INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2007; **55**:29–39
Published online 19 June 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1278

Parallel CFD fire modelling on office PCs with dynamic load balancing

A. J. Grandison*, †, E. R. Galea, M. K. Patel and J. Ewer

Fire Safety Engineering Group, University of Greenwich, London, U.K.

SUMMARY

Parallel processing techniques have been used in the past to provide high performance computing resources for activities such as Computational Fluid Dynamics. This is normally achieved using specialized hardware and software, the expense of which would be difficult to justify for many fire engineering practices. In this paper, we demonstrate how typical office-based PCs attached to a local area network have the potential to offer the benefits of parallel processing with minimal costs associated with the purchase of additional hardware or software. A dynamic load balancing scheme was devised to allow the effective use of the software on heterogeneous PC networks. This scheme ensured that the impact between the parallel processing task and other computer users on the network was minimized thus allowing practical parallel processing within a conventional office environment. Copyright © 2006 John Wiley & Sons, Ltd.

Received 4 January 2006; Revised 29 March 2006; Accepted 20 April 2006

KEY WORDS: CFD; fire modelling; parallel processing; PC; MPI; dynamic load balancing

INTRODUCTION

Fire simulation based on computational fluid dynamics (CFD) [1–3] has been underway for approximately 25 years. Despite the growing appeal of CFD fire modelling, there are several inhibitors which slow the wide scale adoption of the technique. While each of the specific technological inhibitors are rather different, in the eyes of the user, they all ultimately produce the same common problem, the costs associated with undertaking CFD-based fire simulations. These issues have many manifestations but usually involve the difficulty associated with, and the time required to;

*Correspondence to: A. J. Grandison, Fire Safety Engineering Group, University of Greenwich, London, U.K.
†E-mail: ga02@gre.ac.uk

Contract/grant sponsor: UK CAA

set up complex geometries; the expertise level required by the engineer to correctly interpret, set up and coax the software to a solution and to interpret the results generated (skills not normally associated with fire engineering); and finally the computer power and time required to run the software. Each of these factors will impact the costs associated with producing a final design solution and hence the viability of the approach as a routine design tool. In this paper, the excessive computational resource required to perform CFD fire simulations is addressed.

To a certain extent, the need for vast amounts of computational power is partially mitigated by the ever improving cost-performance of desktop PCs. However, as with the provision of higher capacity motorways (highways), providing more capacity encourages greater usage resulting in capacity never being able to satisfy demand. In fire engineering this means that the more computing resources are provided, the more complex or more refined will be the models produced by engineers and hence the greater the demands placed on the performance capabilities of the computer platform. Thus, simply relying on computer manufacturers to produce faster computers will not meet the demands of the fire modelling community. However, parallel processing has the potential to meet this computational demand. It should also be noted that in the future with the advent of dual-core and multi-core processors currently being developed by Intel and AMD, software will need to be 'parallel' to take advantage of the additional computational power.

In the past, parallel processing techniques have successfully been applied to CFD-based fire modelling simulations [4–6]. However, these implementations were based on expensive, specialized, parallel processing equipment, unlikely to be attractive to most fire engineering practices. Modern day parallel processing is generally based around connecting commodity PC parts using dedicated networking, e.g. Beowulf clusters [7]. In comparison, relatively little attention has been given to harnessing the computational power of conventional Microsoft^{TM} WINDOWS PCs attached to a local area network (LAN). Although inferior in performance to a dedicated Beowulf style cluster, this computing resource is widely available and likely to already exist within the offices of most engineers. Furthermore, it is likely that considerable processing power is available within most engineering offices where desktop PCs remain idle or underutilized for large parts of the day and evening. Using parallel computing techniques, this wasted capacity could be harnessed to perform CFD simulations.

One example available in the literature utilizing typical PC equipment for parallel processing was performed by Law and Turnock [8]. Their work clearly demonstrated the potential computational power available from non-specialized equipment. Here the development of a parallel version of the CFD fire simulation software SMARTFIRE [9–11] is described with particular emphasis on testing of the dynamic load balancing (DLB) scheme operating in conventional office conditions.

OVERVIEW OF SMARTFIRE

SMARTFIRE [12, 13] is an open architecture CFD environment written in C++ that is comprised of four major components: the CFD numerical engine, graphical user interfaces (GUIs), an automated meshing tool and an intelligent control system.

In fire field modelling, the fluid is governed by a set of three-dimensional partial differential equations. This set consists of the continuity equation, the momentum equations in three space dimensions, the energy equation, the user equations for mass and mixture fraction, and the equations for the turbulence model, in this case the k–ε model which incorporates buoyancy modification.

