was assessed, since it was already known that some of the experiments were pointless (simultaneous relaxation reduction and time step size increase). Furthermore, the rules were only to be used in particular search plans (SPEED_UP or DIVERGENCE_RECOVERY), and therefore it was not necessary to find rules involving experiments from two different search plans. Consequently, only around 100 hypothetical rules were assessed and 29 were found to be correct. The results of the rule validation procedure for experiment no 2 (20% relaxation increase, no time step size change) are shown in Table 7-1.

Rule assessed	Rule confirmed [%] ([cases])	Rule refuted [%] ([cases])	Final assessment
Exp 1 div => exp 2 div	60.5% (75)	39.5% (49)	Refuted
Exp 5 div => exp 2 div	52.7% (69)	47.3% (62)	Refuted
Exp 6 div => exp 2 div	62.8% (59)	37.2% (35)	Refuted
Exp 8 div => exp 2 div	42.5% (77)	57.5% (104)	Refuted
Exp 9 div => exp 2 div	52.7% (79)	47.3% (71)	Refuted
Exp 12 div => exp 2 div	46.4% (77)	53.6% (89)	Refuted
Exp 13 div => exp 2 div	62.9% (73)	37.1% (43)	Refuted
Exp 16 div => exp 2 div	83.9% (52)	16.1% (10)	**Possible**
Exp 17 div => exp 2 div	97.0% (32)	3.0% (1)	**Confirmed**

Table 7-1 Rule validation results for experiment no 2

The table shows that there were 9 potential rules but most of them proved to be not valid. There were three different assessment results possible:

- Refuted – the rule was found to be incorrect.
- Confirmed – the rule was found to be true.
- Possible – the rule might be true but the result were inconclusive.

The data for the analysis was produced automatically and therefore there was a small percentage of diverging cases that were wrongly assessed as converging (and vice versa) due to the limitation of the automatic divergence detection algorithm. Consequently, in the rule validation, if the rule was confirmed in more than 90% of cases then it was assessed as valid. If the rule was only confirmed in less than 80% of cases then the rule was classified as not valid. The in-between range (80-90%) was classified as 'possible' and all the cases that contravened the rule were inspected visually in order to make the final decision (confirmed or refuted). The 'possible' assessment for one of the rules in Table 7-1 was changed to 'confirmed' after visual inspection. Consequently, the following rules emerged for predicting whether the experiment no 2 will diverge:

- If the experiment no 16 (relaxation increased by 50%, time step size reduced by 50%) diverged then the experiment no 2 (relaxation increased by 20%, no time step size change) would also diverge.
- If the experiment no 17 (relaxation increased by 20%, time step size reduced by 50%) diverged then experiment no 2 (relaxation increased by 20%, no time step size change) would also diverge.

The rules of the second type (which help to eliminate the experiments unlikely to provide better improvement than already obtained) were discovered using a similar method. Although only 8 new rules were found, the associated reduction in the search cost was still significant.

All the rules are enumerated in Appendix A but there were also some general conclusions that emerged from this analysis:

- If an experiment diverges then any experiments with the same relaxation parameters but a larger time step size will also diverge.
- If an experiment diverges then any experiments with the same time step size but increased relaxation parameters will also diverge.
- If an experiment diverges then any experiments with a larger time step size and increased relaxation parameters will also diverge.

- If increasing the relaxation provides speed improvement then smaller increase will not provide better improvement.

These conclusions are in perfect agreement with the experts' intuition, which provided additional confirmation.

The rules were implemented in the control module that performed the heuristic search and proved to be very effective in reducing the search cost by eliminating the unnecessary experiments during the search. The following example shows a small section of the output from the heuristic system generated during a real simulation:

```
(1)
      Start experiments request received
            Choosing the strategy:
            SPEED_UP  strategy selected
      Experiments for the time step no. 17 initialised
(2)
            Experiment no. 25 initialised
            The modifiers are: 1, 1, 1
            Experiment no. 25 has completed
            Assessment results:
                  Convergence: 230.112
                  Irregularities: 0.00240728
                  Oscillations: 0.0732877
(3)
            Experiment no. 16 initialised
            The modifiers are: 1.5, 1.7, 0.5
            Experiment no. 16 has completed
            Assessment results:
                  Convergence: -1
                  Irregularities: 0.0246837
                  Oscillations: 0.25
            Experiment no. 1 skipped as it is likely to diverge
            Experiment no. 8 skipped as it is likely to diverge
            Experiment no. 12 skipped as it is likely to diverge
(4)
            Experiment no. 17 initialised
            The modifiers are: 1.2, 1.2, 0.5
            Experiment no. 17 has completed
            Assessment results:
                  Convergence: -1
                  Irregularities: 0.0116086
                  Oscillations: 0.291096
            Experiment no. 1 likely to diverge but already skipped
            Experiment no. 2 skipped as it is likely to diverge
            Experiment no. 8 likely to diverge but already skipped
            Experiment no. 9 skipped as it is likely to diverge
            Experiment no. 12 likely to diverge but already skipped
            Experiment no. 13 skipped as it is likely to diverge
            Experiment no. 16 skipped as it is likely to diverge
(5)
            Experiment no. 6 initialised
            The modifiers are: 1, 1, 1.5
            Experiment no. 6 has completed
            Assessment results:
                  Convergence: -1
                  Irregularities: 0.0222255
                  Oscillations: 0.365753
            Experiment no. 5 skipped as it is likely to diverge
            Experiment no. 8 likely to diverge but already skipped
            Experiment no. 9 likely to diverge but already skipped
            Experiment no. 12 likely to diverge but already skipped
            Experiment no. 13 likely to diverge but already skipped
(6)
   None of the experiments provided satisfactory improvement
```

At point (1) the plan for the experiments is selected. The previous time step must have converged because SPEED_UP plan is chosen. At (2) the first experiment is performed (this is the no-changes experiment, which is used to assess the improvement provided by subsequent experiments). It converges within 230 sweeps and does not contain excessive oscillations or irregularities. At (3) the next experiment is run but this one diverges. A set of rules is applied and it is predicted that the experiments 1, 8, 12 are also likely to diverge and therefore should not be executed. Next experiments, at (4) and (5), also diverge and further control actions are eliminated by the appropriate rules. At point (6) the plan has been fully executed but since none of the experiments provided any speed improvement, modifications to the control parameters will not be applied. The cost of the search was substantially reduced – the original plan contained 12 experiments but only 4 of them were actually executed. Therefore, this technique plays an important part in reducing the execution time (although the actual improvement will vary).

7.5 Summary

This chapter describes various enhancements incorporated in ICS ver 2.0. The analysis of the results produced by ICS 1.0 helped in the development of new methods for improving the efficiency and effectiveness of the control system. Several different states were identified and then linked to specific control actions that had proved to be effective in those particular situations. Dedicated search plans were created for each state, which helped significantly reduce the number of experiments performed during the search (as comprehensive search was largely eliminated). The compound evaluation function was also significantly improved and led to more accurate heuristic and more efficient search. Statistical analysis gave rise to goal-driven search plans and a technique for dynamic plan modification. These improvements resulted in the further reduction of the number of experiments performed during each search. The new enhancements were fully implemented in the next version of the control system (ICS 2.0) and the final results are presented in the Chapter 8

Chapter 8

Results

8.1 Introduction

In Chapter 7 we proposed several improvements designed to overcome one of the main deficiencies of the ICS 1.0 – high cost of the heuristic search. It was expected that the modifications would significantly improve the performance in the new version of the ICS (2.0). It was also hoped that the new version would provide performance improvements similar to those obtained by the experts.

In this chapter, we compare the simulations that use the ICS 2.0 against non-controlled cases with the same set-up. A detailed analysis of the results is presented with particular emphasis on simulation speed, fault recovery and accuracy. For clarity each of these factors is analysed and assessed separately.

In order to determine the effectiveness of the control techniques across a wide range of cases, three different scenarios with varying heat output curves (peak output varied between 54kW to 2.1MW) were simulated. All cases had simple geometry (compartment size between *2.8m x 2.8m x 2.18m* and *6m x 4m x 3.3m)* and did not use combustion model. Some cases used six-flux radiation model.

8.2 Speed comparison

The same scenarios that were used in the validation of the ICS 1.0 (Chapter 6) were also used for the ICS 2.0 assessment. Consequently, the results obtained by the ICS 1.0 are also presented in this chapter to assess the effect of the improvements implemented in version 2.0. The ICS 1.0 provided full convergence assurance and fault recovery but failed to produce performance improvement due to excessive search cost. The new version was designed to overcome that problem and provide reduction in execution time while still guaranteeing full convergence of all time steps.

The speed comparison is based on the number of iterations performed. The actual execution time was not measured as the simulations were conducted on a multiprocessor machine that sometimes ran two (or even more) cases in parallel and therefore it was decided that the number of iterations was a more reliable performance measure. For the non-controlled simulation its length is simply defined as a total number of iterations required to complete all the time steps. However, for the ICS-controlled cases two speed measures are required:

- **Theoretical simulation length** – number of iterations required to complete all the time steps not including the iterations performed during heuristic search.
- **Real simulation length** – number of all iterations performed during the simulation (both "normal" and search-induced).

This is best explained using an example – assume that an ICS-controlled simulation used a total of 2000 iterations. However, only 1200 iterations were spent on calculating the results while the remaining 800 were used by the heuristic search. Consequently, the theoretical simulation length is 1200 but the real length is 2000 iterations. The theoretical length indicates how long it would take to run the simulation if we knew beforehand what control parameters were appropriate for each time step. However, it is not yet possible to find the correct parameters without performing a search, which requires additional iterations and therefore the duration of the simulation is longer.

For convenience, two separate measures are used for speed assessment: **theoretical speed improvement** and **real speed improvement**. These are calculated by comparing the corresponding length (theoretical or real) of the ICS-controlled simulation against the length of the non-controlled one.

8.2.1 54kW case with coarse mesh

As in the previous tests, the first case was a Steckler-size (2.8m x 2.8m x 2.18m) room with a fire on the floor in the centre of the room. The fire was defined by a growing heat release curve with the peak heat output of 54.3kW (after 50s). The room had a single vertical vent (door). The mesh was very coarse and contained approximately 6000 cells. The time simulated was 100s and the simulation used the default set of relaxation parameters. No radiation model was used.

The non-controlled simulation required 9380 sweeps while the theoretical simulation length for the ICS 1.0 was only 4200 sweeps. The cost of the search was not recorded but it was estimated at 16,000 iterations. Consequently, the real simulation length was 20,000 (estimated) It was more than twice the length of the non-controlled run.

In comparison, the ICS 2.0 provided tangible benefits. First, the path to the solution found by the ICS required only 1696 sweeps which was a considerable improvement, compared to both the non-controlled (9380) and ICS 1.0-controlled (4200) simulations. Consequently, the theoretical speed improvement was 82% (compared with the non-controlled run). After including the cost of experiments (2007 iterations), the real speed improvement was 60.5%. Table 8-1 shows the relevant values for all three simulations. One can easily see that, although the ICS 1.0 found control parameters capable of reducing the simulation time, the cost of the search was 4 times larger than the cost of the simulation. Consequently, the simulation controlled by the ICS 1.0 ran twice as long as the non-controlled one. The improvement introduced to version 2.0 greatly reduced the search cost and improved the heuristic, which resulted in 60% reduction in simulation time. However, the search still remained computationally very expensive and used more sweeps than the actual simulation (2007 vs. 1696).

Figure 8-1 presents the convergence history for all three runs. The speed improvement makes it clear that the heuristics used by ICS 2.0 is much better than the previous version. Furthermore, the convergence history of the non-controlled simulation indicates that the default control parameters were not optimal for this case and caused substantial performance deterioration.

Type of control	Simulation sweeps	Search sweeps	Total sweeps	**Theoretical improvement**	**Real improvement**
Non-controlled	9380	N/A	9380	N/A	N/A
ICS 1.0 controlled	4200	~16000	~20000	**55.2%**	**~(-100%)**
ICS 2.0 controlled	1696	2007	3703	**81.9%**	**60.5%**

Table 8-1 Performance improvement comparison (54kW, coarse mesh)

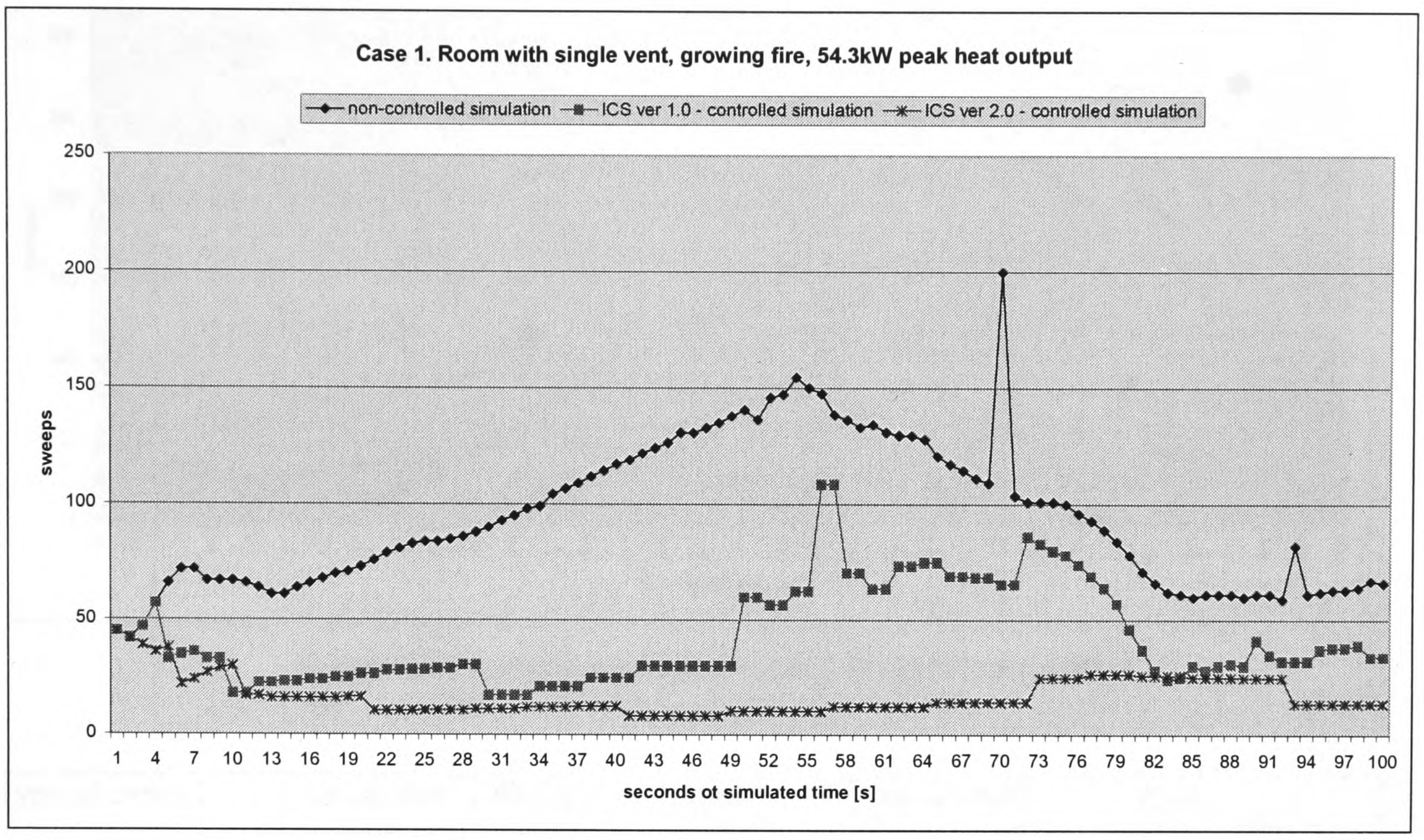


Figure 8-1 Convergence speed history (54kW, coarse mesh)

8.2.2 250kW case with fine mesh

This scenario was a variation of the previous case. The geometry remained unchanged (Steckler-size, 2.8m x 2.8m x 2.18m, fire in the centre of the room, single vertical vent). However, the fire was defined by a fast growing heat release curve with the peak heat output of 250kW (after 73s of simulated time). The mesh was substantially refined and contained approximately 20,000 cells. A fine mesh provides accurate and more detailed results but requires much longer processing time. Again, the simulation used the default set of relaxation parameters and the simulated time was 100s. Figure 8-2 presents the convergence history for the following runs:

- non-controlled,
- controlled by ICS 1.0,
- controlled by ICS 2.0 (experiments performed every 5s),
- controlled by ICS 2.0 (experiments performed every 10s).

The corresponding improvement factors are shown in Table 8-2.

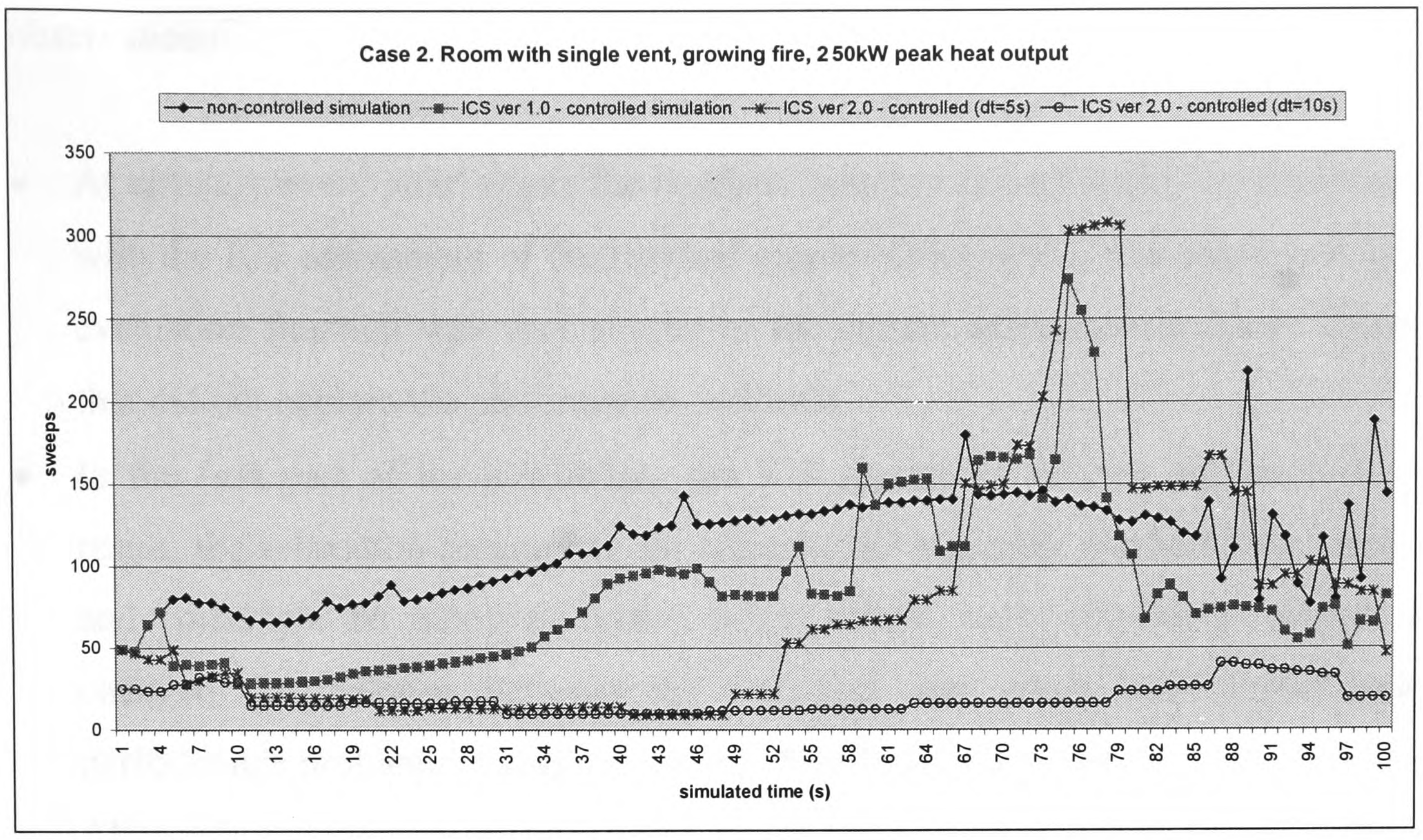


Figure 8-2 Convergence speed history (250kW, fine mesh)

Type of control	Simulation sweeps	Search sweeps	Total sweeps	**Theoretical improvement**	**Real improvement**
Non-controlled	11170	N/A	11170	**N/A**	**N/A**
Controlled by ICS ver 1.0	8373	~32000	~40000	**25%**	**~(-300%)**
Controlled by ICS ver 2.0 (dt = 5s)	7354	9168	16522	**34%**	**-48%**
Controlled by ICS ver 2.0 (dt = 10s)	1845	2306	4151	**83%**	**63%**

Table 8-2 Performance improvement comparison (250kW, fine mesh)

The graph clearly shows that the first run controlled by the ICS 2.0 failed to provide any significant improvements. In fact, during the period 70-100s the convergence was substantially slower than in the non-controlled simulation. The ICS 2.0 did manage to produce an overall theoretical speed improvement (34%) but with the search cost included, the complete simulation took almost 50% longer than the non-controlled run. This demonstrates that the control system is not infallible and that it can occasionally perform worse than a non-controlled run.

In order to determine the causes of the poor performance, the ICS-controlled simulation was thoroughly analysed by a CFD expert. The expert made several interesting observations:

- At virtually every point where the heuristic search was performed, the expert agreed with the ICS assessment of the residual graphs. Once again, this confirmed that the evaluation function was very similar to the human assessment method. However, that did not explain the performance problems.
- In the first part of the simulation, the ICS repeatedly relaxed the solution. As a result, the relaxation parameters for some of the variables reached their maximum and could not be modified further while others were still being relaxed. This changed the balance between the variables and could have contributed to performance problems during subsequent divergence recovery.
- Although the expert agreed with the assessment performed by ICS, he had reservations about the scheduling of the control actions. The expert suggested that some of the performance-oriented searches in the middle part of the simulation should not have been performed, as that was a period of a continuous increase in the heat release rate. It is believed that when the heat release rate is increasing any performance gains are short-lived and performance-oriented adjustments often result in divergence within a few steps.

As a result of this analysis, a second ICS-controlled run with a different set-up was performed. In order to reduce the number of performance-oriented actions, the search was executed every 10s of simulated time (as opposed to 5s in the first run). It was hoped that it would also alleviate the "out of balance" problem as fewer control actions should mean less chance of reaching maximum relaxation limits.

The results turned out to be surprisingly good. Firstly, the simulation did not experience any of the problems that plagued the previous runs. There was no abrupt performance deterioration and the convergence speed remained consistent throughout the whole simulation. Several time steps diverged but the divergence recovery was swift and efficient. Secondly, the speed improvement exceeded the expectation (83% theoretical improvement and 63% real improvement). The results strongly indicate that the problems identified by the expert were indeed responsible for the poor performance in earlier runs. However, these issues should be investigated more thoroughly and the conclusions used for enhancing the future versions of the ICS.

The 250kW simulation shows that the ICS 2.0 does not guarantee performance improvements in every case. It is not necessarily a design flaw but rather an inherent feature of a heuristic system. Like all such systems, ICS provides performance improvements most of the time but not all the time. Human experts also encounter occasional problems while controlling CFD simulation and therefore one should expect ICS to behave in a similar way (since the ICS was based on control methods used by them). It is believed that additional research should provide further improvements to the heuristics and consequently reduce the probability of performance degradation. It must also be stressed that even if there is no reduction in the simulation time ICS still assures full convergence of every time step.

8.2.3 2.1MW case with fine mesh

The third case consists of a big room (6m x 4m x 3.3m) with a fire in the centre of the room, 0.3m above the floor (on a stand) and a simple vertical vent (door). The fire is described by a growing heat release curve with the peak heat output of 2.1MW (after 450s). The mesh contains approximately 20,000 cells. All simulations used the default set of relaxation parameters and ran for 550s of simulated time. This case is significantly different from the previous ones as it contains a very big fire (2.1MW) and uses six-flux radiation model.

Type of control	Simulation sweep	Search sweeps	Total sweeps	**Theoretical improvement**	**Real improvement**
Non-controlled T = 5s	41,987	N/A	41,987	**N/A**	**N/A**
Non-controlled T = 1s	43,844	N/A	43,844	**N/A**	**N/A**
Controlled by ICS ver 1.0 (d_exp_t = 25s)	24,248	~25,000	~50,000k	**45%**	**~(-25%)**
Controlled by ICS ver 2.0 (d_exp_t = 25s)	13,771	3870	17641	**68%**	**60% (with t=1s)**

Table 8-3 Performance improvement comparison (2.1MW, medium mesh)

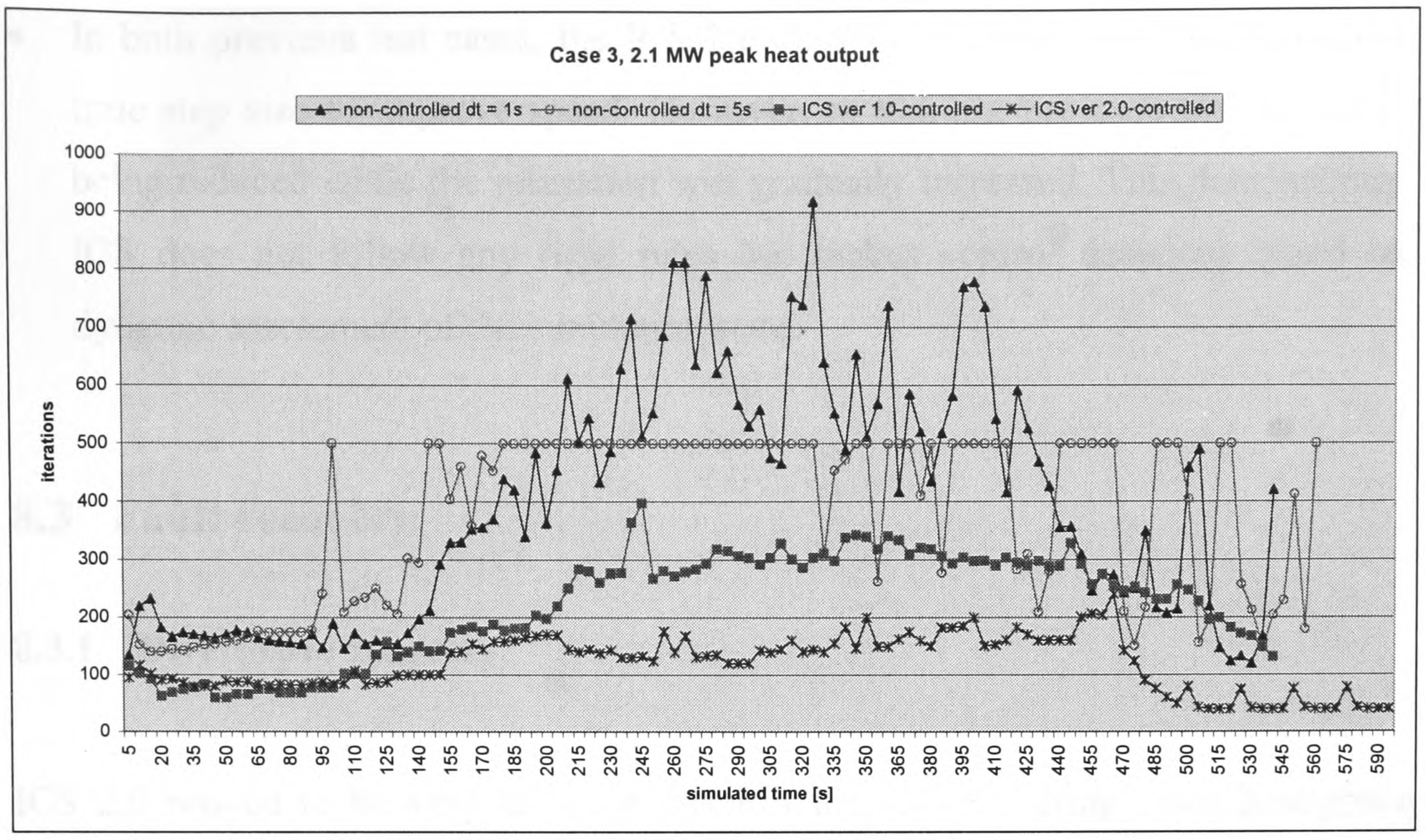


Figure 8-3 Convergence speed history (2.1MW, medium mesh)

The first three runs were already described in Chapter 6. The fourth run was set up to mirror the simulation controlled by the ICS 1.0. The results are presented in Figure 8-3 and Table 8-3. The convergence history graph clearly shows that there was a very good theoretical performance improvement (68%). Furthermore, the search did not require many iterations and consequently, the real improvement was also very significant: 60%. This case has some interesting properties, which make it different from the ones described previously:

- It was very difficult to make the non-controlled simulation fully converge. Even with a small time step size, almost 10% of time steps did not converge completely. However all time steps from the ICS-controlled simulation fully converged.
- It was the first simulation where the cost of experiments (heuristic search) was smaller than the cost of the simulation. In fact, only 22% of all iterations were used for search, with the remaining 78% producing the simulation results.
- It is easy to determine from the graph when the simulation reached a steady state (around 516s). From that point, the convergence speed remained constants with periodical "blips" introduced by search-related restarts. It is worth noting that no further adjustments were made after 516s even though a search was still being invoked every 25s of the simulated time. That suggests the final control parameters were very close to the optimal ones as no control actions were able to improve performance any further.

- In both previous test cases, the ICS 2.0 chose to increase both the relaxation and time step size to improve speed. However, in this big-fire case the time step was being reduced while the relaxation was gradually increased. This demonstrates that ICS does not follow any rigid rules but makes control decisions based on the dynamic assessment of the simulation state.

8.3 Fault recovery

8.3.1 Divergence recovery

ICS 2.0 proved to be very effective and efficient in recovering from divergence. In overwhelming majority of cases the full convergence was obtained after only two experiments. The convergence was normally restored by either reducing the time step size by 50% or removing 20% relaxation. Neither of these was significantly better than the other one. In some cases removing the relaxation worked better while in others reducing the time step had the advantage. Several examples of divergence recovery were presented in Chapter 6 and therefore are not repeated here.

8.3.2 Oscillation removal

Oscillation removal was not performed as often as divergence recovery. Consequently it is difficult to assess its effectiveness. When it was used, it required between 3 and 5 experiments to remove excessive oscillations. The most effective adjustments were similar to the ones used for restoring convergence: reducing the time step size by half or removing 20% relaxation. There was one case where only removing 50% relaxation and reducing time step size by 50% produced the required effect. Figure 8-4 shows an example of oscillation removal from a real simulation.

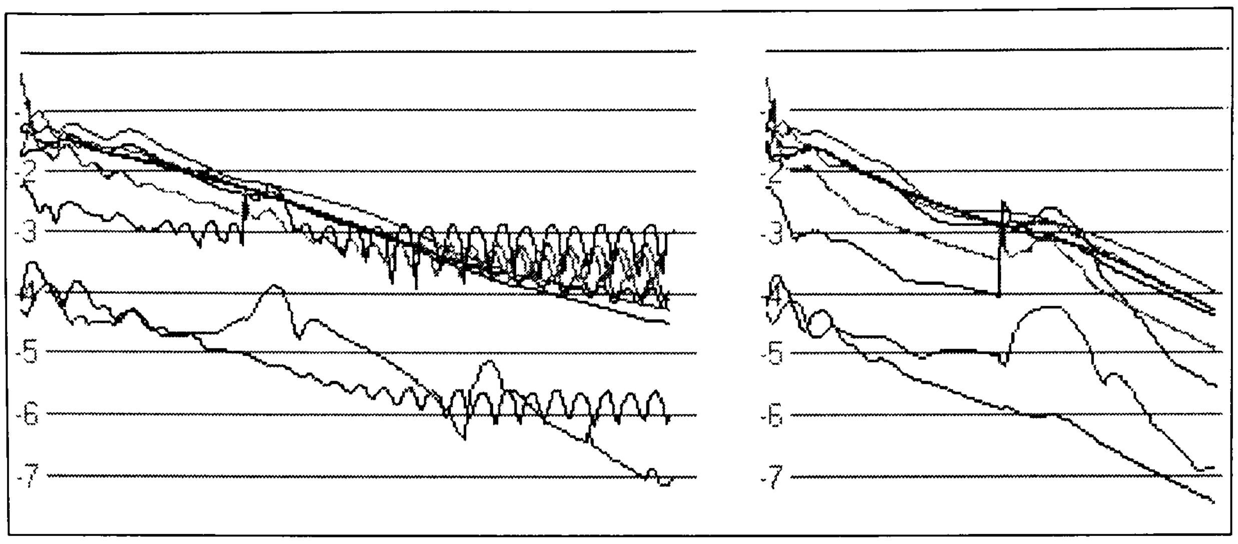


Figure 8-4 Oscillation removal by reducing the time step size by 50%

8.4 Accuracy assessment (Steckler case)

Accuracy of ICS-controlled simulations was assessed using Steckler fire case (Steckler-82). This case is very often used for the validation of fire models, as experimental results are readily available and can be compared with the model's predictions. As a result, this scenario is very well documented and researched, which helps assess the accuracy of the final results. Steckler case has other interesting properties, which suited our purposes very well:

- Short simulation time (due to simple geometry and a small fire).
- Readily available full case specification with correct mesh created by experts.
- Availability of results from a "golden-standard" case – identical scenario but simulated with fine mesh (100,000 cells) and low tolerance (high accuracy). This simulation was run as part of SMARTFIRE code validation (Grandison-01) and therefore did not have to be run again.
- Predefined areas from which the results had to be collected and compared: velocities and temperatures in the doorway and temperature in the corner of the room.

The scenario consisted of a small room (2.8m x 2.8m x 2.18m) with centrally located fire (defined as a cube with 0.3m height). There was a single vent – a doorway positioned on one of the walls. The vent measured 0.74m wide and 1.83m high. The fire was defined as a volumetric heat source with constant heat output 62.9kW. A six-flux radiation model was used. All cases ran for 200s of simulated time at which point a steady state solution was obtained. The mesh contained 10,000 cells.

At t = 200s, the following simulation results were stored for the purpose of accuracy assessment:

- Vertical temperature distribution in the room corner,
- Vertical temperature distribution in the centre of the doorway,
- Vertical velocities in the centre of the doorway.

Three different runs were performed:

1. Non-controlled simulation, convergence tolerance = 0.001.
2. Non-controlled simulation, convergence tolerance = 0.0001.
3. ICS 2.0-controlled simulation, convergence tolerance = 0.001.

In addition, the results from the "golden-standard" run (very fine mesh and low tolerance) are also shown in the graphs. The "golden-standard" simulation was performed as a part of SMARTFIRE validation.

8.4.1 Corner stack temperatures

The temperatures in the room corner are depicted in Figure 8-5. The results from the golden-standard simulation are significantly different from the other three runs using much coarser mesh. This could be due to a more sophisticated radiation model used in the golden-standard run. There is very little difference between the two non-controlled runs (tolerance 0.001 and 0.0001). Only close to the ceiling, the temperatures predicted by the more accurate run are slightly higher than in the other simulation.

The ICS-controlled simulation does not match the non-controlled results exactly. In the middle of the range the results are virtually identical, while at 0.8m and 0.6m the non-controlled simulations predict slightly higher temperature (by 3-4 K). Furthermore, in the upper layer, the ICS-run simulation predicted higher temperature than the other runs (by 2-3 K). Nonetheless, these differences are small and do not have a major effect on the accuracy of the final solution. It is also difficult to determine which run was more accurate as the comparison with golden-standard is inconclusive (in the upper layer, the results from the non-controlled simulation appear more accurate while in the remaining part of the graph the prediction from the ICS-controlled simulation are closer to the golden-standard). It is interesting that the results from two non-controlled runs (with different tolerances) are virtually identical. This suggests that the absolute convergence of each time step might not be as essential to the accuracy of the final solution as it is believed. It is also worth noting that the results from the more accurate non-controlled run (tolerance=0.0001) are closer to the results produced by the ICS than the ones from the other run (tolerance=0.001).

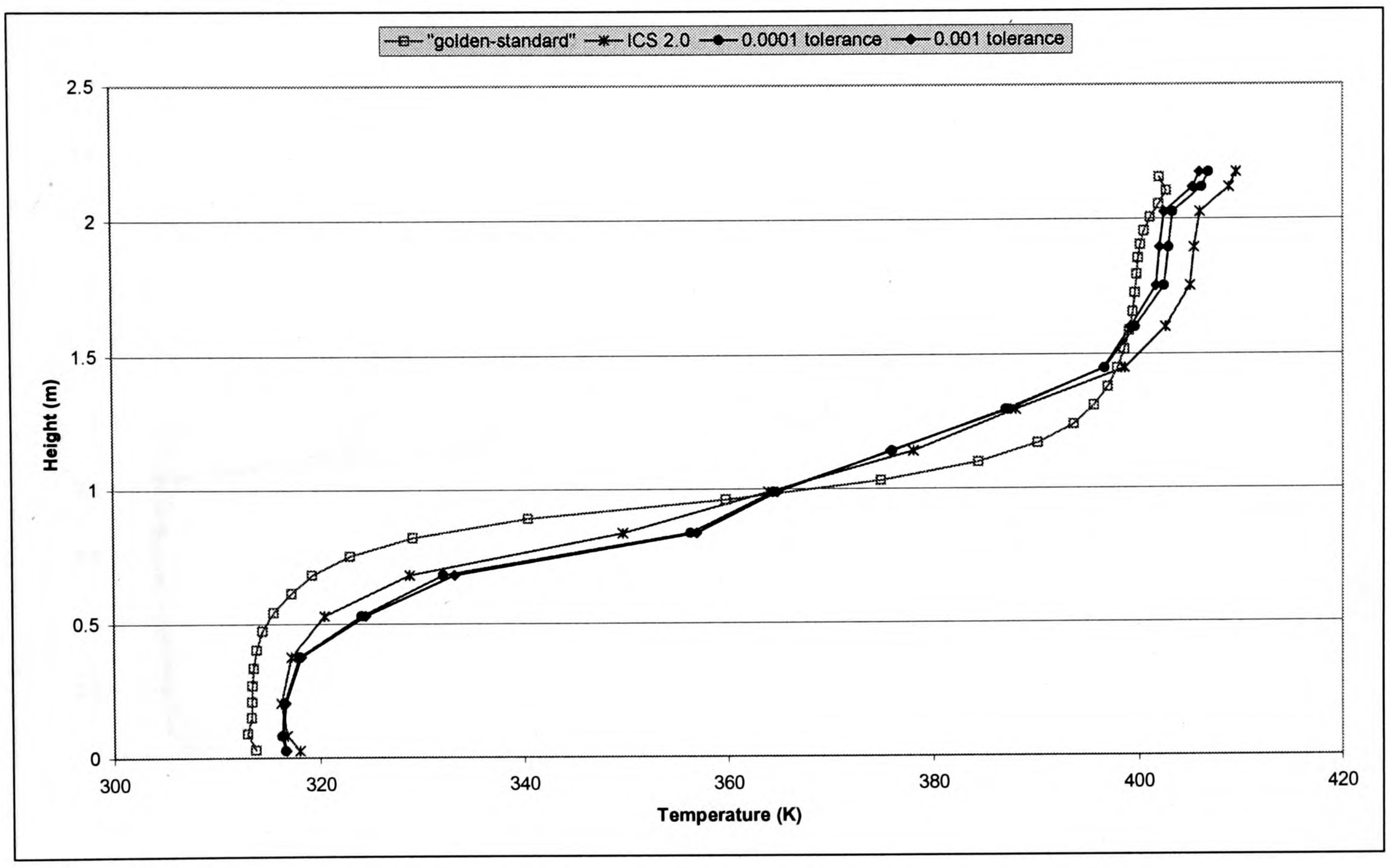


Figure 8-5 Comparison of corner stack temperatures at 200s

8.4.2 Doorway temperatures

Figure 8-6 shows the temperatures in the centre of the doorway (vertically). Again, there is very good agreement between both non-controlled runs. Generally, the golden-standard run produced similar predictions as the other simulations. The results are clearly different in the middle of the profile but the main reason for that is believed to be the fine mesh used in the "golden standard" case. The ICS 2.0 and non-controlled simulation predicted substantially different temperatures at the height of 1m (336 vs. 367 K). Again, the "golden-case" scenario cannot be used to determine which of those results is more accurate. Although the value 336K (ICS) appears to be matching the golden-case results better, this can be misleading as it only matches the interpolation between two cells. Consequently, it cannot be reliably assessed as more accurate.

In the lower section of the doorway as well as in the top section of the vent all the results are in very good agreement. Just as in Figure 8-5 (temperature in the corner), the ICS-controlled simulation predicted the highest temperature in the upper part of the domain.

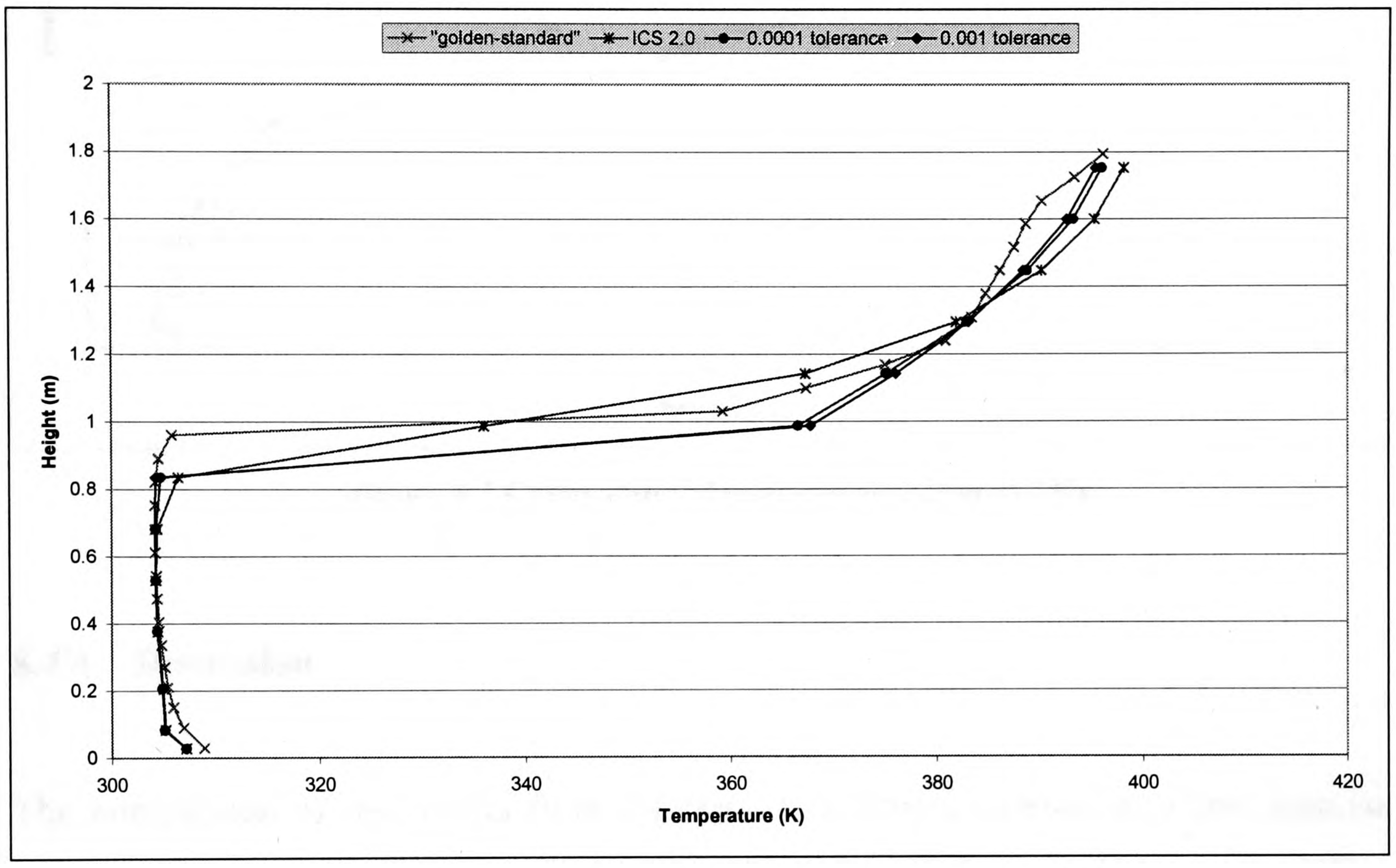


Figure 8-6 Comparison of doorway temperatures at 200s

8.4.3 Doorway velocities

Doorway velocities are presented in Figure 8-7. There is good agreement between all the simulations. In the lower part of the door, the non-controlled runs predict the highest velocity while the golden-standard results are the lowest, with the ICS in the middle. A similar situation occurs at the centre of the door (1m). In the upper section, the non-controlled simulations and the ICS-controlled one are virtually identical. Again, the results from the ICS-controlled simulations seem to match the golden-standard better than the non-controlled ones. However, this is not sufficient to declare them as more accurate.

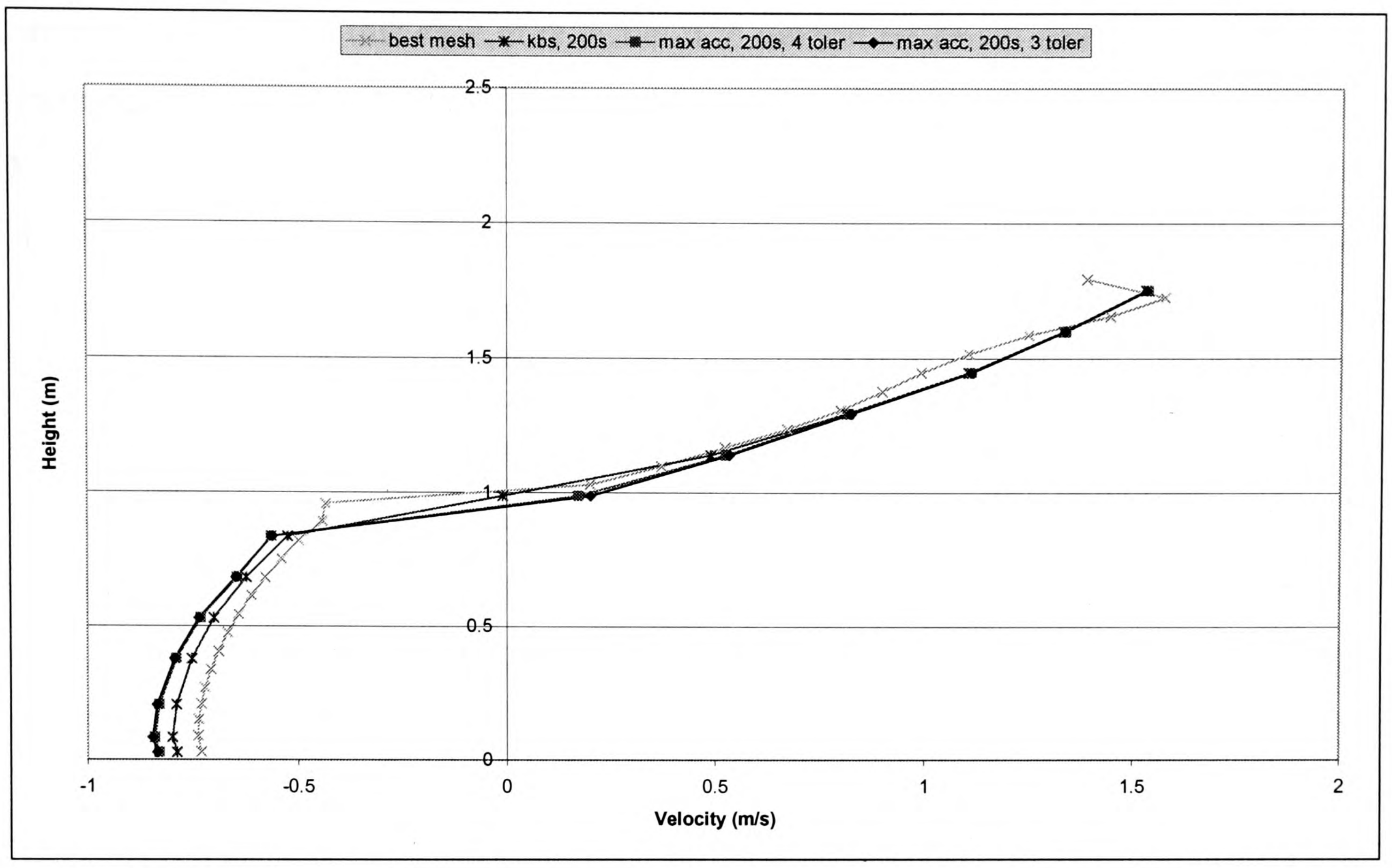


Figure 8-7 Comparison of doorway velocities at 200s

8.4.4 Discussion

The comparison of the results from different simulations resulted in a few surprising conclusions. It was believed that the two non-controlled simulations with different tolerance levels should produce similar but nevertheless different results. However, the analysis revealed they were virtually identical. The few differences that were observed never exceeded 1K in range 350-400K (and the differences in velocity were even smaller). Consequently, the assumption that the approximation error accumulates over several time steps and results in significant final differences was not confirmed. This might indicate that the full convergence of all time steps is not strictly necessary to obtain accurate final solution. Perhaps as long as none of the time steps clearly diverges, the final solution remains accurate. However, these are only speculations and further research is needed before the relationship between full convergence and accuracy of the final solution is better understood.

The data also showed that the results produced by the ICS-controlled simulation were slightly different than those produced by the fully converged non-controlled simulation using the same tolerance. This shows that the control process has some effect on the final solution. Using this case, it is difficult to determine which results (if any) are more accurate. Comparison with golden-standard results failed to produce clear answers. More research is needed to clarify this issue further. It is possible that the larger time step size used in the ICS-controlled simulation was responsible for the differences. Several other potential explanations of this behaviour are presented in the next chapter.

Nevertheless, it is clear, that the ICS-controlled simulation produced physically sound results, which were in good agreement with the non-controlled simulation using the same mesh, and with the golden-standard simulation. Further research is needed to reveal the cause of the observed discrepancies between automatically controlled simulation and non-controlled one.

8.5 Summary

This chapter presents a comparison between the non-controlled and ICS-controlled simulations. It demonstrates that the ICS 2.0 is capable of providing significant performance improvements over normal runs. In the best cases it reduced the number of iterations by over 60%. However, the heuristic search still remains rather expensive. The theoretical improvement factor suggests that even better performance improvements could be obtained if the cost of the search were reduced. The results indicate that up to 80% reduction in the number of iterations can be achieved in some cases.

The ICS 2.0 proved to be very competent in ensuring full convergence of all time steps and was also very successful in recovering from solution faults.

The simulation results show good agreement with those produced by non-controlled runs although some minor differences were observed. It proved difficult to determine which set of result was the most accurate. Consequently, this issue should be a subject of further research.

Chapter 9

Conclusions

9.1 Overview

In Chapter 1 we posed the questions that were to be answered in the course of this research. This chapter presents the summary of all our findings and attempts to answer the initial questions. First, we concentrate on the system's ability to emulate human experts in controlling a fire modelling software. Then we look at the ICS 2.0 in more detail and analyse the main benefits provided by the automated system:

- Significant performance improvements.
- Full convergence assurance.
- Reliable automatic recovery from solution faults.

Finally, we discuss whether the use of the ICS has any effect on the accuracy of the final solution.