DOI: 10.1002/fld

The generalized governing equation for all variables is expressed in the following form:

$$\frac{\partial \rho \Phi}{\partial t} + \mathrm{div}(\rho \mathbf{U} \Phi) = \mathrm{div}(\Gamma_\Phi \nabla \Phi) + S_\Phi \tag{1}$$

where Φ represents the fluid variable; ρ and $\mathbf{U}$ are the local density and velocity vector; Γ_Φ is the effective exchange coefficient of Φ; S_Φ represents the source term for the corresponding variable Φ and time t is an independent variable.

The SMARTFIRE code will not be discussed in detail here as it has been described many times, interested readers are referred to Reference [13] for more details. Here we will only briefly describe the CFD engine. The CFD engine has many additional physics features that are required for fire field modelling [1, 2]. These include models for radiation, combustion and conjugate heat transfer through walls. For radiation, the user can select from either a six-flux radiation model [14] or a multiple ray radiation model [15]. The software can represent the combustion process either through a simple volumetric heat and smoke release model or a more complex gaseous combustion model (using the eddy dissipation model) [16]. The main flow solver algorithm makes use of the SIMPLE [17] and SIMPLEC [18] iterative solvers and can solve problems on a fully unstructured mesh.

The CFD engine is the only component of the software that would benefit from parallelization as the other software components are highly interactive in nature and do not require excessive computation. Thus, the parallelization process focuses on the CFD engine of the software.

PARALLEL IMPLEMENTATION

In order to make the parallel version of the software flexible and more likely to meet the requirements of fire engineers, the parallel implementation was designed to fulfil the following criteria:

- There should be no difference between the input or output files for the serial and parallel implementations of the software. This will allow applications to be designed and the results visualized using the familiar serial components of the software.
- The parallel implementation should work on an arbitrary number of processors.
- Minimal additional investment (in time and hardware/software) should be required by the fire engineer to effectively run the software.
- The software must run effectively on both homogeneous and heterogeneous networked computers.
- The parallel implementation is intended to function in a Microsoft Windows environment.

An additional requirement imposed by the software developers was that there should only be one CFD source code i.e. separate parallel and serial CFD source codes would not be developed. Having a single CFD source code is desirable in order to minimize the effort required to maintain the product.

The parallel code was developed by modifying the serial source code. The source code uses a series of compiler directives and conditional statements to differentiate between serial and parallel source code. In addition, approximately 4000 lines of new code were required. Approximately 200 lines of code were needed for the parallel communication necessary during the computational

DOI: 10.1002/fld

phase of the code. Approximately 500 lines of code were required for the renumbering, load balancing and memory allocation processes. The rest of the amendment was primarily concerned with the I/O process associated with the geometry file, the restart file and other output files.

In the following section, details of the selected parallelisation strategy are described. It should however be noted that when using the parallel software, these details are hidden from the user. These details are presented here for completeness.

Parallelization strategy

The parallel strategy used for the CFD code is based on a systematic partitioning of the problem domain onto an arbitrary number of sub-domains and is similar to the methods used by other workers [4–6, 19, 20]. This is known as domain decomposition. Each sub-domain is computed on a separate processor and runs its own copy of the CFD code. At the boundary of the domain partitions, each sub-domain needs to communicate with its neighbouring sub-domain to maintain the data dependencies that exist within the serial code. This communication was implemented using the freely available MPI parallel library produced by Argonne National Laboratory (MPICH) [21]. In the work presented here, a simple one-dimensional (1D) partitioning scheme was used. More advanced partitioning schemes, such as Jostle [22] and Metis [23], have also been implemented within parallel SMARTFIRE but will not be discussed here.

Types of parallel network

There are essentially two types of parallel network that are available within a typical engineering office environment; homogeneous and heterogeneous networks. A *homogeneous* network of computers is composed of a number of 'identical' computers, i.e. computers with the same processor specification. Homogenous networks are commonly found in large corporate offices where the corporate IT strategy requires that computers are purchased and upgraded on a large contract. A *heterogeneous* network of computers is composed of a number of 'non-identical' computers with different processor specification. Heterogeneous networks are commonly found in smaller engineering office environments where computers are bought and upgraded depending on the requirements of a particular user on an *ad hoc* basis. In this work, all the heterogeneous computers used the same operating system, i.e. Microsoft WINDOWS.

Dynamic load balancing

To take advantage of a heterogeneous network of PCs, a DLB scheme was devised to determine the relative computational power of each processor. In addition, as parallel SMARTFIRE is designed to be used on a conventional LAN of PCs some provision for other users using the same PC on the LAN had to be provided. This was necessary for two reasons, first the entire parallel fire simulation would be held up if just a single computer within the network became busy with another process, and secondly the other user's process would be detrimentally affected by the parallel fire simulation. To mitigate both of these effects, the software monitors the workloads on all of its parallel computers. If a computer is found to have an additional computational load then the problem is redistributed among the processors so that only a small amount of processing, about 5% of the original load, is placed on the 'busy' processor by the parallel fire simulation. This ensures the majority of the 'busy' processor processing power (about 95%) is available to the