9.2 Emulation of human control actions

This research demonstrated that the human ability to control the fire modelling software could be successfully emulated by an intelligent software agent. Furthermore, it was shown that an automated intelligent system could often provide superior results (both in term of speed and reliability) when compared to a human controlled simulation, as human experts tend to perform control actions less frequently and rarely investigate all possibilities available. It is believed that experts are capable of providing better results and their search technique is more efficient than that of the ICS but their knowledge is never fully applied, as they are not willing to commit the time and energy necessary to obtain better improvement. As a result, the intelligent agent based on the knowledge obtained from the experts proved to be capable of delivering much better performance, as it did not suffer from typical problems affecting human experts like short attention span, subjective assessment, low boredom threshold or tiredness.

During the initial research it became apparent that there was not enough knowledge available to build an efficient control system. Experts used a very intuitive approach, which varied significantly depending on the expert being interviewed. It was not known precisely what should be the main goal of the control procedure – some experts stressed the need for time performance enhancements while others emphasised improved stability of the simulation and accuracy of the solution. In order to gather additional empirical data regarding the subject, an extensive set of experiments was performed. These experiments revealed information that was not known to the experts before. For example, it was unveiled that by reducing the time step size one could improve the stability of the simulation without incurring performance penalty.

In the search for an appropriate model capable of emulating human control methods, a pure rule-based approach was quickly rejected as not reflecting the control methods used by human experts. Extensive analysis of human control actions and the experiment results gave rise to search-based architecture with a heuristic evaluation function. The underlying heuristic search represents an expanded model of the human control technique and incorporates methods used by experts for assessing the simulation quality. The solution is augmented by further AI techniques like planning (with dynamic rule-driven plan modification) or pattern recognition.

During the process of knowledge acquisition it was established that the residual graphs were the most important source of information about the simulation state. The interviews with experts helped identify the main features in the graphs that influenced their control decisions:

- Convergence speed.
- Scale of irregularities present in the graph.
- Oscillations in residual values.

As a result, an appropriate evaluation function was created, which made use of advanced pattern recognition algorithms designed to extract those features from residual graphs. The function was tested on the data acquired during the experiments and then, after successful validation, used in real simulations. After initial tests the heuristic function was further improved and it is now believed to be similar to the intuitive technique used by experts in residual graph assessment. Its suitability was further confirmed by comparing the assessment results between the automated system and a human expert. Furthermore, significant improvements obtained by ICS using the new evaluation function provided the final and ultimate validation.

9.3 Speed improvements

The test cases show that the final version of ICS provides substantial reductions in simulation time. In the best cases the system was able to find a set of control parameters capable of reducing the simulation time by up to 86% (34% in the worst case so far) when compared to the non-controlled simulation. It is a substantial improvement, as the 86% reduction means that a simulation that normally takes a week could be completed in a single day. However, taking these improvement figures at their face value is misleading since they do not represent the *real performance improvement*. These figures describe the *potential reduction in simulation time* but not the *real reduction in execution time*. The appropriate control parameters have to be found before they can be applied hence the control system first performs a heuristic search in order to obtain the appropriate set of parameters. The computational cost of the search is substantial and can sometimes outweigh the cost of the actual simulation. Therefore, to calculate the real reduction in the execution time one has to take into account the cost of the search as well as the simulation. Consequently, the *real improvement* is smaller than the *potential reduction* but the difference between them varies depending on the case and/or the initial control parameters. In one of the simulations the potential improvement was 83% and the real improvement was 63%. It is not uncommon for the search cost to be larger then the simulation cost. In the example mentioned above, the real simulation required 1845 sweeps while the heuristic search used 2306. Therefore, 25% more time was spent on searching for control parameters than on performing the CFD simulation.

The cost of the heuristic search is usually considerable and therefore there is no guarantee that the actual reduction in the simulation time will be obtained. In all test cases so far performed the ICS was always able to achieve a theoretical speed improvement, but on a few occasions the execution time was longer than in the non-controlled simulation due to the excessive cost of the search. In one instance the theoretical improvement was 34% but the simulation took almost 50% more time than the non-controlled one because of the heuristic search cost. In such cases the reduction of the search frequency is normally sufficient to bring the cost of the experiments down. Several methods of reducing the search cost while still delivering improved performance were proposed and the relevant details are presented in Chapter 10. The majority of cases however, do show substantial *real speed improvement* with the reduction in simulation time reaching 60% whilst some of the techniques outlined in Chapter 10 promise to further reduce the search cost.

9.4 Reliability and fault recovery

In all ICS-controlled simulations the full convergence of every time step was assured as the current version of the system is designed to guarantee reaching the default tolerance in every time step. When convergence cannot be achieved the system will progressively reduce the time step size until it is reached. However, if the specified tolerance is unattainable then the simulation will stall. This is a preferred behaviour as it prevents a non-converged time step from slipping through undetected.

The ICS completely eliminates faulty runs (fairly common in non-controlled simulations) that diverge and produce inaccurate results. This feature alone can generate huge savings in run time, as the non-controlled simulations are often left unsupervised for a long period (a weekend for example) and consequently, faults are detected long after they have occurred in which case the whole simulation has to be restarted and run again. The ICS-controlled simulations do not suffer from that problem since divergence is automatically detected and appropriate corrective actions are triggered. The ICS proved to be very competent in divergence recovery and was always able to find an effective solution to all divergence problems it so far faced. Consequently, automatically controlled simulations are much more robust and can usually obtain correct final results regardless of the initial control parameters.

The ability of the ICS to recover from solution excursions and faults was thoroughly tested. Divergence occurs fairly often in automatically controlled simulations since the performance considerations require aggressive control actions, which in turn increases the danger of divergence. Nevertheless, the problems were properly detected and the appropriate recovery strategy was initiated. The recovery procedure restores convergence and at the same time attempts to minimise the adverse effects on performance. In all the tests the system was able to recover from divergence competently and efficiently. This is another important feature of the ICS, as uncorrected divergence can seriously affect the accuracy of the results. Furthermore, in non-controlled simulations divergence is often easy to overlook.

9.5 Results accuracy

This research was based on the assumption (shared by all experts) that the accuracy of the results can be guaranteed by ensuring that every time step fully converges (i.e. all residual errors reach the required tolerance). It was therefore believed that since all time steps in ICS-controlled simulations converge, the accuracy of the results would be superior to the non-controlled simulations. However, a dedicated comparison study did not provide conclusive evidence to back this claim. It was confirmed that the ICS-controlled simulation produced physically sound results, which were in good agreement with a non-controlled run using the same mesh, and with a golden-standard simulation (i.e. using a fine mesh). Some results produced by the ICS-controlled run were closer to the golden-standard whilst in other areas the non-controlled simulation seemed more accurate. Consequently, it was impossible to determine which results were better. The study, however, revealed that continuous changes to the control parameters did have some influence on the results. Further research is necessary to determine the exact effect of various control actions on accuracy of the results. There are three potential factors, which experts believe to be significant:

- Changes in time step size may affect the solution process by excluding important but short-duration flow features (if the time step size is increased) or including them (if the time step size is decreased). This inclusion or exclusion may have some influence on the final results.

- It is possible that changes to relaxation parameters affect the convergence assessment (e.g. it is widely known that massive under-relaxation can stagnate the solution giving a false impression of convergence). However, in all the test cases the ICS has never substantially under-relaxed the solver.
- Perhaps implementation glitches affect the results accuracy – for instance, the restart procedure (performed many times in an ICS-controlled simulation) may introduce subtle inaccuracies which are compounded in the course of several time steps and restarts.

Despite all the reservations, the results produced by the ICS-controlled simulation proved to be physically correct and sufficiently accurate. The control architecture presented in this thesis was validated on a difficult problem and it may also prove useful when applied to other numerical packages as a tool for enhancing their performance and reliability.

Chapter 10

Further work

10.1 Introduction

Although the final version of the ICS presented in this dissertation answered all the research questions and provided results that exceeded the initial expectations, it also opened a way to further enhancements that promise to deliver even better performance and more effective solution control. Several very promising opportunities for further work were identified and this chapter presents a brief description of these ideas. Most of the recommendations contained here focus on the reduction of the search time (as it remains the most significant factor affecting the performance) but some other issues are also mentioned.

10.2 Flexible experiment scheduling

It is believed that substantial benefits can be obtained by introducing a more sophisticated scheduling of the search throughout the simulation. Currently, the look-ahead search is performed with a predetermined frequency, which has detrimental effect on performance. A more flexible scheduling strategy could reduce the computational cost associated with the search. Perhaps several consecutive searches should be performed at the beginning of the simulation in order to obtain optimum control parameters for the particular case, while in its the later stages they would normally be performed less frequently. In a more advanced approach the system might try to customise the search frequency depending on the simulation state (e.g. current rate of change in heat release). This solution could be further enhanced through the introduction of an algorithm capable of recognising flow features in the simulated domain (like flashover, decay phase or circular flow) and adjusting the search frequency accordingly. However, this would make the code less generic, as the rules would have to be implementation-specific. Furthermore, flexible scheduling only applies to the performance-oriented searches as the recovery searches are always triggered by an anomalous graph and therefore not scheduled.

10.3 Using "spread-out search" technique

In the current version of the system, the simulation results obtained during the search are discarded and the time step is restarted using the best control parameters found. This is clearly inefficient, as the experimental results could be reused. Consequently, the final state of each experiment should be saved and when the search is finished, the simulation would use the results from the most advanced experiment as the restart point when the new control parameters are applied to the next time step. Since the result of each experiment would have to be saved, there is a significant cost in both space and time, which should be investigated to determine whether this solution is feasible.

A more advanced version of the above strategy is a technique provisionally called "spread-out search". If successfully implemented it promises the ultimate performance with a minimum search overhead. In the current system each search is performed at a discrete point in the simulation and all the computational effort expended during the search is used exclusively for determination of better control parameters, which does not advance the actual simulation. A much more efficient approach is to advance the simulation while searching for the improved set of parameters at the same time. In that case the search is performed over several time steps (becomes spread-out) and only the results produced by diverged experiments are discarded. This approach needs substantial research since it requires a reliable evaluation function that can compare several consecutive time steps. Furthermore, this technique can only be used when the simulation state does not change substantially over several time steps, which is not always the case. However, it is expected that a system capable of using such a technique would provide superior performance.

10.4 Assessment of experiment cost vs. expected gain

Some of the experiments (especially at the end of the simulation) are unlikely to provide performance improvement that would justify the cost associated with the particular experiment. A dynamic assessment of the predicted cost vs. potential gain might reduce the number of experiments run and consequently, reduce the overall simulation time by improving the efficiency.

10.5 Other areas for improvement

The following list presents other areas that should benefit from further research:

- Improving the feature extraction algorithms can lead to increased accuracy of the heuristics (evaluation function).
- More effective pruning of the search tree, e.g. using historical data to assist in selecting the experiments that are most likely to produce improvements.
- Researching the dependencies between variables may lead to the development of variable-specific control actions. This would require separate assessment procedure for each variable and therefore would make it possible to perform control actions that are more accurate and targeted at the source of the problem.
- As more data becomes available, it should be possible to further improve the rules used to dynamically modify the search plan, which should lead to more effective search tree pruning.

References

[Anderson-95]

Anderson J D, *Computational Fluid Dynamics: The Basics with Applications,* Mc Graw Hill, 1995

[Autere-97]

Autere A, Lehtinen J, *Admissible heuristics for robot motion planning using A* algorithm*, Proceedings of Applications of Artificial Intelligence in Engineering XI, USA, 1996

[Borenstein-91]

[Borenstein J, Koren Y, *The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots*, IEEE Journal of Robotics and Automation, Vol. 7, No. 3, pp 278-288, June 1991

[Bracewell-65]

Bracewell R, *The Fourier Transform and Its Applications*, McGraw-Hill, U.S.A., 1965

[Brandt-77]

Brandt A, *Multi-level adaptive solutions to boundary value problem*, Math. Computing, Issue 31, pp. 333-390, 1977

[Brandt-73]

Brandt A, *Multi-level adaptive technique (MLAT) for fast numerical solution in boundary value problem*, Proceedings of the 3rd International Conference on Numerical Methods in Fluid Mechanics, Vol. 1, pp. 82-89, Springer, Berlin, 1973

[Cai-97]

Cai X, Keyes D E, and Venkatakrishnan V, *Newton-Krylov-Schwarz: An implicit solver for CFD*, in Proceedings of Eighth International Conference on Domain Decomposition Methods (R. Glowinski et al., eds.), Wiley, New York, 1997, pp. 387-400.

[Choi-97]

Choi Y, Merkle C, *The application of preconditioning to viscous flows*, Journal of Computational Physics, 105, pp. 207-223, 1993

[Chatfield-85]

Chatfield C, *The Analysis of Time Series: An Introduction*, Chapman and Hall, London, 1985

[Cuthill-69]

Cuthill E, McKee J, *Reducing the band width of sparse symmetric matrices*, Proceedings of ACM Nat. Conference, 1969, pp. 157-172.

[Earnshaw-85]

Earnshaw R A, *A Review of Curve Drawing Algorithms*, Proc. of the Advanced Study Institute on Fundamental Algorithms for Computer Graphics, Edited by Earnshaw R A, Springer-Verlag, Berlin, 1985

[Elnagar-95]

Elnagar A, Hay J, *A heuristic approach for local path planning in 3D environments*, Proceedings of ICECS International Conference on Electronics, Circuits and Systems, pp. 326-332, Jordan, Oman, 1995

[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.

[Ewer-99a]

Ewer J, Galea E, Patel M, Taylor S, Knight B and Petridis M, *SMARTFIRE: An Intelligent CFD Based Fire Model*, Journal of Fire Protection Engineering, Vol. 10, No. 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, Proc. Interflam 99, Edinburgh, UK, June/July 1999, Vol. 2, pp 939-950

[Ewer-99c]

Ewer J, Galea E, Patel M and Knight B, *Enhancing the Numerical Performance of Fire Field Models*, CMS Press, Paper No. 99/IM/52, 1999

[Ewer-98]

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

[Ewer-93]

Ewer J, Petridis M, Cowell D and Knight B, *An Intelligent User Interface for Computational Fluid Dynamics Software*, Proceedings of AIENG'93, pp 77-92, 1993

[Fedorenko-64]

Fedorenko R P, *The speed of convergence of one iterative process*, Computational Mathematics and Mathematical Physics, 4(3), pp. 227-235, USSR, 1964

[Fiorini-98]

Fiorini P, Shiller Z, *Motion planning in dynamic environments using velocity obstacles*, International Journal of Robotics Research, vol. 17, issue 7, pp. 760-772, Sage Publications, USA, 1998

[Fletcher-64]

Fletcher R, Reeves C M, *Function Minimization by Conjugate Gradients*, Computer Journal, Issue 7, 149-154, 1964

[Floyd-67]

Floyd R W. *Nondeterministic Algorithms*, Journal of the ACM, 14(4), pp. 636-644, October 1967

[FLOW3D]
Flow3d - CFD Flow modelling software, Flow Science Inc., Santa Fe, New Mexico, USA, (http://www.flow3d.com)

[FLUENT]
Fluent - CFD Flow modelling software, Fluent Inc., Lebanon, New Hampshire, USA, (http://www.fluent.com)

[Galea-99]
Galea E, Knight B, Patel M, Ewer J, Petridis M and Taylor S, *SMARTFIRE V2.01 build 365, User Guide and Technical Manual*, Smartfire CD and bound manual, 1999

[Galea-96]
Galea E.R, *On the field modelling approach to the simulation of enclosure fires, Journal of Fire Protection Engineering*, vol 1 (1), pp 11-22

[Grandison-01]
Grandison A J, Galea E R, Patel M K, *Fire Modelling Standards/Benchmark*, University of Greenwich, CMS Press, 2001

[Hackbush-85]
Hackbush W, *Multi-grid methods and applications*, Springer, Berlin, 1985

[Hackbush-76]
Hackbush W, *On the convergence of a multi-grid iteration applied to finite element equations*, Universitat Koln, Report 77-8

[Hestenes-51]
Hestenes M R, *Iterative Methods for Solving Linear Equations* [originally published in 1951 as NAML Report No. 52-9], Journal of Optimization Theory and Applications, no. 4, 323–334, 1973

[Hiraishi-98]

Hiraishi H, Ohwada H, Mizoguchi F, *Time-constrained heuristic search for practical route finding*, Proceedings of 5th Pacific Rim International Conference PRICAI'98: Topics in Artificial Intelligence, pp.389-398, Springer-Verlag, Berlin, Germany, 1998

[Inoue-91]

Inoue Y, Yoshimura T, Kitamura S, *The collision avoidance algorithm of manipulators based on global subgoals and a heuristic graph search*, Transactions of the Society of Instrument and Control Engineers, vol. 27, issue 8, pp. 922-928, 1991, Japan

[Janes-01]

Janes D, Ewer J, Galea E, Patel M and Knight B, *Automatic Dynamic Control of CFD Based Fire Modelling Simulations*, Interflam 2001 Conference Proceedings, Vol. 1, pp 811 - 822

[Jia-99]

Jia F, PhD Thesis: *The Simulation of Fire Growth and Spread within Enclosures using an Integrated CFD Fire Field Model*, University of Greenwich, CMS School, UK, 1999

[Jia-97]

Jia F, Galea E and Patel M, *The Prediction of Fire Propagation in Enclosure Fires*, Fire Safety Science - Proc. 5th Intl. Symp., 1997, pp 439-450

[Karp-72]

Karp R M, *Reducibility among combinatorial problems*, in Miller R E and Thatcher J W, editors, Complexity of Computer Computations, pp. 85-103, Plenum, New York, 1972

[Kerrison-94]

Kerrison L, Mawhinney N, Galea E, Hoffman N and Patel M, *A Comparison of Two Fire Field Models With Experimental Room Fire Data*, Fire Safety Science – Proc. Of the Fourth Intl. Symp., Ottawa, Canada, July 1994, pp 161-172

[Keyes-98]

Keyes D E, Kaushik D K, Smith B F, *Prospects for CFD on Petaflops Systems*, ICASE Report No. 97-73, Virginia, USA, 1997

[Knight-91]

Knight B and Petridis M, *A Design for Reliable CFD Software*, Reliability and Robustness of Engineering Software II, Ed. Brebbia C, Ferrante A, pp 3-17, Elsevier, 1991

[Knupp-94]

Knupp P, Steinberg S, *Fundamentals of Grid Generation,* CRC Press, 1994

[Koenig-98]

Koenig S, Simmons R G, *Solving robot navigation problems with initial pose uncertainty using real-time heuristic search*, Proceedings of 4th International Conference on Artificial Intelligence Planning Systems, pp. 145-153, Atlanta, USA, 1998

[Lawler-85]

Lawler, E L, Lenstra J K, Rinnooy Kan A H G and Shmoys D B, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, 1985

[Lee-97]

Lee D, Lynn J F, van Leer B, *A local Navier-Stokes Preconditioner for all Mach and Cell Reynolds Numbers*, AIAA Paper 97-2024, June 1997

[Lee-93]

Lee D, van Leer B, *Progress in Local Preconditioning of the Euler Navier-Stokes Equations*, AIAA Paper, 93-3328, July 1993

[Le Riche-69]

Le Riche P J, *Procedure Curve*, Computer Journal, pp 291, 1969

[Lohner-97]

Lohner R, *Renumbering strategies for unstructured grid solvers operating on shared-memory cache-based parallel machines*, in Proceedings of the 13th AIAA CFD Conference, Snowmass, CO, June 1997, pp. 1015-1025. AIAA Paper 97-2045-CP.

[Loyd-59]

Loyd S, *Mathematical Puzzles of Sam Loyd: Selected and Edited by Martin Gardner,* Dover, New York, 1959

[Masters-95]

Masters T, *Neural, Novel & Hybrid Algorithms for Time Series Prediction*, John Wiley & Sons, New York, 1995

[Masters-94]

Masters T, *Signal and Image Processing with Neural Networks*, John Wiley & Sons, New York, 1995

[Mavriplis-98]

Mavriplis, D J, *On convergence acceleration techniques for unstructured meshes*, ICASE Report No. 98-44, Virginia, USA, 1998

[Michie-66]

Michie D, *Game-playing and game-learning automata*, in Fox L, editor, Advances in Programming and Non-Numerical Computation, pp. 183-200. Pergamon, New York, 1966

[Moin-97]

Moin P, Kim J, *Tackling turbulence with supercomputers*, Scientific American, January 1997

[Montgomery-76]

Montgomery D C, Johnson L A, *Forecasting and Time Series Analysis*, McGraw-Hill, U.S.A., 1976

[Morano-93]

Morano E, Dervieux A, *Looking for O(N) Navier-Stokes solutions on non-structured meshes*, 6th Copper Mountain Conference on Multigrid Methods, pp. 449-464, 1993

[Ollivier-95]

Ollivier-Gooch C, *Towards problem-independent multigrid convergence rates for unstructured mesh methods in inviscid and laminar flows*, Proceedings of the 6th International Symposium on CFD, Lake Tahoe, NV, Sept. 1995

[Ott-93]

Ott R L, *An Introduction to Statistical Methods and Data Analysis*, Wadsworth Inc., U.S.A., 1993

[Pantakar-80]

Pantakar S V, *Numerical Heat Transfer and Fluid Flow*, Intertext Books, McGraw Hill, New York, 1980.

[Pearl-84]

Pearl J, *HEURISTICS: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, London, 1984

[Petridis-95]

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

[Petridis-93]

Petridis M and Knight B, *A Blackboard Approach for the Integration of an Intelligent Knowledge Based System into Engineering Software*, Knowledge Based Systems for Civil and Structural Engineering, Ed Topping B, pp 49-56, Civil Comp Press, 1993

[Petridis-92]

Petridis M, Knight B, *FLOWES: An Intelligent CFD System*, Engineering Applications of Artificial Intelligence, Vol 5(1), pp 51-58, 1992

[Pierce-96]

Pierce N, Giles M, *Preconditioning on stretched meshes.* AIAA paper 96-0889, Jan 1996

[Press-92]

Press, W H, Teukolsky S A, Vetterling W T, Flannery B P, *Numerical Recipes in C* (Second Edition), Cambridge University Press, USA, 1992

[Rausch-95]

Rausch W, Bonhaus D, *Implicit multigrid algorithms for incompressible turbulent flows on unstructured grids*, Proceedings of the 12th AIAA CFD Conference, AIAA Paper 95-1740-CP, San Diego CA, June 1995

[Reuman-74]

Reuman K, Witham A P M, *Optimising Curve Segmentation in Computer Graphics*, Proceedings of the International Computing Symposium 1973, Edited by Gunther A, North Holland, pp 467-472, 1974

[Shaw-92]

Shaw, C T, *Using Computational Fluid Dynamics*, Prentice Hall, 1992

[Shewchuk-94]

Shewchuk J R, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, August 1994, Shewchuk,J.R., School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, (available at http://www.cs.cmu.edu/~jrs/jrspapers.html).

[Shoval-94]

Shoval S, Borenstein J, Koren Y, *Mobile Robot Obstacle Avoidance in a Computerized Travel Aid for the Blind*, Proceedings of the 1994 IEEE Robotics and Automation Conference, San Diego, California, 1994, pp. 2023-2029.

[STARCD]

StarCD - CFD Flow modelling software, CD-adapco group, (http://www.cd-adapco.com)

[Steckler-82]

Steckler K, Quintere J and Rinkinen W, *Flow Induced By Fire in a Compartment*, NBSIR 82-2520, National Bureau of Standards, Washington, 1982

[Stiefel-52]

Stiefel E, *Uber einige Methoden der Relaxationsrechnung*, Zeitschrift fur Angewandte Mathematik und Physik, no. 1, 1–33, 1952

[Tatsuya-96]

Tatsuya A, Hiroyuki U, Rie O, Shiro I, Isao T, *Fuzzy control to accelerate numerical simulation of heat transfer and fluid flow*, Ishikawajima-Harima Engineering Review, Japan, 1996, vol. 36, issue 5, pp. 341-346

[Taylor-97a]

Taylor S, *PhD Thesis: An investigation into Automation of Fire Field Modelling Techniques*, University of Greenwich, CMS School, UK, September 1997

[Taylor-97b]

Taylor S, Petridis M, Knight B, Ewer J, Galea E and Patel M, *SMARTFIRE: An Integrated CFD Code and Expert System for Fire Modelling*, Fire Safety Science, Proceedings of the 5th Intl. Symp. Ed. Hasemi Y, 1997, pp1285-1296

[Taylor-96]

Taylor S, Galea E.R, Patel M, Petridis M, Knight B and Ewer J, *SMARTFIRE: An Intelligent Fire Field Model*, Proc Interflam 96, Cambridge, UK, March 1996, pp 671-680

[Taylor-95]

Taylor S, Galea E.R, Patel M, Petridis M, Knight B and Ewer J, *SMARTFIRE: An Integrated CFD environment using knowledge based reasoning for fire simulation modelling*, Conference Abstracts, First European Symp on Fire Safety Science, Zurich, August 1995

[Turkel-87]

Turkel E, *Preconditioned Methods for Solving the Incompressible and Low Speed Compressible Equations*, Journal of Computational Physics, Vol. 72, pp. 277-298, 1987

[Ulrich-00]

Ulrich I, Borenstein J, *VFH*: Local Obstacle Avoidance with Look-Ahead Verification*, Proceedings of the 2000 IEEE International Conference on Robotics and Automation, San Francisco, CA, 2000, pp. 2505-2511

[Ulrich-98]

Ulrich I, Borenstein J, *VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots*, Proceedings of the 1998 IEEE International Conference on Robotics and Automation, Leuven, Belgium, May 1998, pp 1572-1577

[Venkatakrishnan-93]

Venkatakrishnan V, Mavriplis D J, *Implicit solvers for unstructured meshes*, Journal of Computational Physics, 105 (1993), pp. 83-91.

[Wang-99]

Wang Z, Jia F, Galea E R, Patel M and Ewer J, *Simulating One of the CIB W14 Round Robin Test Cases using the SMARTFIRE Fire Field Model*, CMS Press, December 1999

[Wendt-92]

Wendt J F (Editor), Anderson J D, Degrez G, Dick E, and Grundmann R, *Computational Fluid Dynamics-An Introduction*, Springer-Verlag, 1992

[Wesseling-92]

Wesseling P, *An Introduction to Multigrid Methods*, John Wiley & Sons, Chilchester, 1992

[Zhang-97]
Zhang J, *PhD Thesis: Multigrid acceleration techniques and applications to the numerical solution of partial differential equations*, George Washington University, USA, 1997

[Zingg-97]
Zingg D, Pueyo A, *An efficient Newton-GMRES solver for aerodynamic computations*, Proceedings of the 13th AIAA CFD Conference, AIAA Paper 97-1955-CP, pp. 712-721, June 1997

Appendix A

Rules used in search tree pruning

Divergence rules

General rule template:

If applying ***<u><changes A></u>*** led to divergence then applying ***<u><changes B></u>*** would also lead to divergence.

No	Changes A			Changes B		
	Lin. relax.	FTS relax.	Time step	Lin. relax.	FTS relax.	Time step
1	+50%	+70%	0%	+50%	+70%	+100%
2	+50%	+70%	0%	+50%	+70%	+50%
3	+20%	+30%	0%	+50%	+70%	0%
4	+20%	+30%	0%	+50%	+70%	+100%
5	+20%	+30%	0%	+20%	+30%	+100%
6	+20%	+30%	0%	+50%	+70%	+50%
7	+20%	+30%	0%	+20%	+30%	+50%
8	0%	0%	+100%	+50%	+70%	+100%
9	0%	0%	+100%	+20%	+30%	+100%
10	0%	0%	+50%	0%	0%	+100%
11	0%	0%	+50%	+50%	+70%	+100%
12	0%	0%	+50%	+20%	+30%	+100%
13	0%	0%	+50%	+50%	+70%	+50%
14	0%	0%	+50%	+20%	+30%	+50%
15	+20%	+30%	+100%	+50%	+70%	+100%
16	+50%	+70%	+50%	+50%	+70%	+100%
17	+20%	+30%	+50%	+50%	+70%	+100%
18	+20%	+30%	+50%	+20%	+30%	+100%
19	+20%	+30%	+50%	+50%	+70%	+50%
20	+50%	+70%	-50%	+50%	+70%	0%
21	+50%	+70%	-50%	+50%	+70%	+100%
22	+50%	+70%	-50%	+50%	+70%	+50%
23	+20%	+30%	-50%	+50%	+70%	0%

24	+20%	+30%	-50%	+20%	+30%	0%
25	+20%	+30%	-50%	+50%	+70%	+100%
26	+20%	+30%	-50%	+20%	+30%	+100%
27	+20%	+30%	-50%	+50%	+70%	+50%
28	+20%	+30%	-50%	+20%	+30%	+50%
29	+20%	+30%	-50%	+50%	+70%	-50%

Convergence speed rules

General rule template:

If applying ***<changes A>*** led to performance improvement then applying ***<changes B>*** would not provide better improvement.

No	**Changes A**			**Changes B**		
	Lin. relax.	FTS relax.	Time step	Lin. relax.	FTS relax.	Time step
1	+50%	+70%	0%	+20%	+30%	0%
2	+50%	+70%	+100%	+20%	+30%	+100%
3	+50%	+70%	+100%	0%	0%	+100%
4	+20%	+30%	+100%	0%	0%	+100%
5	+50%	+70%	+50%	0%	0%	+50%
6	+50%	+70%	+50%	+20%	+30%	+50%
7	+20%	+30%	+50%	0%	0%	+50%
8	+50%	+70%	-50%	+20%	+30%	-50%

The following is an abridged version of the Technical Reference for SMARTFIRE [SMARTFIRE-03], which we include with the kind permission of John Ewer of the University of Greenwich.

Appendix B

TECHNICAL REFERENCE FOR *SMARTFIRE*

NOMENCLATURE

Symbol	Meaning	Equation of first mention
$\underline{U}$	Velocity	1.1
T	Time	1.1
u_I	Ith velocity component	1.2
P	Pressure	1.2
S_I	Ith variable/model source term	1.2
x_I	Ith co-ordinate direction	1.2
H	Enthalpy	1.3a
K	Conductivity	1.3a
C_p	Specific heat capacity	1.3a
T	Temperature	1.3b
K	Kinetic energy	1.4a
P	Turbulent production rate	1.4a
G	Buoyancy rate	1.4a
$C_{1\varepsilon}$	Turbulent constant	1.4b
C_3	Turbulent constant	1.4b
$C_{2\varepsilon}$	Turbulent constant	1.4b
G	Gravity	1.4d
C_μ	Turbulent constant	1.4f
R	Radiation flux	1.5.1a
A	Absorption coefficient	1.5.1a
E	Emmisivity	1.5.1a
S	Scatter co-efficient	1.5.1a
I,J,K,L,M,N	Six-flux radiation directional fluxes	1.5.2a
F	Scalar quantity	1.6
F	Mixture fraction	1.7b
f_s	Stoichiometric value of mixture fraction	1.7b
m_f	Fuel mass fraction	1.7b
m_a	Aire mass fraction	1.7b
m_p	Product mass fraction	1.7b
V_p, V_p^0	Volume of cell at current/previous time	2.1.1
N	Normal component	2.1.2a
A_f	Face area	2.1.2a
d_{AP}	Distance between A and P cell nodes	2.1.3a
F_f	Strength of convection	3b
D_f	Strength of diffusion	3b
a_p	Pth cell coefficient	3c

a_{nb}	Neighboring cell coefficients	3c
e,w,n,s	East, West, North and South neighbors	4.1.1a
ρ	Density	1.1
μ_{eff}	Effective viscosity	1.2
ε	Dissipation rate	1.4a
μ_{lam}	Laminar viscosity	1.4a
υ_t	Turbulent kinematic viscosity	1.4a
σ_k	Turbulent Prandtl number for *k*	1.4a
σ_ε	Turbulent Prandtl number for ε	1.4b
β	Expansion coefficient	1.4d
σ	Stefan-Boltzmann constant	1.5.1b
Γ_I	Ith variable diffusion coefficient	1.6
ϕ	Dependent variable	2.0
ϕ_p	Pth cell variables	3c
ϕ_{nb}	Neighboring cell variables	3c
ϕ^T	Transpose of vector	4.1.2b
Δy_p	Cell center to wall distance	5.2.3
τ_w	Wall shear	5.2.4a
φ	Dependent variable	5.2.5

B.1 *BASIC EQUATIONS USED IN SMARTFIRE*

The equations representing the conservation of mass, momentum and energy for transient flows in Cartesian co-ordinates assume the following form:

B.1.1 Mass Conservation

For any flow situation, the flow field should satisfy the mass continuity equation given by:

$$\frac{\partial \rho}{\partial t} + div(\rho \underline{u}) = 0 \tag{1.1}$$

B.1.2 Momentum Conservation

The conservation of momentum in the three co-ordinate directions is given by the equation:

$$\frac{\partial(\rho u_i)}{\partial t} + div(\rho \underline{u} u_i) = -\frac{\partial P}{\partial x_i} + div(\mu_{eff}\, grad u_i) + S_{u_i} \tag{1.2}$$

where u$_i$ is the velocity in the x, y and z directions and P is the pressure.

B.1.3 Energy Conservation

For energy conservation, we solve the enthalpy form of the equation given by:

$$\frac{\partial(\rho h)}{\partial t} + div(\rho \underline{u} h) = div\left\{\left(\frac{k}{c_p} + \frac{\rho \upsilon_t}{\sigma_T}\right) grad(h)\right\} + S_h \tag{1.3a}$$

where the temperature is evaluated from the expression

$$T = \frac{h}{c_p} \tag{1.3b}$$

B.1.4 Turbulence Model

The buoyancy modified two-equation (k-ε) turbulence model represents turbulent flow. The model consists of the turbulent kinetic energy equation

$$\frac{\partial k}{\partial t} + div(\rho \underline{u} k) = div\left(\left[\mu_{lam} + \frac{\rho v_t}{\sigma_k}\right] grad k\right) + P + G - \rho\varepsilon \tag{1.4a}$$

and the dissipation rate equation

$$\frac{\partial \varepsilon}{\partial t} + div(\rho \underline{u} \varepsilon) = div\left(\left[\mu_{lam} + \frac{\rho v_t}{\sigma_\varepsilon}\right] grad\,\varepsilon\right) + \frac{\varepsilon}{k}\left[C_{1\varepsilon}\left(P + C_3 \max(G,0)\right) - C_{2\varepsilon}\rho\varepsilon\right] \qquad (1.4b)$$

where P represents the turbulent production rate

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

(1.4c)

and G represents the Buoyancy term, given by

$$G = -\beta g \rho v_t \frac{\partial T}{\partial y} \quad or \quad G = g v_t \frac{\partial \rho}{\partial y} \qquad (1.4d)$$

and

$$\beta = -\frac{1}{\rho}\frac{\partial \rho}{\partial T} \qquad (1.4e)$$

The apparent turbulent viscosity is evaluated by using the expression

$$\upsilon_t = C_\mu \frac{k^2}{\varepsilon} \qquad (1.4f)$$

The turbulence model contains five constants that are adjustable. The standard k-ε turbulence model employs the first five values for the constants given in the table below. The final value is for the buoyancy correction applied to the standard turbulence model.

C_μ	σ_k	σ_ε	$C_{1\varepsilon}$	$C_{2\varepsilon}$	C_3
0.09	1.0	1.22	1.44	1.92	1.0

B.1.5 Radiation Models

It is essential that when simulating fires, we represent adequately the characteristics of heat transfer and energy balance in the model. Within the fire model there are two primary modes of heat transfer, namely convection and radiation. While convective heat transfer is accounted for by the transport equations, radiative heat transfer requires a separate sub-model. Within ***SMARTFIRE***, three radiation models are provided. These are (a) the Radiosity model, (b) the Six-Flux Radiation model and (c) the Multiple Ray Radiation model. The Radiosity model is simple in nature and involves the solution of a single extra variable that is the radiant potential within each cell. While this ensures that the model is efficient in terms of CPU time it is a crude representation of radiation. The modified Six-Flux radiation model solves for six equations, one in each co-ordinate direction (both positive and negative directions) and makes the model more accurate but less efficient in terms of CPU time, when compared with the Radiosity model. The Multiple Ray radiation model solves for ray intensity in a number of supplied ray directions. This model is potentially very costly when using a large number of rays but benefits when good choices of ray direction helps to spread heat in more realistic ways that either the Radiosity or Six Flux radiation models. These models are presented below:

B.1.5.1 Radiosity Model

The equation for the Radiosity, R, takes the form

$$\frac{d}{dx_i}\left[\frac{4}{3(\alpha+s)}\frac{dR}{dx_i}\right]+\alpha\ (E-R)=0 \qquad (1.5.1a)$$

where α is the absorption coefficient, *s* is the scattering coefficient and *E* is the black body emissive power of the fluid calculated using

$$E=\sigma T^4 \qquad (1.5.1b)$$

where *T* is the temperature of the fluid and σ is the Stefan-Boltzmann constant.

Transfer of heat through radiation leads to a source in the enthalpy equation equal to the negative of the source in the Radiosity equation, given below:

$$S_{radiosity}=-\alpha\ (E-R) \qquad (1.5.1c)$$

B.1.5.2 Six-Flux Radiation Model

In the six-flux radiation model heat fluxes R_i , are calculated by solving additional conservation equations in each component direction which have the form:

$$\left.\begin{aligned}
\frac{dI}{dx} &= -(\alpha + s)I + \alpha E + \frac{s}{6}(I + J + K + L + M + N) \\
\frac{dJ}{dx} &= +(\alpha + s)J - \alpha E - \frac{s}{6}(I + J + K + L + M + N) \\
\frac{dK}{dy} &= -(\alpha + s)K + \alpha E + \frac{s}{6}(I + J + K + L + M + N) \\
\frac{dL}{dy} &= +(\alpha + s)L - \alpha E - \frac{s}{6}(I + J + K + L + M + N) \\
\frac{dM}{dz} &= -(\alpha + s)M + \alpha E + \frac{s}{6}(I + J + K + L + M + N) \\
\frac{dN}{dz} &= +(\alpha + s)N - \alpha E - \frac{s}{6}(I + J + K + L + M + N)
\end{aligned}\right\} \qquad (1.5.2a)$$

where α is the absorption coefficient, s is the scattering coefficient E is the black body emissive power of the fluid and I, J, K, L, M and N the six coordinate direction radiative fluxes.

Transfer of heat through radiation leads to a source in the enthalpy equation given by:

$$S_{six-flux} = \alpha\ \left((I - E) + (K - E) + (M - E) + (J - E) + (L - E) + (N - E)\right) \qquad (1.5.2b)$$

B.1.5.3 Absorption Coefficient

The absorption coefficient is evaluated using the following piecewise linear approximation:

T < 50 °C	$\forall = \forall_{ambient}$
T > 50 °C and T < (T_{plume}/2)	$\forall = \forall_{ambient} + (c(T_{plume}/2) - \forall_{ambient})/((T_{plume}/2)-50)(T-50)$
T > (T_{plume}/2)	$\forall = cT$

(1.5.4)

B.1.6 Species Conservation

The conservation of any scalar quantity, f, is represented by the equation given below:

$$\frac{\partial(\rho f)}{\partial t} + div(\rho \underline{u} f) = div\left(\Gamma_f\, grad(f)\right) + S_f \qquad (1.6)$$

B.1.7 Auxiliary Equations

The following support equations are used to calculate essential calculated variables.

B.1.7.1 Density

The density is calculated using the Ideal gas law, given by

$$\rho = PW / RT$$

where P is pressure, W is molecular weight, R is Universal gas constant and T is temperature.

B.1.7.2 Buoyancy

The following "BOUSSINESQ" equation is used to calculate the Buoyancy source term in each control volume:

$$B = - \alpha (T - T_{ref}) \rho_{ref} V g$$

where	B	is the Buoyancy source term,
	α	is the thermal expansion coefficient,
	T	is the control volume Temperature,
	T_{ref}	is the reference Temperature,
	ρ_{ref}	is the reference Density,
	V	is the Volume
and	g	is the acceleration due to gravity.

Conversely the Non-Boussinesq approximation is formulated as follows for compressible (IDEAL GAS LAW) fluids:

$$B = - (\rho_{ref} - rho) V g$$

where	B	is the Buoyancy source term,
	ρ_{ref}	is the reference Density,
	ρ	is the current Density,
	V	is the Volume
and	g	is the acceleration due to gravity.

B.2 *GENERAL SCALAR EQUATION*

From the brief introduction of the equations used to represent complex fluid flow, it is clear that all the equations can be cast into a generalised form given by

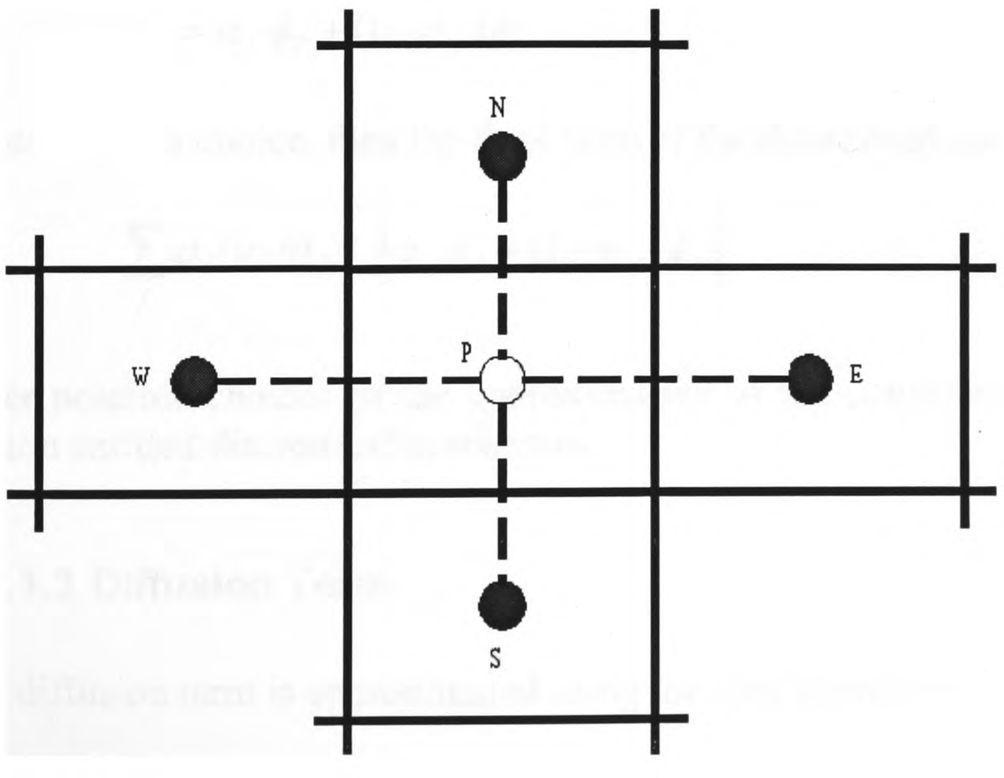
$$\underbrace{\frac{\partial(\rho\phi)}{\partial t}}_{Transient} + \underbrace{div(\rho \underline{u}\, \phi)}_{Convection} = \underbrace{div(\Gamma_\phi grad(\phi))}_{Diffusion} + \underbrace{S_\phi}_{Source} \qquad (2.0)$$

where ρ is density, $\underline{\boldsymbol{u}}$ is the vector velocity, Γ is the diffusion coefficient for the quantity ϕ and $\boldsymbol{S}$ is the source term for ϕ at any point. The four terms in the equation are the transient term, the convection term, the diffusion term and the source term. This equation is commonly known as the convection-diffusion form of transport equations.

B.2.1 Approximations of the Terms

All the terms in the convection-diffusion equation need to be approximated in order for the equation to be solved. In what follows there are no details as to how this process is achieved, however the final forms of the approximations are presented for completeness. The steps are

(1) Discretise the flow domain into a collection of control volumes,
(2) Integrate the individual terms over the surface or volume of each control volume,
(3) Approximate first derivatives using upwind values,
(4) Approximate face values for dependent quantities using sensible averaging approaches,
(5)`Finally construct the full system of algebraic equations to solve.



The previously mentioned approximations are based on a typical control volume arrangement as depicted above. This is a typical 2D Control Volume set-up. The computational molecule shows the influence of neighbouring control volumes (North, South, West and East) on the control volume of interest (labelled P). It should be noted that ***SMARTFIRE*** actually uses a 3D unstructured mesh but similar concepts apply.

B.2.1.1 Transient Term

The transient term is approximated using the backward difference technique. This gives

$$\int_{t-\Delta t}^{t}\int_V \frac{\partial(\rho\phi)}{\partial t}dV\,dt \cong \frac{V_P\rho_P\phi_P - V_P^0\rho_P^0\phi_P^0}{\Delta t} \tag{2.1.1}$$

B.2.1.2 Convection Term

The convection term is the most important term in the equation. Care must be taken in approximating this term, as inappropriate approximations can lead to large errors or instability problems if not handled with care. This gives

$$\int_V div(\rho\underline{u}\phi)dV = \int_S \rho(\underline{u}\cdot\underline{n})\phi dS \;\cong\; \sum_f \rho_f(\underline{u}\cdot\underline{n})_f A_f\phi_f$$

(2.1.2a)

In this equation the value of ρ_f is given the value in the upwind element. Thus

$$\rho_f = \rho_P \quad if \quad (\underline{u}\cdot\underline{n})_f > 0.0 \quad and \quad \rho_f = \rho_A \quad if \quad (\underline{u}\cdot\underline{n})_f < 0.0 \tag{2.1.2b}$$

The convected quantity, ϕ, at the face needs further approximation. One possible approach is to use arithmetic averaging, e.g.

$$\phi_f = \alpha_f\phi_P + (1-\alpha_f)\phi_A \tag{2.1.2c}$$

Assuming this choice, then the final form of the discretised convection term becomes

$$\sum_f \rho_f(\underline{u}\cdot\underline{n})_f A_f\left[\alpha_f\phi_P + (1-\alpha_f)\phi_A\right] \tag{2.1.2d}$$

Other possible choices of the approximation of the convected, ϕ, are presented in the section entitled discretisation schemes.

B.2.1.3 Diffusion Term

The diffusion term is approximated using the approximation:

$$\int_S div(\Gamma_\phi grad(\phi))dV = \int_S \Gamma_\phi grad(\phi)\cdot\underline{n}dS \;\cong\; \sum_f (\Gamma_\phi)_f A_f\left(\frac{\phi_A-\phi_P}{d_{AP}}\right) \tag{2.1.3a}$$

where the diffusion coefficient is approximated by

$$(\Gamma_\phi)_f = \frac{(\Gamma_\phi)_A(\Gamma_\phi)_P}{\alpha_f(\Gamma_\phi)_P + (1-\alpha_f)(\Gamma_\phi)_A} \qquad (2.1.3b)$$

B.2.1.4 Source Term

In general, since the source term is a function of the dependent variable, ϕ, a linearized form is used in the final discretised equation to ensure diagonal dominance of the system matrix. This form is as given below:

$$S_\phi = S_C - S_P\phi = V_P(S_C - S_P\phi_P) \qquad (2.1.4)$$

B.2.1.5 Turbulence Source Term Linearisation

The turbulence sources are further modified to give better handling and ensure diagonal dominance. Currently there are three distinct methods for turbulence source linearisation for the dissipation rate and kinetic energy equations. Assuming that the linearisation can be written in the form:

S_C - S_L N

then the following gives the formulae for the coefficients for each of the methods:

B.2.1.5.1 Method 1

k source terms :- $S_C = \nu_t \rho G$
$S_L = C_D \nu_t \rho / (C_\mu l^2)$

$\in$ source terms :- $S_C = C_D \rho \nu_t C_{1\in} G \nu_t / (C_\mu l^2)$
$S_L = C_{2\in} C_D \rho \nu_t / (C_\mu l^2)$

B.2.1.5.2 Method 2

k source terms :- $S_C = (C_{2\in} - 1.0) C_\mu C_D k^2 \rho / \nu_t + 1.5 \nu_t \rho G$
$S_L = \nu_t \rho G 0.5 / k + C_{2\in} C_\mu C_D k \rho / \nu_t$

$\in$ source terms :- $S_C = (C_\mu C_D \rho k / \nu_t) (C_{1\in} \nu_t G + (C_{2\in} - 1.0) \in)$
$S_L = (2.0 C_{2\in} - 1.0) C_\mu C_D \rho k / \nu_t$

B.2.1.5.3 Method 3

k source terms :- $S_C = \nu_t \rho G$

$$S_L = C_D\, C_\mu\, k\, \rho\, /\, \nu_t$$

$\in$ source terms :-

$$S_C = C_D\, \rho\, k\, C_{1\in}\, G\, C_\mu$$
$$S_L = C_{2\in}\, C_D\, \rho\, k\, C_\mu\, /\, \nu_t$$

where	**G**	is the magnitude of the rate of strain,
	ρ	is the density,
	ν_t	is the turbulent viscosity,
	k	is the kinetic energy,
	$\in$	is the dissipation rate,
	l	is the mixing length

and $\mathbf{C_D}$, $\mathbf{C_\mu}$, $\mathbf{C_{1\in}}$ and $\mathbf{C_{2\in}}$ are constants, taking the values 0.1643, 0.09, 1.44 and 1.92 respectively.

B.3 *OVERALL DISCRETISATION EQUATION*

Having obtained expressions for the discretised form of each of the terms in the conservation equation in the previous section, the discretised form of the full equation is obtained by simply adding together all these contributions. Assuming arithmetic averaging for the evaluation of the face value of ϕ in the convection term, the discretised equation becomes

$$\frac{(\rho_P\phi_P V_P - \rho_P^0\phi_P^0 V_P^0)}{\Delta t} + \sum_f \left[\rho_f(\underline{u}\cdot\underline{n})_f \{\alpha_f\phi_P + (1-\alpha_f)\phi_A\} A_f - (\Gamma_\phi)_f \left(\frac{\phi_A - \phi_P}{d_{AP}}\right) A_f \right] = (S_C - S_P\phi_P)V_P \tag{3a}$$

Defining the quantities F_f and D_f as

$$\left.\begin{aligned} F_f &= A_f\rho_f(\underline{u}\cdot\underline{n})_f \\ D_f &= A_f(\Gamma_\phi)_f / d_{AP} \end{aligned}\right\} \tag{3b}$$

where F_f is the strength of the convection of ϕ and D_f is the diffusion conductance, the equation can be further be simplified and written in the following form:

$$a_P\phi_P = \sum_{nb} a_{nb}\phi_{nb} + b \tag{3c}$$

where the summation is over all neighboring elements and the equations for the coefficients, a_p and a_{nb} in the above equation are given by

$$\left.\begin{aligned} a_{nb} &= D_f - (1-\alpha_f)F_f \\ a_P &= \sum_f (D_f + \alpha_f F_f) + a_P^o - S_P V_P \\ a_P^o &= \frac{\rho_P^0\phi_P^0 V_P^0}{\Delta t} \\ b &= S_C V_P + a_P^o\phi_P^0 \end{aligned}\right\} \tag{3d}$$

The convected quantity of the dependent variable, ϕ, can be represented in a number of ways. The most commonly used technique is to use the upwind value. This is known as the upwind scheme, where the face value is approximated by the following rule:

$$\phi_f = \phi_P \quad if \quad F_f > 0.0 \quad and \quad \phi_f = \phi_A \quad if \quad F_f < 0.0 \tag{3e}$$

In which case the coefficients become

$$\left.\begin{aligned} a_{nb} &= D_f + \max(-F_f, 0.0) \\ a_P &= \sum_f \left[D_f + \max(F_f, 0.0)\right] + a_P^o - S_P V \\ a_P^o &= \frac{\rho_P^0 \phi_P^0 V_P^0}{\Delta t} \\ b &= S_C V_P + a_P^o \phi_P^0 \end{aligned}\right\} \qquad (3f)$$

In general a wide choice is available for the evaluation of the convected face value of ϕ. To incorporate a generalized version of the final discretised equation, we introduce a function A(|P|) which allows for any differencing scheme to formulated and incorporated, where P is the Peclet number given by F_f / D_f. This gives

$$\frac{\rho_P \phi_P V_P - \rho_P^0 \phi_P^0 V_P^0}{\Delta t} + \sum_f \left[\left\{D_f A(|P_f|) + \max(-F_f, 0.0)\right\}(\phi_P - \phi_A) + F_f \phi_P\right] = (S_C - S_P \phi_P) V_P \qquad (3g)$$

where the expression for A(|P|), for various schemes are given in the Table below.

Scheme	**Formulae for** $A(\|P\|)$
Central Differencing	$1 - 0.5\ \|P\|$
Upwind	1
Hybrid	$\mathrm{Max}\,(0, 1 - 0.5\ \|P\|)$
Power Law	$\mathrm{Max}\,(0, (1 - 0.1\ \|P\|)^5)$
Exponential	$\|P\|\ /\ \exp(\|P\|) - 1$

In which case the coefficients become

$$\left.\begin{aligned} a_{nb} &= D_f A(|P|) + \max(-F_f, 0.0) \\ a_P &= \sum_f \left[D_f A(|P|) + \max(F_f, 0.0)\right] + a_P^o - S_P V \\ a_P^o &= \frac{\rho_P^0 \phi_P^0 V_P^0}{\Delta t} \\ b &= S_C V_P + a_P^o \phi_P^0 \end{aligned}\right\} \qquad (3h)$$

<u>Note</u>: The relative merits of the various differencing schemes are not discussed here.

B.4 *ALGEBRAIC EQUATIONS*

The algebraic equations, as described in the previous section, need to be solved using appropriate methods. The choice of solution techniques used will effect both the accuracy of the solution and the effort required obtaining the solution. Thus it is important that several solution techniques are available for solving a multitude of different problems. The next section briefly describes the solution techniques used within *SMARTFIRE*.

B.4.1 Solution of the Algebraic Equations

The starting point for the solution of any equation is the set of algebraic equations. In this case we use the following representation:

$$Ax = b \tag{4.1}$$

where A is a matrix of n x m elements and x and b are vectors of n elements.

The finite-volume discretisation approach results in a set of algebraic equations, which when represented in matrix form generates quite a large system matrix. Due to the nature of the stencil used, the system matrix although large, is quite sparse. Since the resulting algebraic equations are not linear in nature as the coefficients are themselves functions of other dependent variables, it is prudent to use iterative solvers to attain solutions efficiently.

A number of solution techniques are available in ***SMARTFIRE*** namely, the Jacobi Over Relaxation (JOR) method, the Successive Over Relaxation (SOR) method, Conjugate Gradient Method (CGM), Bi-Conjugate Gradient method (BiCG) and the Whole Field Solver. The two most frequently used solution techniques for point by point linear equations are the JOR method and the SOR method. Each of these techniques are briefly described below.

B.4.1.1 JOR / SOR SOLVER

For the JOR technique a typical brick-shaped cell is updated as follows:

$$\phi_{p,JOR} = \frac{\left(a_e\phi_{e,old} + a_w\phi_{w,old} + a_n\phi_{n,old} + a_s\phi_{s,old} + a_h\phi_{h,old} + a_l\phi_{l,old} + b\right)}{a_p} \tag{4.1.1a}$$

whereas for the SOR techniques a typical brick-shaped cell is updated as follows:

$$\phi_{p,SOR} = \frac{\left(a_e\phi_{e,new} + a_w\phi_{w,new} + a_n\phi_{n,new} + a_s\phi_{s,new} + a_h\phi_{h,new} + a_l\phi_{l,new} + b\right)}{a_p} \tag{4.1.1b}$$

The final cell value is then updated, using linear relaxation as follows:

$$\phi_{p,new} = relax * (\phi_{p,old} - \phi_{p,(JOR\ or\ SOR)}) + \phi_{p,(JOR\ or\ SOR)} \qquad (4.1.1c)$$

B.4.1.2 WHOLE FIELD SOLVER

The whole field solver, as the name suggests aims at solving the equations at the end of the sweep, when all the nodal points have been visited and the system matrix build up. The whole field solver differs from a point wise solver (for example JOR, SOR) by using extra inner loops and back-substitution to ensure that the updated solution influences every other control volume.

B.4.2 SIMPLE Solution Procedure

The fire model comprises a set of highly non-linear and tightly coupled equations. When solving such a set of equations, the order and manner in which the solution progresses is vital. Several solution procedures can be used to solve the equations. In ***SMARTFIRE*** the SIMPLE solution procedure is used to solve the equation set. The procedure is outlined below:

Step 1: Guess the initial pressure field p*

Step 2: Solve momentum equations with guessed pressure field to obtain u*, v*, w*

Step 3: Solve for the pressure correction p'

Step 4: Calculate the new pressure field using p* and p'

Step 5: Calculate the new velocity components u, v and w using u*, v*, w* and p'

Step 6: Solve the other conserved quantities i.e. Enthalpy, Temperature, Turbulence, Concentration, Radiation, density, viscosity, etc.

Step 7: Treat the corrected pressure p as p* and return to step 2.

Step 8: Repeat Steps 2 to 7 until the solution has converged

Step 9: Repeat Steps 2 to 8 for the next time step

B.4.2.1 Dependent Variable Storage Considerations

Although the staggered grid storage arrangement, where the velocity components in each coordinate direction are stored at the cell-face, has been the most widely used technique for pressure based solution schemes, it has been recognized that the storage requirements for such schemes is very large. In ***SMARTFIRE***, a collocated grid arrangement, where velocity components in each coordinate direction are stored at the cell-center, is used. The consequences of such an approach are:

Huge reduction in storage requirements for geometrical related quantities, and
The prediction of undesirable "checker-board" pressure fields.

To alleviate the prediction of the checker-board pressure fields, ***SMARTFIRE*** uses the Rhie and Chow technique to predict the flux at the cell-faces, where they are needed, by means of an algorithm that is free from the checker-board oscillations. Thus the face velocity depends on the pressure values prevailing at the cell centers of the neighboring cells (without using interpolation), and interpolated values of the other quantities used within the momentum equation. This approach is similar to the staggered approach, where the fluxes at the faces are identical for both the non-staggered and staggered approach.

B.4.3 Convergence and Relaxation Methods

Since the equations of fluid flow are coupled and non-linear, it is important that when using iterative solution techniques, as presented in the previous section, those relaxation techniques are used to control the solution. Relaxation techniques aid the solution to converge. Within ***SMARTFIRE***, there are three relaxation techniques available. These are the solver relaxation, linear relaxation and false time step relaxation techniques. Details of these techniques are presented in the next two sub-sections.

B.4.3.1 Linear relaxation

The linear relaxation technique allows the variation of a solved for variable, ϕ, in a linear fashion. The amount by which the variable is allowed to vary is controlled by the expression

$$\phi_{updated} = \alpha\phi_{calculated} + (1-\alpha)\phi_{old} \qquad (4.3.1)$$

where α takes values between 0 and 1.7. The terms over relaxation and under relaxation are defined in the range of α as presented in the table below.

$\alpha < 1.0$	Under relaxation
$\alpha=1.0$	No relaxation
$\alpha>1.0$	Over relaxation

This technique can be applied for any variable that needs updating both within the solver and externally.

B.4.3.2 False time step relaxation

Using the concept of inertial relaxation, otherwise known as false time step relaxation, an equation of the form:

$$\phi_{updated} = \frac{\sum_{nb} a_{nb}\phi_{nb} + b + F_t\phi_{previous}}{(a_P + F_t)} \qquad (4.3.2)$$

is produced, where F_t is termed the false time step, the units of which are the same as those of the a_P coefficient, i.e. kg/s. Thus the larger F_t is the stronger the relaxation. This technique is normally used for both steady and transient simulations.

B.4.4 Residual Calculation Methods

All solvers need termination criteria. In the case of ***SMARTFIRE***, several options are available at several stages within the solution stage. Various forms are presented below.

B.4.4.1 Solver residual

This is the maximum error term evaluated from substituting the newest x solution vector into the system of algebraic linear equations and evaluating the maximum difference between the left and right hand sides of the equation. i.e.

$$\underline{r} = \underline{\underline{A}x_{p,new}} - \underline{b} \qquad (4.4.1)$$

The solver residual is used to determine if convergence has been reached and hence the solver inner iterations can cease.

B.4.4.2 Variable residual

The variable residual is a measure of the solution error between solution sweeps and is used to check for convergence for individual variables, using an expression of the form:

$$\phi_{p,new} - \phi_{old} = r \qquad (4.4.2)$$

A variety of named methods for calculating the variable residual are available as listed below.

1	ABSOLUTE L1 NORM
2	ABSOLUTE L2 NORM

3	ABSOLUTE LI NORM
4	RELATIVE L1 NORM
5	RELATIVE L2 NORM
6	RELATIVE LI NORM
7	REFERENCE L1 NORM
8	REFERENCE L2 NORM
9	REFERENCE LI NORM

The norms have the following evaluation methods:

B.4.4.2.1 Absolute - L_1 norm

$$\|\Phi\|_1 = \sum |\Phi_{new} - \Phi_{last}|$$

B.4.4.2.2 Absolute - L_2 norm

$$\|\Phi\|_2 = \sqrt{\sum (\Phi_{new} - \Phi_{last})^2}$$

B.4.4.2.3 Absolute - L_∞ norm

$$\|\Phi\|_\infty = MAX\left(|\Phi_{new} - \Phi_{last}|\right)$$

B.4.4.2.4 Relative norms

The **RELATIVE** norms use, essentially, the same definitions as the above except that the $(\Phi_{new} - \Phi_{last})$ is replaced by the term $((\Phi_{new} - \Phi_{last}) / \Phi_{new})$. It is not recommended that the **RELATIVE** norms are used because division by Φ_{new} can overflow where Φ is very small or zero.

B.4.4.2.5 Reference norms

The **REFERENCE** norms allow a user to choose a suitable reference value for the calculation of the norms. The calculation method employed for calculation of reference norms is essentially the same as for absolute norms except that the term $(\Phi_{new} - \Phi_{last})$ is replaced by the term $((\Phi_{new} - \Phi_{last}) / \Phi_{reference})$. Clearly the value of $\Phi_{reference}$ should be suitably large to prevent overflow caused by division by zero or near zero. Ideally the reference norm should use the maximum expected value of Φ for the whole simulation. This will tend to normalise the residuals to variable independent values, which can be more readily compared between other variables and other simulations.

B.5 *BOUNDARY CONDITIONS*

To complete the definition of the fire problem it is necessary to specify a set of boundary and initial conditions. This sections deals with the boundary conditions that are available within ***SMARTFIRE***. It is very important that the user specifies the boundary conditions correctly and also understands its impact within the numerical solution procedure. The section details the boundary conditions related to each equation solved in terms of the names as used within the ***SMARTFIRE*** User Interface.

B.5.1 Pressure Equation

B.5.1.1 Outlet or Free boundary

$$P = P_{\text{external}} \quad \text{or} \quad P = P_{\text{fixed}} \tag{5.1.1}$$

B.5.2 Momentum Equation

B.5.2.1 Inlet

Prescribe u, v, w as V̲ (5.2.1)

B.5.2.2 Outlets

Calculated u, v and w in internal cell (5.2.2)

B.5.2.3 Walls for laminar flows

$$F_{shear} = area * \mu\rho \frac{1}{\Delta y_p} u \tag{5.2.3}$$

B.5.2.4 Wall Functions for turbulent flows

Using the widely used log-law for the wall, and the y^+ ($= \frac{\Delta y_p}{\nu}\sqrt{\frac{\tau_w}{\rho}}$) limits, the shear force is represented using the expression

$$F_{shear} = -\tau_w * area \quad = \quad -\mu \frac{u_p}{\Delta y_p} * area \tag{5.2.4a}$$

This expression is normally modified depending on the flow regime it is applied in. For turbulent flows

$$F_{shear} = -\rho C_\mu^{1/4} k_p^{1/2} \frac{u_p}{u^+} * area \qquad (5.2.4b)$$

B.5.2.5 Symmetry

$$\frac{\partial \varphi}{\partial n} = 0 \qquad (5.2.5)$$

B.5.3 Energy Equation

B.5.3.1 Fire as Enthalpy volumetric source

S_H = Fixed value (or) $S_H = (A+Bt+Ct^2+De^{Et})$ (5.3.1)

Where A, B, C, D and E are user defined constants.

B.5.3.2 Walls

The general description of the wall boundary condition for enthalpy is

$$-\lambda_w \partial T/\partial n \,|_{w} = H_c(T_w - T_{gas}) + \varepsilon\sigma T_w^4 - \varepsilon \dot{q}_r \qquad (5.3.2.0)$$

where λ_w is the conductivity of the wall material, T_w is the wall surface temperature, T_{gas} is the air temperature next to the wall, H_c is the convective heat transfer coefficient, ε is the wall emissivity and $\dot{q}_r$ is the radiative heat flux at the wall surface.

B.5.3.2.1 Adiabatic

$\lambda_w = 0$ in (5.3.2.0) (5.3.2.1a)

$\frac{\partial H}{\partial n} = 0$ if radiation is taken into account, (5.3.2.1b)

B.5.3.2.2 Fixed Temperature (Dirichlet)

T_w = prescribed value, (5.3.2.2)

B.5.3.2.3 Fixed Flux (Neumann)

$$\frac{\partial H}{\partial n} = \textit{fixed value} \qquad (5.3.2.3)$$

B.5.3.2.4 Flux / Temperature (Convective)

$$\frac{\partial H}{\partial n} = H_c(T_w - T_{gas}) \qquad (5.3.2.4)$$

where H_c and T_w are specified by users.

B.5.3.2.5 Flux / Temperature / Materials (Conductive)

H_c (prescribed) is used (with wall material properties) to calculate an estimated wall T in terms of (5.3.2.0) that is used instead of prescribed Tw to then find the heat flux (5.3.2.5)

B.5.3.2.6 Turbulent Wall Layer (Calculated Flux)

(1) Estimate a y+ distance using the turbulent wall layer function.

(2) Use the y+ value to calculate an H_C value based on the wall function, i.e.

$$\frac{\partial \mathrm{H}}{\partial \mathrm{n}} = \mathrm{H}_c(T_w - T_g) = (\rho C_\mu^{1/4} k_p^{1/2} C_p)(T_w - T_g)/T^+ \qquad (5.3.2.6)$$

(3) Use the calculated H_C value in the same way as 5.3.2.5

B.5.4 Turbulence Equations

B.5.4.1 Kinetic Energy

Wall: k is obtained by solving the discretised governing equation (5.4.1)

B.5.4.2 Dissipation Rate

$$\text{Wall:} \quad \varepsilon = \frac{0.1643}{K} \frac{k^{1.5}}{y} \qquad (5.4.2)$$

B.5.4.3 Log-Law

$$y^+ = \frac{1}{\kappa}\ln(Ey^+) \qquad (5.4.3)$$

B.5.5 Radiation Equation

B.5.5.1 Solid Surface

$$\left.\begin{array}{ll} I = \varepsilon_w E_w + (1-\varepsilon_w)J & \textit{for lower surface} \\ J = \varepsilon_w E_w + (1-\varepsilon_w)I & \textit{for higher surface} \end{array}\right\} \qquad (5.5.1)$$

Where E_w is calculated from T_w (which uses the following conditions)

$T_w = T_{solid}$ for solid material adjacency
$T_w = T_{calculated}$ for adiabatic, CONDUCTIVE or TURBULENT heat boundaries

B.5.5.2 Free Space

$$\left.\begin{array}{ll} I = E_w & \textit{for lower surface} \\ J = E_w & \textit{for higher surface} \end{array}\right\} \qquad (5.5.2)$$

B.5.5.3 Fixed Temperature

Same as 1.5.5.1 But E_w is calculated from a prescribed T_w as

$$E_w = \sigma T_w^{\,4} \qquad (5.5.3)$$

B.5.6 Concentration Equation

B.5.6.1 Fixed Value

$$\phi = value \qquad (5.6.1)$$

B.5.6.2 Fixed Flux (Neumann)

$$\frac{\partial\phi}{\partial n} = value \qquad (5.6.2)$$

B.5.6.3 Linear Flux

$$\frac{\partial \phi}{\partial n} = flux\_coeff * (\phi_{Ambient} - \phi_P) \qquad (5.6.3)$$

B.5.6.4 Symmetry Plane

$$\frac{\partial \phi}{\partial n} = 0 \qquad (5.6.4)$$

B.6 *SMARTFIRE VARIABLE NAMES*

The available variable names currently used within *SMARTFIRE* and will be displayed in the various graphical user interface windows and in result files. The names are also recognised by the CFD engine command script parser. The following variables are used for the indicated problem types:

Solved variables	Models	Units
PRESSURE	flow	Pa
U-VELOCITY	flow	m s^{-1}
V-VELOCITY	flow	m s^{-1}
W-VELOCITY	flow	m s^{-1}
ENTHALPY	heat	J
KINETIC ENERGY	flow + turbulence	m^2 s^{-2}
DISSIPATION RATE	flow + turbulence	m^2 s^{-3}
RADIOSITY	radiosity radiation	W m^2
RADIATION X NEG	six-flux radiation	W m^2
RADIATION X POS	six-flux radiation	W m^2
RADIATION Y NEG	six-flux radiation	W m^2
RADIATION Y POS	six-flux radiation	W m^2
RADIATION Z NEG	six-flux radiation	W m^2
RADIATION Z POS	six-flux radiation	W m^2
RAY INTENSITY	multiple ray radiation	W m^2
MIXTURE FRACTION	all combustion models	-
FUEL	mixing controlled combustion	-
SMOKE	smoke	-
Calculated variables	**Models**	**Units**
TEMPERATURE	all models	K
BUOYANCY	flow + heat	kg m s^{-2}
DENSITY	compressible flow	kg m^{-3}
ABSORPTION COEFF	all radiation models	m^{-1}
RADIATION SUM	multiple ray radiation	W m^2
FUEL	diffusion controlled combustion	-
OXIDANT	all combustion models	-
PRODUCT	all combustion models	-

Additionally there are a number of support variables, as follows:

Material properties	**Models**	**Units**
DENSITY	non-compressible flow models	kg m^{-3}
SPECIFIC HEAT	all models	J kg^{-1} K
LAMINAR VISCOSITY	all models	kg m^{-1} s^{-1}
CONDUCTIVITY	all models	W m^{-1} K^{-1}
variables	**Models**	**Units**
VELOCITY MAGNITUDE	flow	m s^{-1}
REAL U VELOCITY	flow	m s^{-1}
REAL V VELOCITY	flow	m s^{-1}
REAL W VELOCITY	flow	m s^{-1}

These variables, when used, should be typed exactly as they appear here with upper case characters and single spaces where indicated. The velocity components should be hyphenated rather than using spaces or underscores (although ***SMARTFIRE*** has been developed to be flexible and will generally understand alternative forms).

The distinction between the variables is as follows. **Solved variables** are carried through the domain by transport processes and are thus solved using one of the available solvers after a coefficient matrix has been constructed based on the transport, creation and destruction of the property. There are a large number of controls associated with solved variables. Conversely **Calculated variables** are merely calculated from combinations of other variables and geometric quantities. There are only a few controls associated with calculated variables. **Material properties** are stored on a cell-by-cell basis. Only certain configurations will require these Material properties to be updated because of dependencies on other system variables. Material properties are only set via the INFORM file. There are very few controls associated with them. Lastly, **Monitoring and displayable variables** are maintained automatically to provide more or succinct information that would not be available by accessing the required Solved or Calculated variables. There are no controls associated with such monitor variables.

The basic names that appear above can also have similarly named support variables in certain lists. The solved and calculated variables all have a residual auxiliary variable that is used for visualisation of the residuals of the parent variable. For example the ENTHALPY variable has a partner variable named ENTHALPY_RES that is created automatically by the CFD Engine. ENTHALPY_RES will always contain the cell-by-cell values of enthalpy error residual.

It should be noted that some variables (e.g. DENSITY) are used / calculated differently depending on the type of simulation that is being performed (i.e. the models activated).

B.7 *SMARTFIRE SYSTEM OF UNITS*

Most of the ***SMARTFIRE*** system is actually independent of the type of units system used, as long as all of the values entered are consistent, however there are certain properties and variables that do require a given system of units. To ensure solution consistency, and to enable future developments, the S.I. system of units should be used for all properties, numerical values and geometric properties. This must, therefore, include all of the entries in the INFORM file and the geometry specification file.

The prescribed unit's system is briefly described here to ensure consistency:

Property	**Units**	**Dimensions**
Mass	kilogram (kg)	[M]
Length	metre (m)	[L]
Time	second (s)	[T]
Temperature	degree Kelvin (K)	[K]
Volume	m^3	$[L^3]$
Area	m^2	$[L^2]$
Density	kg m^{-3}	$[ML^{-3}]$
Force	Newton (N) = kg m s^{-2}	$[MLT^{-2}]$
Pressure	Pascal (Pa) = N m^{-2}	$[ML^{-1}T^{-2}]$
Velocity	m s^{-1}	$[LT^{-1}]$
Acceleration	m s^{-2}	$[LT^{-2}]$
Energy	Joule (J) = N m	$[ML^2T^{-2}]$
Power	Watt (W) = J s^{-1}	$[ML^2T^{-3}]$
Dynamic Viscosity	N s m^{-2} = Pa s	$[ML^{-1}T^{-1}]$
Conductivity	W m^{-1} K^{-1}	$[MLT^{-3}K^{-1}]$
Specific Heat Capacity	W kg^{-1} K	$[L^2T^{-3}K]$

Table B-1: System of Units used by *SMARTFIRE*.

Appendix C

Steckler case

Overview

Steckler case is a standard fire test case used by a number of CFD model developers, as there exist very well documented and readily available experimental results. It is therefore often used to as a benchmark case to test the capabilities of the fire models in predicting the temperature and flow distributions in a small compartment with a steady non-spreading fire.

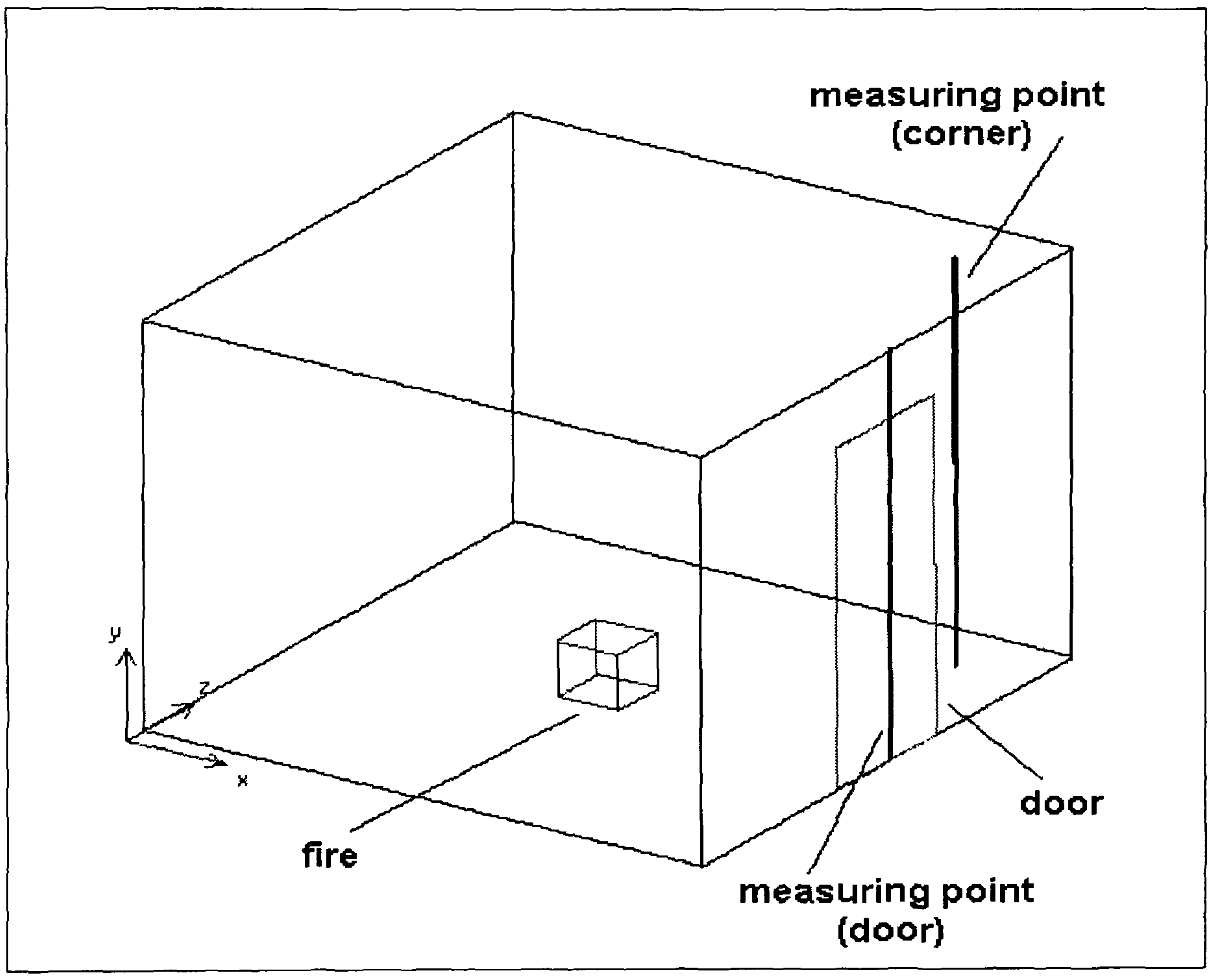


Figure C-1 Steckler case set-up

The Steckler case consists of the following:

- A small compartment measuring 2.8 m x 2.8 m with 2.18 m in height, with a doorway centrally located in one of the walls, measuring 0.74 m wide by 1.83 m high.
- All walls and ceiling were 0.1m thick and covered with ceramic fibre insulation.

- Centrally located methane burner measuring 0.3 m in diameter and producing constant 62.9kW heat output.
- The first set of sensors (temperature and velocity) is situated vertically in the middle of the doorway.
- The second set of sensors (temperature only) is located in a corner close to the doorway, 0.305m from each of the walls.

A CFD model is of course only an approximation of the real-life case. Consequently, this section describes the details of the SMARTFIRE setup used in this thesis to model Steckler case. We will list the mathematical models, describe the grid setup and present a set of final results.

Mathematical models

The following models were used in Steckler simulation:

- Standard CFD models (continuity, momentum and energy conservation laws – i.e. Navier-Stokes equations in the general form of Convection-Diffusion equations).
- Six-flux radiation model
- Buoyancy-modified k-ε turbulence model
- Volumetric heat source (as a substitute for full combustion model)

The following values for major material properties and physical constants were assumed:

- Fully compressible gas – air.
- External pressure: 1.01325e+05 Pa
- Gravity: -9.81 m/s
- Initial temperature: 293.75K
- Initial pressure: 1.01325e+05 Pa
- Initial velocities: 0

Mesh set up

We used a structured mesh, generated automatically with a tool incorporated within SMARTFIRE. The mesh was rather coarse and therefore caused some problems for an automatic mesh generator. The result was not very good (e.g. the heat source became slightly distorted) but sufficient for our purposes and it was used for the accuracy assessment runs. The mesh contained 9860 cells (29 x 17 x 20) and its internal structure is shown in Figure C-2.

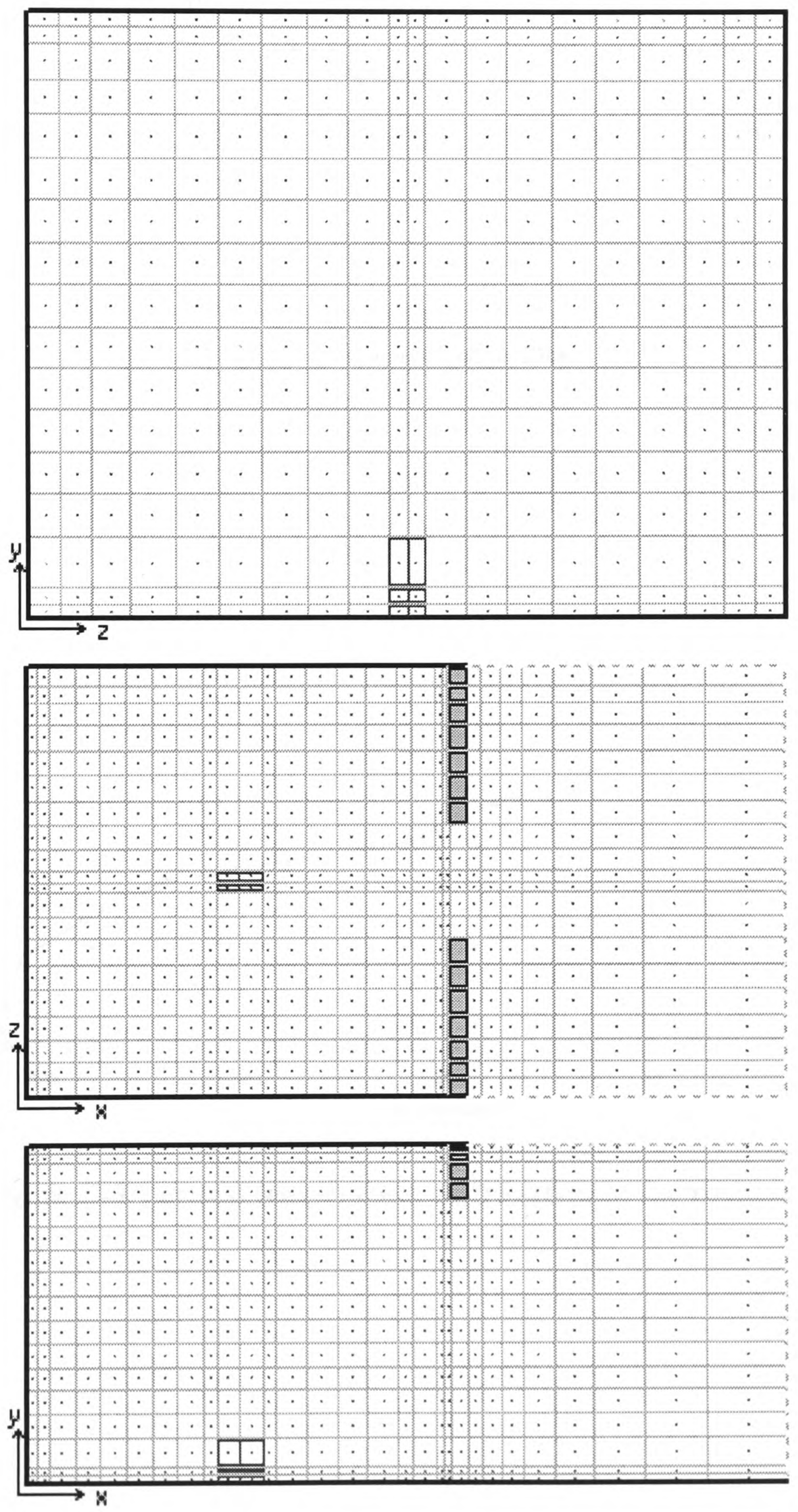


Figure C-2 Steckler case mesh (X, Y and Z planes)

Example results

There exists empirical data for certain points in the compartment (gathered by Steckler [Steckler-82]) and therefore it became almost a tradition in Fire Modelling that the solution of the Steckler case is presented as the set of the following results:

- vertical temperature distribution in one of the corners close to the door
- vertical temperature distribution in the centre of the doorway
- vertical velocity distribution in the centre of the doorway

For this case the results are normally gathered at a point where the solution reaches a steady state (200s in our case). Consequently, the values are easy to compare across different CFD codes. The results from SMARTFIRE are shown in Figures C-3, C-4 and C-5.

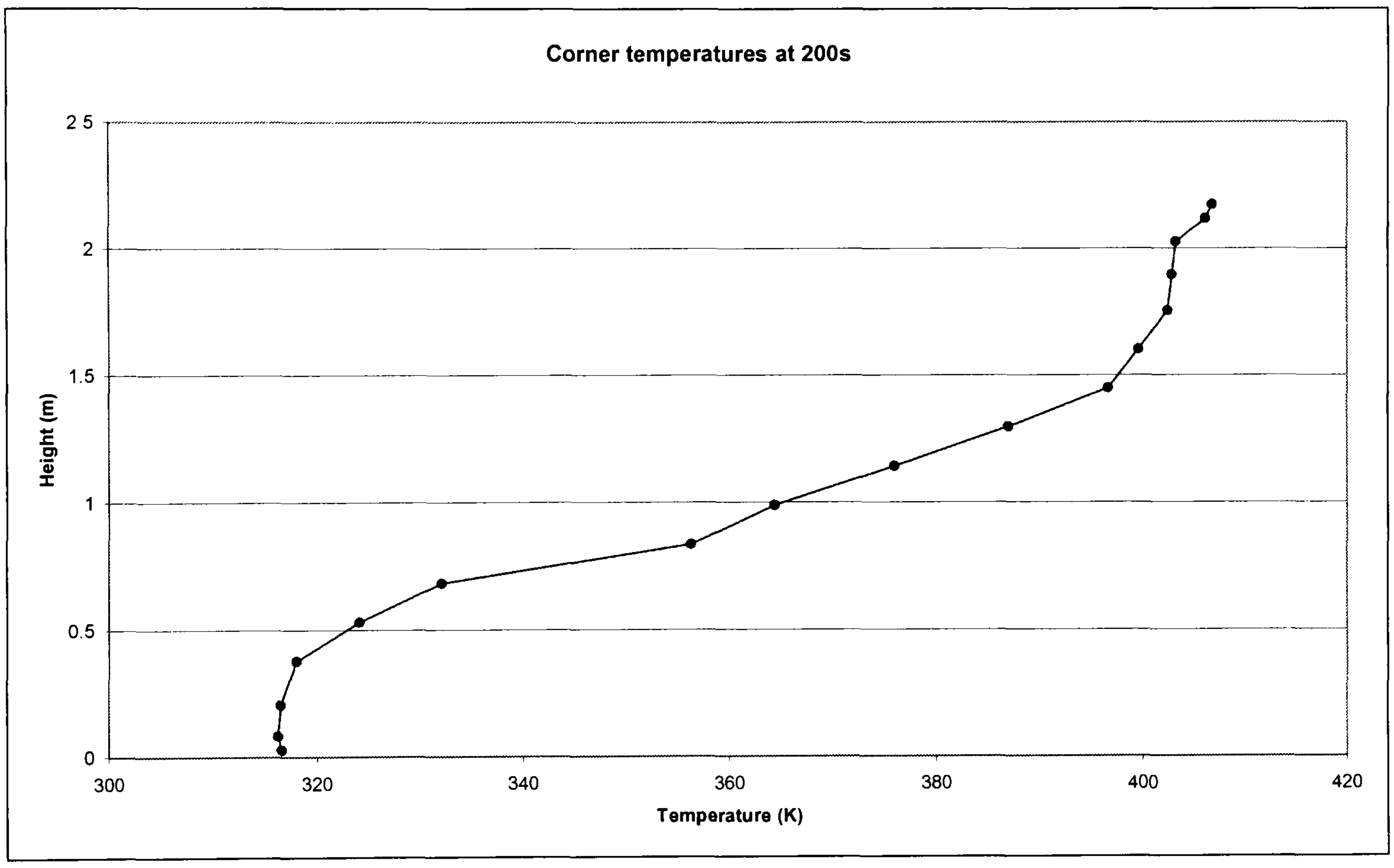


Figure C-3 Vertical temperature distribution in the room corner

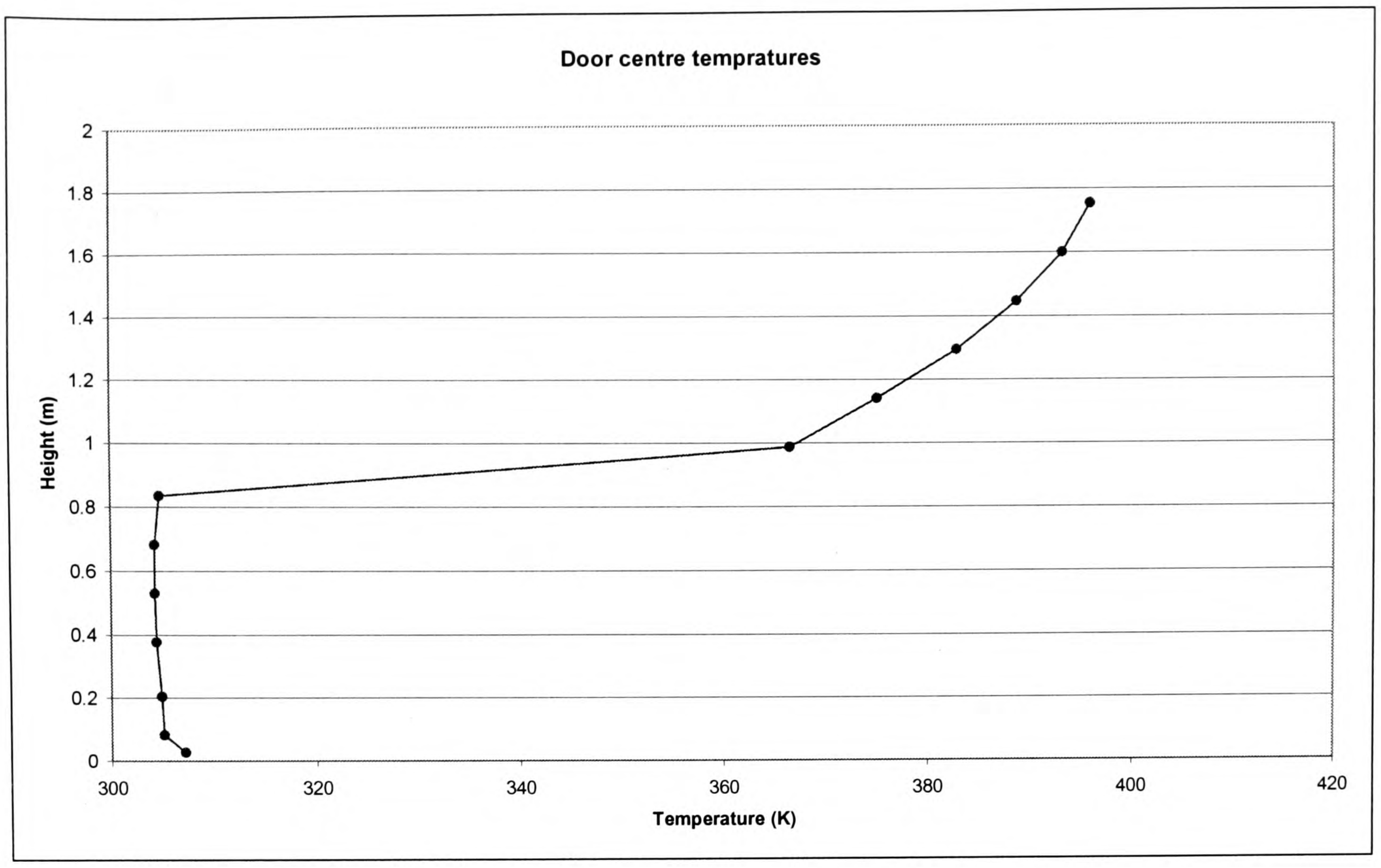


Figure C-4 Vertical temperature distribution in the doorway

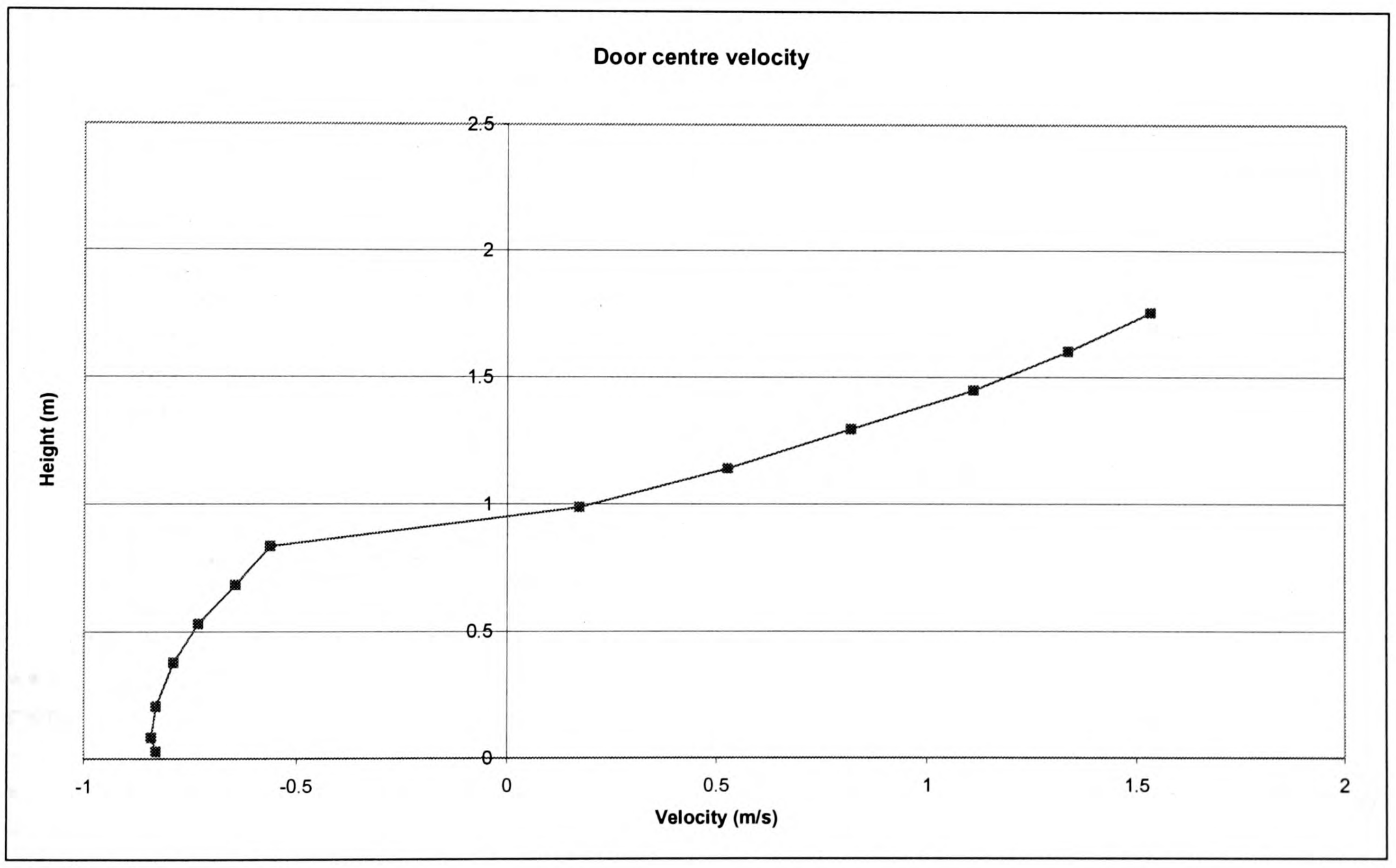


Figure C-5 Velocity distribution in the doorway

Appendix D

Main setup file for Steckler case (.inf file)

```
RUN PROBLEM
   FILENAME = a74_med_mesh
   TITLE Fire case a74_med_mesh
   DIMENSION  3
   ENABLE KBS
   CARTESIAN MESH
   STRUCTURED MESH
   BFC MESH DIMENSIONS
      NX   29
      NY   17
      NZ   20
   END
***   AUTO START
***   SETUP MODE
***   RESTART
END
*
*
*
PROBLEM DEFINE
   TRANSIENT
      TIME STEP              1
      NUMBER OF TIME STEPS   200
   FLOW
   TURBULENT
   HEAT TRANSFER
   SIX FLUX RADIATION
      SCATTERING COEFF          0
      AMBIENT ABSORPTION COEFF  0.01
      MINIMUM ABSORPTION COEFF  3.5
      MAXIMUM ABSORPTION COEFF  7
      MINIMUM ABSORPTION TEMP   323
      MAXIMUM ABSORPTION TEMP   1289
      WALL EMISSIVITY           0.8
   END
***   CROSS PRODUCT TERMS
END
*
*
*
INITIAL VALUES
   U-VELOCITY          0.00
   V-VELOCITY          0.00
   W-VELOCITY          0.00
   PRESSURE            0.00
   TEMPERATURE       303.75
   KINETIC ENERGY      0.01
   DISSIPATION RATE    0.01
END
*
*
*
```

```
MATERIAL PROPERTIES
   NUMBER OF MATERIALS 3
   DEFINE MATERIAL NUMBER  1
      MATERIAL NAME Standard Air
      CONDUCTIVITY CONSTANT  0.02622
      SPECIFIC HEAT CONSTANT 1045.78
      VISCOSITY CONSTANT     1.6e-005
      DENSITY IDEAL GAS LAW
         MOLECULAR WEIGHT    29.35
      NATURAL STATE          GAS
      THERMAL EXPANSION      0.003292
      DISCONTINUITY HANDLING    NO SLIP
   END
   DEFINE MATERIAL NUMBER  2
      MATERIAL NAME Wall Default Material
      CONDUCTIVITY CONSTANT  0.69
      SPECIFIC HEAT CONSTANT 840
      VISCOSITY CONSTANT     1e+010
      DENSITY CONSTANT       1600
      NATURAL STATE          SOLID
      THERMAL EXPANSION      0
      DISCONTINUITY HANDLING    NO SLIP
   END
   DEFINE MATERIAL NUMBER  3
      MATERIAL NAME Non Conducting Material
      CONDUCTIVITY CONSTANT  0.01
      SPECIFIC HEAT CONSTANT 10000
      VISCOSITY CONSTANT     1e+010
      DENSITY CONSTANT       10000
      NATURAL STATE          SOLID
      THERMAL EXPANSION      0
      DISCONTINUITY HANDLING    NO SLIP
   END
END
*
*
*
BOUNDARY CONDITIONS
   NUMBER OF FACE PATCHES    10
   NUMBER OF VOLUME PATCHES  1
*
*             Start             2D             Boundary             patch
      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
* ORIENTATION =>  LOW-X    EXTENT INDICES => 1 1 1 5 1 9
* BOUNDARY IS A WALL PATCH
   DEFINE FACE PATCH NUMBER        1
      STATIONARY WALL
      SOLID RADIATION BOUNDARY
      WALL EMISSIVITY           0.8
      MATERIAL INDEX         2
      PATCH THICKNESS        0.1
      TURBULENT WALL LAYER FOR HEAT
         FLUX COEFFICIENT     10
         AMBIENT TEMPERATURE  303.75
      END
   END
*             End             2D             Boundary             patch
      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
*
*
```

```
*           Start           2D           Boundary           patch
    >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
* ORIENTATION => HIGH-Y    EXTENT INDICES => 1 10 5 5 1 9
* BOUNDARY IS A WALL PATCH
  DEFINE FACE PATCH NUMBER        2
     STATIONARY WALL
     SOLID RADIATION BOUNDARY
     WALL EMISSIVITY          0.8
     MATERIAL INDEX          2
     PATCH THICKNESS         0.1
     TURBULENT WALL LAYER FOR HEAT
        FLUX COEFFICIENT     10
        AMBIENT TEMPERATURE  303.75
     END
  END
*           End           2D           Boundary           patch
    >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
*
*
*           Start           2D           Boundary           patch
    >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
* ORIENTATION =>  LOW-Z    EXTENT INDICES => 1 10 1 5 1 1
* BOUNDARY IS A WALL PATCH
  DEFINE FACE PATCH NUMBER        3
     STATIONARY WALL
     SOLID RADIATION BOUNDARY
     WALL EMISSIVITY          0.8
     MATERIAL INDEX          2
     PATCH THICKNESS         0.1
     TURBULENT WALL LAYER FOR HEAT
        FLUX COEFFICIENT     10
        AMBIENT TEMPERATURE  303.75
     END
  END
*           End           2D           Boundary           patch
    >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
*
*
*           Start           2D           Boundary           patch
    >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
* ORIENTATION => HIGH-X    EXTENT INDICES => 11 11 1 5 1 9
* BOUNDARY IS A FREE-SURFACE PATCH
  DEFINE FACE PATCH NUMBER        4
     OUTLET
     ADIABATIC
     FREE RADIATION BOUNDARY
  END
*           End           2D           Boundary           patch
    >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
*
*
*           Start           2D           Boundary           patch
    >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
* ORIENTATION =>  LOW-Y    EXTENT INDICES => 11 11 1 1 1 9
* BOUNDARY IS A WALL PATCH
  DEFINE FACE PATCH NUMBER        5
     STATIONARY WALL
     SOLID RADIATION BOUNDARY
     WALL EMISSIVITY          0.8
     ADIABATIC
  END
```

```
*            End            2D            Boundary            patch
      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
*
*
*            Start          2D            Boundary            patch
      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
* ORIENTATION => HIGH-Y    EXTENT INDICES => 11 11 5 5 1 9
* BOUNDARY IS A FREE-SURFACE PATCH
  DEFINE FACE PATCH NUMBER        6
     OUTLET
     ADIABATIC
     FREE RADIATION BOUNDARY
  END
*            End            2D            Boundary            patch
      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
*
*
*            Start          2D            Boundary            patch
      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
* ORIENTATION =>  LOW-Z    EXTENT INDICES => 11 11 1 5 1 1
* BOUNDARY IS A FREE-SURFACE PATCH
  DEFINE FACE PATCH NUMBER        7
     OUTLET
     ADIABATIC
     FREE RADIATION BOUNDARY
  END
*            End            2D            Boundary            patch
      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
*
*
*            Start          2D            Boundary            patch
      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
* ORIENTATION => HIGH-Z    EXTENT INDICES => 11 11 1 5 9 9
* BOUNDARY IS A FREE-SURFACE PATCH
  DEFINE FACE PATCH NUMBER        8
     OUTLET
     ADIABATIC
     FREE RADIATION BOUNDARY
  END
*            End            2D            Boundary            patch
      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
*
*
*            Start          2D            Boundary            patch
      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
* ORIENTATION =>  LOW-Y    EXTENT INDICES => 1 10 1 1 1 9
* BOUNDARY IS A WALL PATCH
  DEFINE FACE PATCH NUMBER        9
     STATIONARY WALL
     SOLID RADIATION BOUNDARY
     WALL EMISSIVITY          0.8
     ADIABATIC
  END
*            End            2D            Boundary            patch
      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
*
*
*            Start          2D            Boundary            patch
      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
* ORIENTATION => HIGH-Z    EXTENT INDICES => 1 10 1 5 9 9
* BOUNDARY IS A WALL PATCH
```

```
   DEFINE FACE PATCH NUMBER        10
      STATIONARY WALL
      SOLID RADIATION BOUNDARY
      WALL EMISSIVITY           0.8
      MATERIAL INDEX          2
      PATCH THICKNESS         0.1
      TURBULENT WALL LAYER FOR HEAT
         FLUX COEFFICIENT     10
         AMBIENT TEMPERATURE  303.75
      END
   END
*           End             2D             Boundary            patch
      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
*
*
*           Start              3D             Fire             patch
      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
* EXTENT INDICES => 4 4 1 2 5 5
   DEFINE VOLUME PATCH NUMBER  1
      ENTHALPY   4.99206e+006
   END
*            End             3D             Fire             patch
      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
*
END
*
*
*
RELAXATION
   FALSE TIME STEP
      U-VELOCITY          0.2
      V-VELOCITY          0.2
      W-VELOCITY          0.2
      KINETIC ENERGY      0.1
      DISSIPATION RATE    0.1
      ENTHALPY            0.5
   END
   LINEAR RELAXATION
      PRESSURE            0.4
      U-VELOCITY          0.6
      V-VELOCITY          0.6
      W-VELOCITY          0.6
*      KINETIC ENERGY      0.1
*      DISSIPATION RATE    0.1
*      ENTHALPY            0.2
      DENSITY             0.6
      BUOYANCY            0.6
      ABSORPTION COEFF    1.0
      RADIATION X POS     0.2
      RADIATION X NEG     0.2
      RADIATION Y POS     0.2
      RADIATION Y NEG     0.2
      RADIATION Z POS     0.2
      RADIATION Z NEG     0.2
   END
   SOLVER RELAXATION
***      PRESSURE            0.2
***      U-VELOCITY          0.5
***      V-VELOCITY          0.5
***      W-VELOCITY          0.5
***      KINETIC ENERGY      0.1
***      DISSIPATION RATE     0.1
***      ENTHALPY             0.2
   END
END
*
*
*
SOLVER CONTROL
   OUTER ITERATIONS       50
   FLOW ITERATIONS        1
   GLOBAL TOLERANCE       0.0001
   DEFAULT TOLERANCE      1e-008
   SOLVER TYPE
      PRESSURE             SOR
      U-VELOCITY           JOR
      V-VELOCITY           JOR
      W-VELOCITY           JOR
      KINETIC ENERGY       SOR
      DISSIPATION RATE     SOR
      ENTHALPY             SOR
      RADIATION_X_NEG      SOR
      RADIATION_X_POS      SOR
      RADIATION_Y_NEG      SOR
      RADIATION_Y_POS      SOR
      RADIATION_Z_NEG      SOR
      RADIATION_Z_POS      SOR
   END
   SOLVER ITERATIONS
      PRESSURE             40
      U-VELOCITY            2
      V-VELOCITY            2
      W-VELOCITY            2
      KINETIC ENERGY       20
      DISSIPATION RATE     20
      ENTHALPY             20
      RADIATION_X_NEG      20
      RADIATION_X_POS      20
      RADIATION_Y_NEG      20
      RADIATION_Y_POS      20
      RADIATION_Z_NEG      20
      RADIATION_Z_POS      20
   END
END
*
*
*
RESIDUAL METHODS
   PRESSURE           REFERENCE L2 NORM    6
   U-VELOCITY         REFERENCE L2 NORM    2
   V-VELOCITY         REFERENCE L2 NORM    3
   W-VELOCITY         REFERENCE L2 NORM    2
   KINETIC ENERGY     REFERENCE L2 NORM    0.5
   DISSIPATION RATE   REFERENCE L2 NORM    1
   ENTHALPY           REFERENCE L2 NORM    1e+006
   TEMPERATURE        REFERENCE L2 NORM    1000
   BUOYANCY           REFERENCE L2 NORM    10
   RADIATION_X_NEG    REFERENCE L2 NORM    20000
   RADIATION_X_POS    REFERENCE L2 NORM    20000
   RADIATION_Y_NEG    REFERENCE L2 NORM    20000
   RADIATION_Y_POS    REFERENCE L2 NORM    20000
   RADIATION_Z_NEG    REFERENCE L2 NORM    20000
   RADIATION_Z_POS    REFERENCE L2 NORM     20000
   ABSORPTION_COEFF   REFERENCE L2 NORM     5
END
*
*
*
PRINTOUT CONTROL
CREATE VAR FILE
CREATE RESTART FILE
***   NO RESTART FILE
***   NO DATABASE SAVES
***   NO VISUAL SAVES
***   NO STATUS SAVES
***   NO RESULT SAVES
AUTOMATIC SAVING
***   CREATE STEADY VISUAL EVERY      100
***   CREATE TRANSIENT VISUAL EVERY   1
***   CREATE STEADY RESTART EVERY     100
CREATE TRANSIENT RESTART EVERY  50
***   CREATE STEADY RESULTS EVERY     50
CREATE TRANSIENT RESULTS EVERY  20
***   CREATE STEADY GRAPHS EVERY      10
CREATE TRANSIENT GRAPHS EVERY   10
***   OUTPUT ITERATION NUMBERS    1  ONWARDS
***   OUTPUT TIME STEP NUMBERS    1  ONWARDS
   USE BINARY RESTART FILE
***   USE ASCII RESTART FILE
   CREATE PHI FILE
   FLOWVIS PHI FORMAT
   MONITOR LOCATION   2.83  1.7  1.399
***   SILENT
   SUCCINCT
   PRINTOUT FREQUENCY 1
   CFD PROCESS STEPS  1
***   CREATE DEBUG FILE
   CREATE LOG FILE
***   CREATE TABLE FILE
   DEFINE PLOT NUMBER    1
      TITLE Stack temperatures
      PATH  2.495  0  2.495    2.495  2.2  2.495
      TEMPERATURE
      Y COORD
   END
   DEFINE PLOT NUMBER    2
      TITLE Door temperatures
      PATH  2.83  0  1.399     2.83  2.2  1.399
      TEMPERATURE
      Y COORD
   END
   DEFINE PLOT NUMBER    3
      TITLE Door velocities
      PATH  2.83  0  1.399     2.83  2.2  1.399
      U-VELOCITY
      Y COORD
   END
END
*
*
*
GENERAL INFORMATION
   NOT BOUSSINESQ
```

```
***   BOUSSINESQ
   GRAVITY X COMPONENT     0
   GRAVITY Y COMPONENT     -9.81
   GRAVITY Z COMPONENT     0
   REFERENCE DENSITY             1.17756
   REFERENCE TEMPERATURE         303.75
   PRESSURE AT ZERO COORDINATE   101325
***   DIFFERENCING SCHEME              UPWIND
***   DIFFERENCING SCHEME              HYBRID
***   DIFFERENCING SCHEME              POWER LAW
***   DIFFERENCING SCHEME              EXPONENTIAL
***   KE SOURCE LINEARISATION METHOD   1
***   KE SOURCE LINEARISATION METHOD   2
***   KE SOURCE LINEARISATION METHOD   3
***   MINIMAL STORAGE
***   NOT MINIMAL STORAGE
END
*
*
*
DEBUG CONTROL
****** To use any item: remove * characters to uncomment (and
      activate)
****** You should also activate CREATE DEBUG FILE in printout control
***   DEBUG ITERATION NUMBERS    1   ONWARDS
***   DEBUG TIME STEP NUMBERS    1   ONWARDS
***   DEBUG CELL NUMBERS         1   TO  9860
***   PRESSURE
***   U-VELOCITY
***   V-VELOCITY
***   W-VELOCITY
***   KINETIC ENERGY
***   DISSIPATION RATE
***   ENTHALPY
***   CONVECTIONS
***   PROPERTIES
***   GEOMETRY
******   ALL
******   CHECK VARIABLES
******   CHECK MEMORY
******   CHECK SETUP
END
*
*
*
STOP
```

Appendix E

Automatic Dynamic Control of CFD Based Fire Modelling Simulations – INTERFLAM 2001

AUTOMATIC DYNAMIC CONTROL OF CFD BASED FIRE MODELLING SIMULATIONS

D. Janes, J.A.C. Ewer, E.R. Galea, M.K. Patel and B. Knight
Fire Safety Engineering Group
University of Greenwich, UK
http://fseg.gre.ac.uk

ABSTRACT

Automatic dynamic control of the CFD solution process is being pursued in fire modelling applications for two primary reasons. Firstly, with such a system it may be possible to automate the control of CFD simulations and dynamically adjust the control parameters in order to ensure the stability and convergence of fire field model simulations. In this way, the technique would remove some of the "black art" associated with fire field modelling and make it accessible to a wider audience. Secondly, it is hoped that through the optimal setting of control parameters, these techniques will also lead to a reduction in simulation times. The development and use of these techniques is being explored through the use of the SMARTFIRE fire field model. This paper describes recent progress in the development of automatic dynamic control of fire modelling simulations and presents results from the simulation of a non-trivial fire case using the system. Results indicate that the system can assure convergence and has demonstrated substantial savings in execution time with savings of the order of 60% of overall run-time.

1.0 INTRODUCTION

Over recent years fire field modelling[1,2] has begun the transition from the confines of the research laboratory to the desk of the fire safety engineer. To a certain extent, this move has been driven by the demands of performance based building codes. To facilitate this move, improvements in the usability of the field modelling software must be made. Two areas requiring improvement concern the high run times associated with performing fire simulations and the difficulties associated with ensuring that a converged solution has been achieved. This paper attempts to address these issues.

The majority of the time consumed by a fire field model in performing a simulation is spent in the numerical solution of the Partial Differential Equations (PDE) that define the model[3]. The solution process generally consists of first generating and then iteratively solving a single algebraic equation at each control volume (cell) in the computational domain for each PDE. The general mode of operation of fire field modelling software is that the flow field and pressure fields are unknown at the start of the simulation. The heating due to the fire source(s) and consequent density changes lead to buoyancy forces that drive the flow. The difficulty with this solution technique is that the initial stages of a simulation are comparatively unstable and generally require significant under-relaxation to prevent instabilities from causing divergent solutions. Whilst there is little problem in applying such tight under-relaxation, to start a simulation, the same parameters can have a detrimental effect on the quality or efficiency of the simulation in later stages. In later stages, small changes compounded by excessive under-relaxation can falsely stagnate the solution.

The duration and magnitude of the required relaxation values are largely unknown, are mostly problem specific and may also depend on the algebraic solvers employed and even the particular CFD software used. The obvious solution is to apply significant under-relaxation at the start of the simulation and then, when the processing appears to have stabilised, to apply less under-relaxation for the remainder of the simulation. In practice this is the method adopted in most batch mode systems. However, this technique is far from ideal because similar instabilities can occur later as particular flow features develop. Some flow features which can destabilise a solution are changes in orientation of fire plumes or ceiling jets, changes in height of the neutral plane and the creation or destruction of a re-circulation region within the flow field.

As the complexity of CFD software and modelling capabilities increase so there will be additional difficulties introduced by the temporal effects associated with more sophisticated behaviour such as flash-over, "breaking" windows, opening doors, secondary ignition and fire spread. None of these destabilising effects are handled by crude batch mode software without considerable manual intervention that is both tedious to apply and prone to errors. Ideally, automated intelligent agents are required to monitor the solution status and to make command decisions, based on the solution status, so that processing continues both optimally and in a stable manner. The development of new methods for the intelligent control of CFD convergence characteristics is examined in this paper[4].

Clearly the imposition of dynamic control by an intelligent "agent" requires a suitable interactive and fully featured fire field modelling code. This research was largely made possible by use of the SMARTFIRE CFD engine[5-11].

2.0 OVERVIEW OF THE SMARTFIRE FIRE FIELD MODEL

SMARTFIRE is an open architecture CFD environment written in C++ that is comprised of four major components: CFD numerical engine, Graphical User Interfaces, Automated meshing tool and the Intelligent Control System. One of the main aims of the SMARTFIRE development group is to make fire field modelling more accessible to fire engineers with limited CFD experience. One of the ways this is achieved is by embedding expert knowledge into the CFD software. The functionality and construction of the SMARTFIRE system have been described in previous publications[5-11,17], and so only a brief outline is presented here.

The CFD engine in SMARTFIRE has many additional physics features that are required for fire field modelling[1,2]. These include a six-flux radiation[12] model, a multiple ray radiation model[13], provision for heat transfer through walls, a volumetric heat release model or gaseous combustion model (using the eddy dissipation model)[14] to represent fires, smoke modelling and turbulence (using a two equation K-Epsilon closure with buoyancy modifications).

The main Graphical User Interface (GUI) is used to specify the problem. Through this GUI the user sets the geometry, specifying the location of walls, wall materials, internal compartments and obstacles (such as desks, stairs or partitions). Also specified are the location of vents (along with any fans, inlets or outlets), the nature and location of the fire, the radiation model to be used and gaseous properties such as absorption coefficients. The GUI also provides access to the automated meshing tool and manual mesh editor as well as the CFD engine. Once the problem has been specified, the automated mesh generation tool is used to generate the control volume cells for the problem. It is important to note that embedded expertise is used to determine a mesh and cell budget that is appropriate for a *reasonable* solution to the problem.

The CFD engine is used to simulate the fluid dynamics of fire development by numerically solving a set of differential equations that describe the laws governing the physical processes inside the domain. The solution is found by performing a series of consecutive approximations, whose quality is measured by the residual error (defined as the magnitude of change between two adjacent approximations). The residual error is calculated separately for each solved variable and the time step is said to have "converged" if the residuals of all variables fall below a specified tolerance.

The interactive CFD code has its own unique multiple-windowed user interface. Unlike traditional fire field models, this allows the user to interact with the solution through *observation* of the developing solution (using graphs, visualisations and data explorers) and by allowing the user to make *adjustments* to control parameters. Such adjustments in traditional CFD codes generally involve terminating a simulation, editing input files and restarting. As part of the SMARTFIRE development, the software has undergone considerable validation[11]. Further information concerning SMARTFIRE (including a demonstration) may be found on the developers web site[15].

3.0 BACKGROUND TO DYNAMIC SOLUTION CONTROL

The first attempt to develop automated solution control was based on experience of interactively running CFD simulations in SMARTFIRE. A simple system was implemented for dynamic control using a rule-driven architecture where the control actions were limited to small relaxation adjustments with decision making implemented as a basic state-recognition algorithm. The rules were quite simple and the control decisions were based on tracking the local trends (gradients) of the residual errors. The gradients determined if the particular residual was actually diverging or converging, and hence the relaxation value (for a solved variable) was either increased or reduced. Variables were grouped to reflect the way in which they describe the physical processes in the domain (e.g. PRESSURE and VELOCITIES were assessed together as "flow related variables").

The techniques used in the prototype control system were demonstrated using a simple 2D fire case with a removable partition (to simulate burn through to a second compartment). The control system was able to reduce the overall number of sweeps by 50% when compared with a non-controlled simulation[7,8]. This result was encouraging but the system was not able to control a 3D scenario effectively. These problems were attributed to greater degrees of freedom and greater complexity of 3D cases. Nevertheless, the prototype[16,17] showed that it was possible to automatically generate a converged solution along with considerable savings in run times with correct and efficient control of relaxation parameters.

4.0 AN INTELLIGENT CONTROL SYSTEM FOR CFD

To better understand the complex relationships between individual and combined control parameters in 3D problems, a parametric study was performed on a series of fire cases. The experiments were used to investigate the actions which experts use when interacting with or correcting a simulation. The analysis confirmed the experts' opinions about the stabilising effects of reducing relaxation values and time step size reduction. They also confirmed that increasing relaxation values often has the added benefit of speeding up convergence and consequently accelerating the simulation.

The experiments identified one class of control actions (relaxation values reduced and time step increased) as ineffective since they did not provide any tangible benefits. The experiments also helped to link control actions with a particular simulation state (e.g. simply reducing time step size can restore convergence). The experiments showed that the effects of control changes vary depending on the case but that there are also some generally applicable rules.

A new control architecture based on heuristic searching was implemented. The new system (called an Intelligent Control System - ICS) controls both the relaxation values and the time step size while also adjusting the maximum number of iterations in order to guarantee full convergence of every time step. All the changes to the simulation control parameters are applied between time steps and are preceded by a comprehensive heuristic search in order to obtain the best control parameters at this stage of the simulation. During searches, the simulation does not advance since the current time step is repeated several times using different control parameters. The relevant results (i.e. residual error graphs) are then analysed using a sophisticated assessment algorithm. This algorithm determines which of the experimental time steps provided the best results and applies the appropriate control parameters to subsequent time steps until the next evaluation point is reached.

It was decided that the ICS should constantly monitor the simulation progress and be able to perform purposeful and effective control actions. Consequently, the system had to recognise certain important states during the simulation (e.g. convergence problems) and trigger appropriate control actions. Finally, the improvements had to be evaluated, with emphasis on the following issues:

- *Does the system provide improvements in terms of simulation speed and accuracy of the results?*
- *Can such a system assure the convergence of every time step?*
- *Does the system recover from solution excursions and / or faults?*

5.0 HEURISTIC SEARCH ALGORITHM

Research on the ICS demonstrated that control actions were best applied between time steps. Due to the long time required to assess the results of a single control action, expert users actually perform very few manual adjustments. However, the automated system never gets tired and can examine several different types of changes and can attempt to find an almost-optimal set of control parameters by performing a more comprehensive search. The run-time search for the "best" control parameters, the intelligent assessment of the adjustment results and automatic divergence detection and recovery are the basic principles of the ICS.

The most comprehensive approach would be to test all the possible combinations of sets of parameters for every time step and then select the "path" that provides the shortest execution time. Such an exhaustive search could find the optimal path to the solution but the cost of the search would be enormous. There are two main difficulties associated with such an exhaustive search:

- The control parameters are continuous. Thus it is impossible to create a limited set of different configurations. Consequently, the branching factor of the search tree is infinite. This problem can be overcome by using a discrete subset of changes to the control parameters.

- The cost of exploring a single test set of parameters is of the same order of magnitude as the cost of a single time step. A typical simulation contains at least 100 time steps and runs for several hours or even days. Hence, the cost of an exhaustive exploration is untenable.

There are some special properties of this control scenario that can be used to tackle the second problem. As mentioned earlier, we are not actually interested in finding the optimal path to the solution. The two main goals are (1) to ensure the validity of the results and (2) to reduce the simulation time. These goals can be achieved without finding the optimal path. Actually, finding the shortest route to the solution can prevent the agent from achieving the second goal since the cost of the search forms part of the total execution time. Also, a time step is considered valid if the residuals have converged and its correctness does not depend on the set of parameters used. Consequently, even a sub-optimal path can provide acceptable performance. Generally, there is little benefit in trying to find a better set of parameters for a time step that has already produced the correct results.

6.0 EVALUATION FUNCTION

The heuristic search cannot be performed without an appropriate evaluation function. The experiments have to be evaluated to select the best one. The fire modelling experts believe that most of the information relevant to the control system can be extracted from the residual error graphs. It is also accepted that convergence speed (the number of sweeps required to attain convergence) determines the performance. Generally – the less sweeps that are required to complete a unit of simulated time the better. Unfortunately there was some ambiguity about the influence of various features in the graph on the assessment result. Experts do not currently have the necessary knowledge since interactive CFD codes, that allow residual monitoring and run-time fine tuning, have only been developed recently. Furthermore, experts rarely watch the residual graphs to assess their quality because most CFD simulations take many hours to complete. Their graph assessment is usually limited to determining whether the residuals converge (trend is down) or diverge (trend is consistently upwards). Even such rudimentary assessment is rarely performed more then a few times during the whole simulation because residual monitoring is extremely tedious in typically long simulations.

Fire modelling experts were questioned about the behaviour, quality and performance of many graphs of monitored residual data and simulation results. The experts identified the three most important factors that influence their judgement when asked to sort a set of graphs according to their quality (defined as the best balance of speed and stability). The three most emphasised factors were:

- **Convergence speed** (number of sweeps required to attain convergence).
- **Presence and magnitude of irregularities** (residuals differ significantly from the global trend).
- **Presence and magnitude of oscillations** (strong periodic variations in residual values).

The experts agreed that these are a fairly comprehensive set of factors, which they use in their assessment of the state of a simulation. It was unclear how much influence each of these features has on the assessment result because the experts use an assessment process that is rather subjective. Each expert tends to use their own informal techniques or intuition. Nevertheless, identifying the factors used in the assessment provided the building blocks for a combined evaluation function.

6.1 Evaluation of Convergence Speed

The convergence speed is defined by the following formula:

$$convergence\_speed = \frac{number\_of\_sweeps}{simulation\_time} \qquad \left[s^{-1}\right]$$

Convergence speed can be calculated for the whole or part of a simulation, or for a single time step.

A method of extrapolation has been developed to provide early assessment of the quality of a graph. This early assessment is essential as it helps to quickly identify and terminate diverging time steps. Furthermore, if an experiment is assessed early as not providing the required improvement then it could be stopped immediately. Extrapolation required that both irregularities and oscillations were filtered from the graphs to leave the basic data trends. It was observed that many graphs had a highly segmented behaviour with distinct trends within each segment. It was realised that this could be exploited to give much better estimates of convergence points and hence give better speed and convergence assessment. Analysis of the changes in average graph gradient can be used to determine if a graph has any segments and to find the segment regions.

6.2 Evaluation of Irregularities

An irregularity is any feature in the residual graph that contains (at least in some part) an upward trend. Ideally, all the residuals are steadily decreasing. It is generally accepted that an upward trend is an intrinsic part of any irregularity. A technique has been developed to assess both the duration and the magnitude of such upward trends as a single value. The assessment method employs an algorithm commonly used for approximating a free-hand drawing with a set of line segments. Figure 1 shows an example graph and six consecutive steps of the approximation algorithm. The ends of the graph are first joined with a single line. The graph and the approximating line are analysed to find the graph-point that lies the furthest from the approximating line. The approximating line is split into two segments using this point. This procedure is repeated, producing more line segments and progressively better approximations until the error (the maximum distance from the graph to the approximation) drops below some specified tolerance. The dashed lines show the points at maximum distance from the approximating line. This algorithm is able to track all major changes and does not flatten any steep spikes making it ideal for irregularity detection.

The approximation produces a set of lines that describe the most relevant features of the graph but these lines have to be further analysed in order to obtain a meaningful assessment (See figure 2). The method employed is based on the assumption that both magnitude and duration of the irregularities are equally important for assessment purposes. The procedure calculates the area underneath segments with upward trend and uses this value as the irregularity estimator.

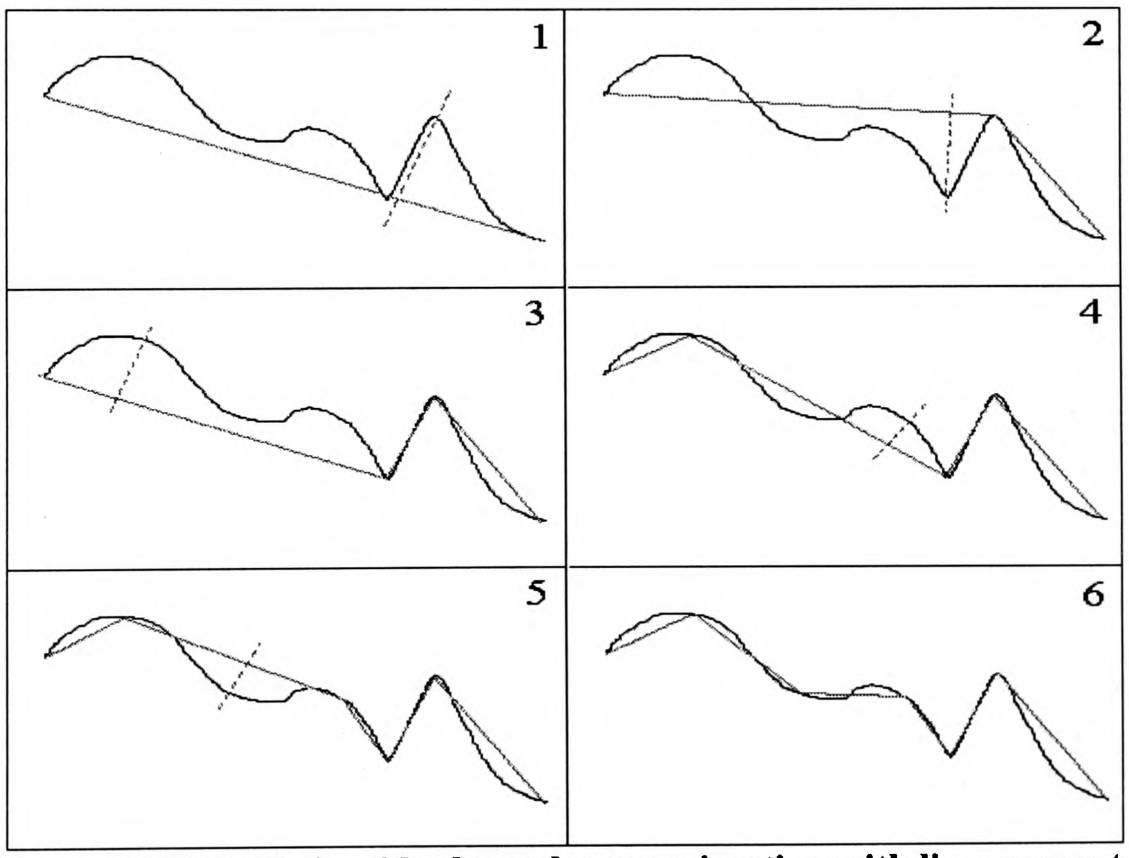


Figure 1: Method of residual graph approximation with line segments.

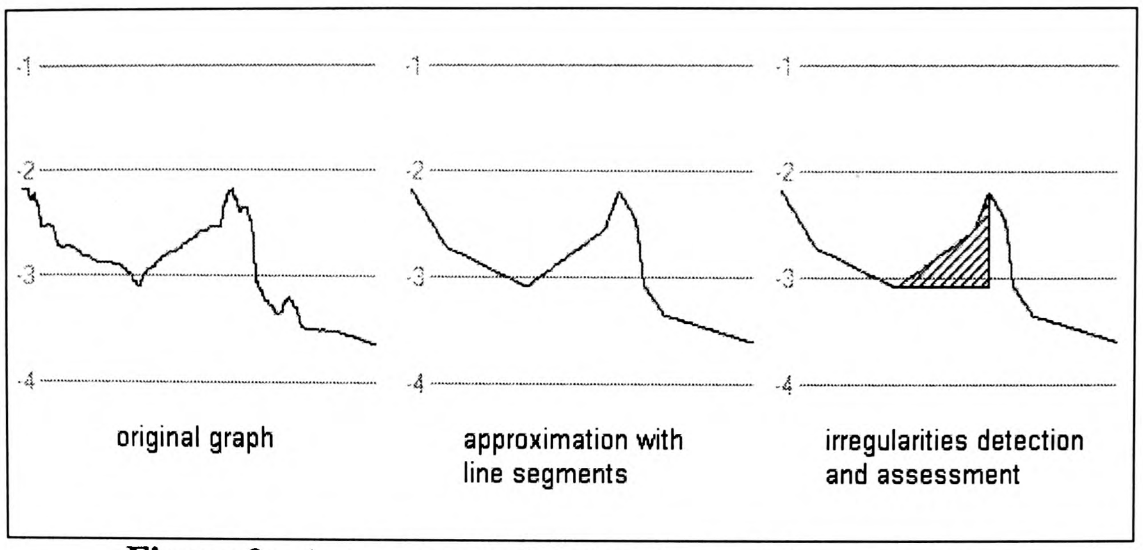


Figure 2: Assessment of residual graph irregularities.

6.3 Evaluation of Oscillations

The residuals contain various fluctuations but only some of them are relevant for the overall assessment. It was decided that a graph contains oscillations if its values change in some periodic way for more than two full cycles with amplitude larger than a specified tolerance. The amplitude constraint is very important since oscillations are quite common in residual graphs but only those with substantial amplitude and duration, are important. A minimum duration of oscillation is used to exclude "irregularities". The technique for oscillation assessment is based on Fourier Transforms and spectral analysis related to Fourier Transform. The problem is well suited to Fourier analysis since the graphs are made up of discrete, evenly spaced points, which facilitates the use of the Fast Fourier Transform. The algorithm was tested on various graphs and provided sufficiently accurate assessments of oscillation.

6.4 A Compound Evaluation Function

Each of the evaluation algorithms described so far was designed to assess a single feature of the residual graphs. The research into evaluation functions led to the important conclusion that an attempt to perform an absolute assessment is bound to fail. A single residual graph that is assessed to be very bad in one simulation can be excellent in another scenario. The only valid assessment must be based on a relative comparison of two different graphs from the same point in the simulation. It may be possible that an absolute assessment function exists but, if so, it must consider a variety of other factors such as geometry, heat release rate, time step size, recent progress and so on. Fortunately such a function is not necessary since the relative evaluation function is more than adequate for assessment purposes.

A prototype assessment function was developed that made use of the fact that at every decision point the search produced a significant number of residual graphs for comparison. The compound evaluation function only had to select the "best" residual graph and thus choose the best set of control parameters. The evaluation function separately assessed and graded all three metrics with clearly diverging graphs being immediately rejected. The assessment process then assigns each graph a number that represents its relative score. The scores were from 1 to n - where n was the total number of the graphs. The combined score for a particular graph is simply the sum of the individual metrics. In the final step the graph with the lowest final score is selected as the "best" one and the control parameters from this particular experiment are applied in subsequent time steps.

6.5 Fault Detection and Recovery

Fault detection is used to check for serious problems and run the heuristic search if necessary. The fire modelling experts identified a number of common problems:

- **Divergence** – Occurs when the iterative numerical solvers fail and the approximations proceed in a wrong direction with ever increasing errors.
- **Major convergence problems** – Similar to divergence but not as serious. The result from the simulation experiencing such problems can be either completely incorrect or close to the correct solution. It is considered safest to assume that the results are also incorrect.
- **Slow convergence** – The solver takes a very long time to achieve convergence (compared to preceding time steps). Usually the results are correct or close to being correct.
- **Oscillations** – The simulation results are not necessarily affected but the presence of oscillations often indicates other underlying problems (e.g. poor mesh quality) that can lead to inaccurate results or performance deterioration.

The "fault" recognition techniques are quite straightforward and are based on the feature extraction techniques, described previously, that are used for speed of convergence assessment.

The ICS consists of two main modules. The first module is responsible for continuous monitoring of the simulation progress, detecting problems and overall supervision of the simulation process. This "Monitor" module performs a continual assessment of the simulation as it is progressing. The second "Search" module performs the heuristic searches. When searching, the simulation is not progressing but the current time step is repeatedly executed with different sets of parameters in order to determine the best set of control parameters. Figure 4 shows part of a simulation controlled by the ICS. Initially the Monitor is engaged and two consecutive time steps are executed. After every time step the results are assessed and a decision is made about initiating a Search. After step t_n the search is not started but after t_{n+1} the Monitor decides to perform a Search to find a new set of parameters. The search could have been triggered because it was scheduled or because some fault was detected. If a Search is scheduled then the next step is used for experiments but if there were problems then the last time step is restarted and used in the Search. At point (5) the heuristic Search starts. A comprehensive set of experiments is conducted and the residual graphs are stored. At the next stage (6) the results of all the experiments are assessed using the evaluation function and the control parameters from the best experiment are applied to the simulation and used in the subsequent time steps. After the Search, the Monitor is engaged again and the simulation proceeds with a new set of control parameters (7).

In order to minimise search costs it was necessary to perform some state recognition so that counter-productive or ineffectual search experiments could be excluded.

The ICS proved to be perfectly capable of detecting faults in the solution process and restoring convergence. All convergence problems were properly detected and the search for a better set of control parameters initiated. The heuristic search was always able to recover from divergence. Due to the nature of the prototype evaluation function, the recovery actions often seriously impaired the time performance but nonetheless were always able to restore convergence.

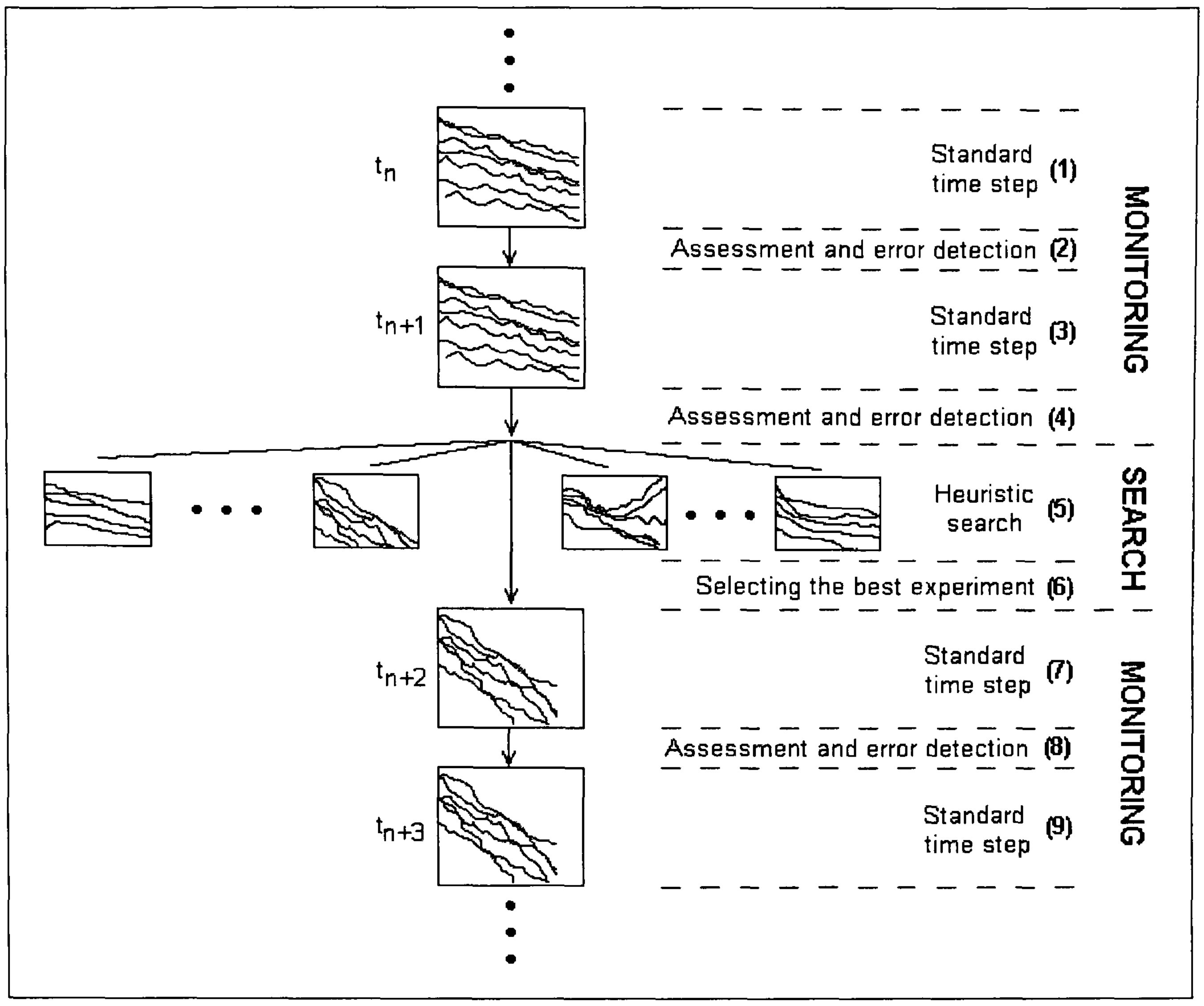


Figure 4: Simulation controlled by ICS.

6.6 Performance Enhancements for Heuristic Searching

It was noted that certain control actions had generally better performance impacts and hence the experiments could be ordered to give optimal performance even from a limited or truncated search.

A statistical analysis of all the convergence data obtained during the tests was performed. This led to the understanding that there were relationships between experiments that could be exploited to exclude ineffective experiments based on the test results of other experiments. Fire modelling experts confirmed the two main potential relationships between experiments as follows:

- If experiment A led to divergence then experiment B will also cause divergence.
- If experiment A led to faster convergence then experiment B will not give any faster convergence.

Thus it was possible to improve the heuristic search process by ordering and prioritising the search process.

7.0 RESULTS FOR A NON-TRIVAL FIRE MODELLING SCENARIO

The ICS has been implemented within a prototype version of SMARTFIRE and a number of fire simulations have been performed using the system. In each case, the ICS has been able to achieve a fully converged state along with a reduction in computer time. Here we describe the results from a non-trivial fire modelling case to demonstrate the benefits that the ICS provides.

The fire test scenario consisted of a single room compartment (R_{sx} = 6.0m, R_{sy} = 3.3m, R_{sz} = 4.0m) with a single doorway (D_{sy} = 1.8m, D_{sz} = 1.0m) rising from the floor in the middle of the longer compartment edge. A large wooden-crib fire (F_{sx} = 1.8m, F_{sy} = 1.0m, F_{sz} = 1.1m) is situated 0.08m above a supporting plinth (of height P_{sy} = 0.2m) that is located at x = 2.1m, y = 0.0m, z = 1.15m (See Figures 5 and 6).

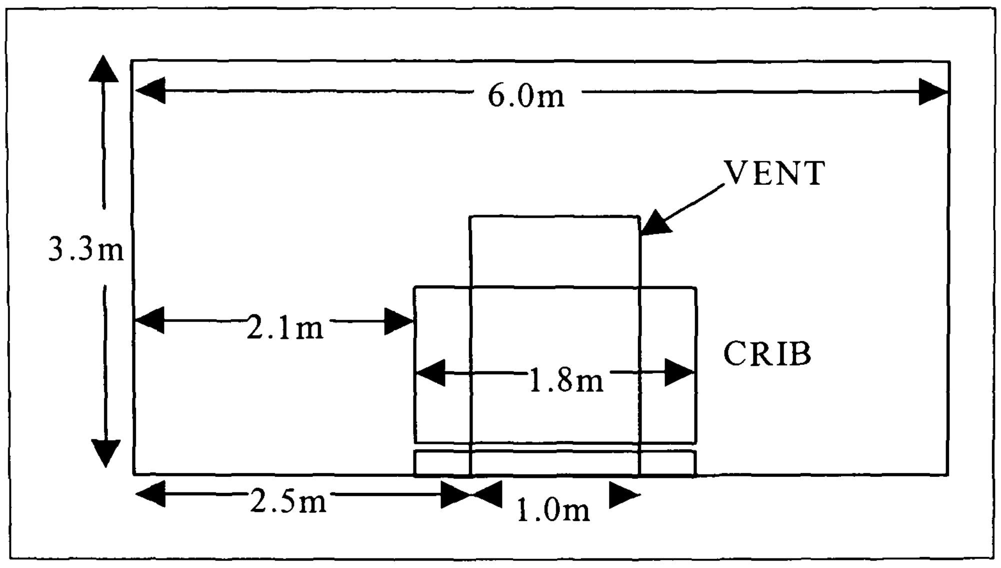


Figure 5: Front view of simulation geometry.

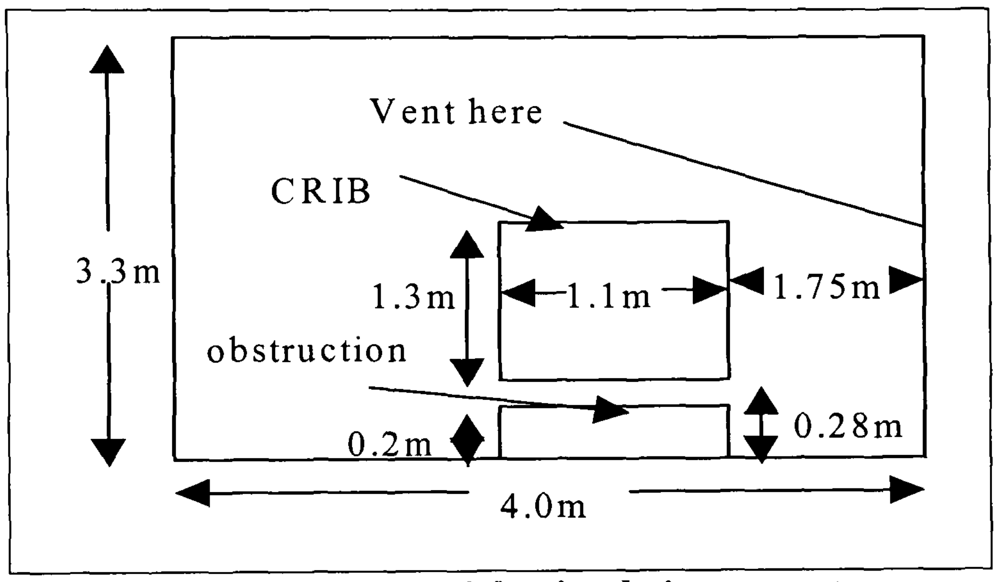


Figure 6: Side view of the simulation geometry.

The fire has a peak heat output of 3.5MW at 450 s and is defined by a 2 stage heat release curve that linearly ramps up to 10% of maximum heat output in 150 s and then rises linearly to the maximum heat output at 450 s and then stays at this maximum heat output. While a combustion model was not used in this example, there is no fundamental reason why one could not be used with similar benefits using this technique. The six-flux radiation model was enabled. The geometry was meshed using the recommended mesh from the SMARTFIRE automatic meshing tool. The mesh contains 22,968 cells (distributed as n_x = 29, n_y = 24, n_z = 33). The initial time step size was set to 1 s. All other convergence parameters, within SMARTFIRE, were set to their default values.

The simulation proceeds as expected with flash-over conditions developing at around 500 seconds into the simulation (see figure 7). The heat plume from the fire rapidly overwhelms the upper part of the room and spills out through the upper portion of the single doorway. The entrainment of cold air through the lower portion of the doorway, whilst substantial, is not sufficient to make the plume lean significantly away from the door (See figure 7). Almost the entire room is engulfed in the heat from the fire and the spill plume ejects strongly from the (relatively small) doorway.

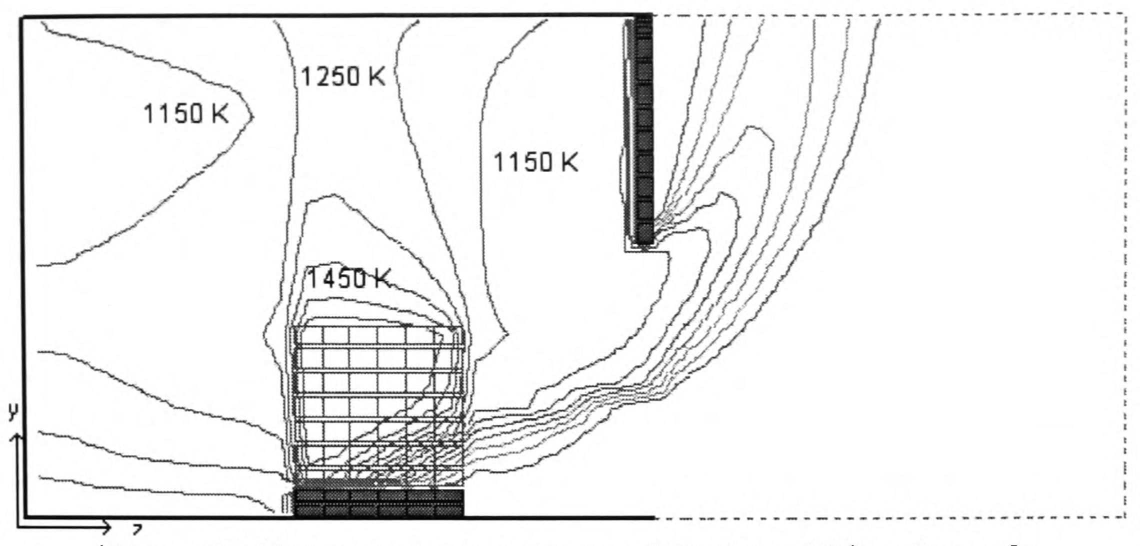


Figure 7: Temperature contours (K) at t = 500 s along the Compartment centre-line produced by the ICS controlled simulation.

The key comparison between the ICS controlled solution and conventional solution is depicted in figure 8. This depicts the number of sweeps required to complete every 5 s of simulated time, for the ICS controlled and conventional simulations. It is important to note that the starting conditions for both the ICS controlled and conventional simulations were identical (e.g. initial time step size of 1 s). Also, in the ICS-controlled simulation the heuristic search was performed after every 25 s of simulated time.

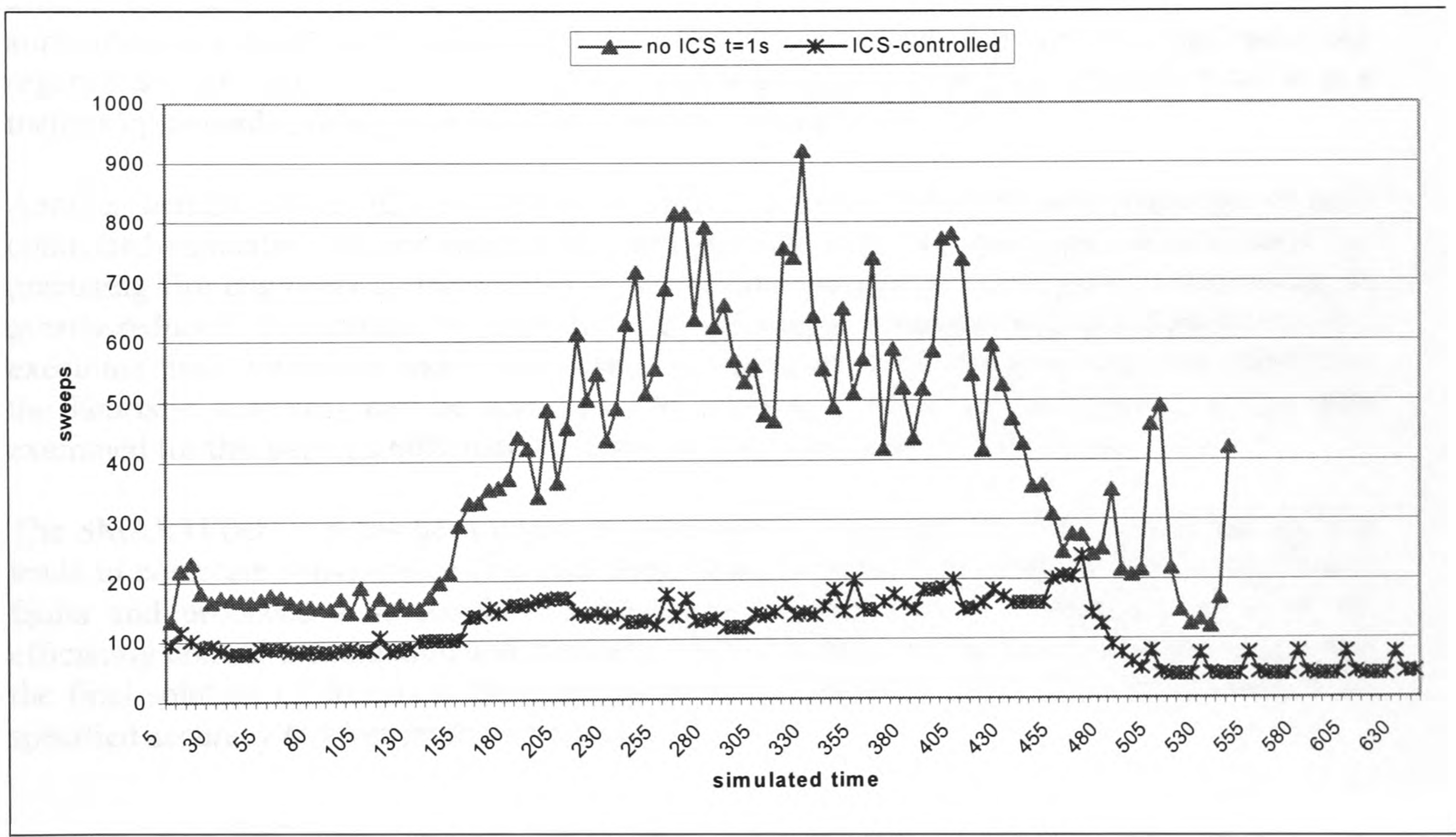


Figure 8: Number of sweeps per 5 s of simulated time for both non-controlled and ICS controlled simulations.

In the non-controlled case (no ICS with $\Delta t = 1$ sec), 10% of the time steps either failed to converge or diverged i.e. performed the maximum sweep number before residual was reduced below the tolerance level. Needless to say, this failure to converge each time step may create some doubt in the eventual solution produced using this approach. In contrast, the ICS-controlled simulation achieved a fully converged state at the end of each time step. Furthermore, it can be clearly seen from figure 8 that at each point the ICS-controlled simulation required substantially fewer sweeps to simulate any given 5-second period. The conventionally controlled system required 43,844 sweeps to complete the simulation while the ICS-controlled simulation required only 17,641 sweeps. This figure includes the experimental sweeps required for the heuristic searches. This reduction in the number of sweeps performed represents a 60% reduction in execution time since the overhead introduced by the ICS is minimal.

In the example, there is a real-time saving of 60% in using the ICS. It should be noted that the search cost consists of the time spent performing the experiments and the time spent assessing them but the cost of the assessments is actually negligible in comparison to the normal simulation and the heuristic search experiments.

8.0 CONCLUSIONS

An automatic dynamic control system for CFD based fire models has been developed. The system, known as an Intelligent Control System or ICS was developed from extensive analysis of human expert control actions, statistical analysis of test data and observations of simulation characteristics. This analysis gave rise to a search-based architecture with a heuristic evaluation function. A compound evaluation function was created, which made use of advanced pattern recognition algorithms designed to extract complex features from the residual graphs. This evaluation function has been successfully used in fire simulations and validated by comparisons between the assessment results from the automated ICS and from human fire modelling experts.

The main improvement gained from the use of the ICS is that the system attempts to ensure that all the time steps within a complex fire simulation converge, therefore improving the accuracy and fidelity of the results. In all the fire simulations examined so far, the control system was always able to recover from problems such as divergence. Furthermore, ICS controlled simulations are much more robust and are often able to obtain achieve converged solutions regardless of the initial control parameters. Automating the divergence recovery process is a major step towards producing a fully unsupervised simulation process.

Another benefit of the ICS is that it eliminates faulty or divergent runs common in non-controlled simulation environments. This can generate huge savings in turn-around times for practising fire engineers as the number of simulations needed to investigate a solution can be greatly reduced. In addition, by achieving fully converged solutions, substantial savings in real execution time (total time taken to simulate the simulation case including any time needed for the heuristic searches) can be achieved compared with non-controlled runs. In the case examined for this paper, a 60% real reduction in simulation time was achieved.

The SMARTFIRE ICS has been tested on a number of examples. In each case, using the ICS leads to complete convergence (for each time step), recovery from any solution excursions or faults and produces a substantial reduction in simulation time. Generally, the ICS can efficiently control a simulation and correctly react to common problems, which guarantees that the final solution (if found) is the best approximation that the model can produce within the specified accuracy (convergence tolerance).

9.0 FURTHER WORK

Further testing of the system is required. This will involve the use of the system on real fire problems. In addition, the cost of the heuristic searches is still significant. Possible improvements to the overall efficiency are being explored.

10.0 ACKNOWLEDGEMENTS

Work on this project was sponsored by UK EPSRC through research grant GR/L56749. In addition, Professor Galea is indebted to the UK CAA for their financial support of his chair in Mathematical Modelling.

11.0 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, pp 11-22.
2. "Combustion Fundamentals of Fire", Editor: Cox G., Academic Press, 1995.
3. Pantakar S.V., "Numerical Heat Transfer and Fluid Flow", Intertext Books, McGraw Hill, New York, 1980.
4. Janes D., "An Intelligent Control System for CFD Modelling Software", Ph.D. Thesis (in preparation), The University of Greenwich, CMS, 2001.
5. 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.
6. 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, no 1, pp 13-27, 1999.
7. 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.
8. Ewer J., Galea E.R., Patel M.K. and Knight B., "The Development and Application of Group Solvers In The Smartfire Fire Field Model", Interflam'99, vol 2, pp 939-950, 1999.
9. Galea E.R., Knight B., Patel M., Ewer J., Petridis M., and Taylor S., "SMARTFIRE V2.01 build 365, User Guide and Technical Manual", SMARTFIRE CD, 1999.
10. 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.
11. "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.
12. Kumar S., Gupta A.K. and Cox G., "Effects of Thermal Radiation on the Fluid Dynamics of Compartment Fires", Fire Safety Science - Proc. of the Third Intl. Symp., pp 345-354, 1991.
13. Raithby G.D., Chui E.H., "A finite volume method for predicting a radiant heat transfer in enclosures with participating media", Journal of Heat Transfer, vol 112, pp 415-423, May 1990.
14. Lewis M.J., Moss M.B. and Rubini P.A., "CFD Modelling of Combustion and Heat Transfer in Compartment Fires", Fire Safety Science, Proc. of the 5th Int. Symp., pp 463-474, 1997.
15. www: http://fseg.gre.ac.uk/
16. Ewer J., Galea E.R., Knight B., Patel M., Janes D., Petridis M., "Fire Field Modelling using the SMARTFIRE Automated Dynamic Solution Control Environment", CMS Press, Paper Number 98/IM/41, ISBN 1899991387, London, 1998.
17. 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, July 2000.