DOI: 10.1002/fld

user who is sitting in front of the computer. Furthermore, the parallel fire simulation is not held up by the 'busy' processor. The DLB strategy is described below:

(1) The problem is distributed among the processors involved using an initial calculated performance for each processor. This initial performance is calculated within the code by timing a small number of numerical tests on each processor within the network. The workload assigned to each processor is linearly proportional to the performance of each processor. The performance indices are normalized so that the maximum performance index is 1.
(2) The timestep is incremented and the code is run to solve the problem for this timestep.
(3) Special operating system interrogation commands are called at the end of the timestep from within the code to establish the $\%\mathrm{CPU}_{\mathrm{total}}$ (the total CPU power being utilized) and $\%\mathrm{CPU}_{\mathrm{SMARTFIRE}}$ (the CPU power being expended running SMARTFIRE) usage on each processor over the duration of the timestep. These would be both 100% if the processor was fully utilized running SMARTFIRE.
(4) Using these figures and the initial processor performance figures κ^{old}, the new processor performance indices are calculated using the following formula:

$$\kappa^{\mathrm{new}} = \frac{\kappa^{\mathrm{old}}}{\%\mathrm{CPU}_{\mathrm{total}}} \times 100 \tag{2}$$

This will give a reasonable estimate of the relative CPU performances assuming that the time to process a problem is linearly proportional to the number of cells. This is reasonable for large cell budgets. The processor performance indices are normalized by dividing by the maximum processor performance index.
(5) If $\%\mathrm{CPU}_{\mathrm{smartfire}}$ is less than 60% of $\%\mathrm{CPU}_{\mathrm{total}}$ for a particular processor then it is assumed that the processor is performing another computational task and should therefore be used minimally for the parallel process. The performance κ^{new} is set to 0.05 for the 'busy' processors. This means that only a very small amount of work is placed on the processor for the parallel task. This avoids the entire parallel job being held up by the busy processor and leaves the vast majority of the processor capacity to work on the other job(s) running on it.
(6) If all the new performance indices are less than 5% different from the old performance indices, then the simulation continues using the old problem distribution. The process then returns to step (2) until the simulation has finished.
(7) However, if any of the new performance indices are more than 5% different from the old performance indices then a restart file is created and the problem is restarted on a new domain decomposition by returning to step (2) and using these new processor performance values to determine the load balance on each processor.

PERFORMANCE OF THE PARALLEL IMPLEMENTATION

In previous publications by Grandison *et al.* [9, 10] and Grandison [11], the performance of the parallel implementation of SMARTFIRE on 'well behaved' networks was described. It was found that good speedups could be achieved on homogeneous and heterogeneous networks of PCs, for example a problem composed of ~100 000 cells would run 9.3 times faster on a network of

DOI: 10.1002/fld

12 800 MHz PCs then on a single 800 MHz PC. It was also found that a network of 8 3.2 GHz Pentium 4 PCs would run more than seven times faster than a single 3.2 GHz Pentium computer.

In the work presented below, the results of the parallel implementation operating in 'adverse' conditions are described. The adverse conditions occur when there is extra load on the parallel system which could take the form of additional network traffic or additional CPU usage on one (or more) of the processing nodes.

Fire test cases

Two test cases were used for testing the parallel implementation of SMARTFIRE on the adverse network conditions. The nature of the test cases and the results generated by the software are not of primary concern in this paper and so the cases will only be briefly described here. All the cases involved fires within an enclosure that was vented to the outside and possessed an extended region beyond the external vent to accurately capture the vent flow.

Case 1. This test case arises from a fire test conducted by the Loss Prevention Council (LPC) [24]. The test involved a burning wood crib within an enclosure with a single vent. The test compartment had a floor area of 6 m × 4 m and a 3.3 m high ceiling. The compartment contained a doorway (vent) measuring 1.0 m × 1.8 m located on the rear 6 m × 3.3 m wall. The walls and ceiling of the compartment were made of fire-resistant board (asbestos) which were 0.1 m thick. The floor was made of concrete. The case was discretized into $(31 \times 24 \times 35)$ 26 040 cells. Further test details and model results can be found in Reference [25].

Case 2. This case was an artificial test case designed for timing and performance assessment. It composed of $(61 \times 39 \times 43)$ 102 297 cells and represented the maximum cell budget that could be run on a serial computer with a 256 Mb RAM. This case consisted of a simple fire (measuring 0.5 m × 0.5 m × 0.5 m) located within a compartment measuring 3 m × 3 m plan area with a height of 2 m. A single vent measuring 1 m × 1 m was located in one of the walls. The walls and ceiling of the compartment were composed of a common brick material, which allows turbulent heat transfer, and the floor was considered to be non-conducting. The fire was represented by a simple volumetric enthalpy (heat) source with a constant heat output of 50 kW.

Fire simulation results

A detailed analysis of the fire simulation results will not be given in this paper as the intention of this work is to demonstrate the speedup of the parallel implementation of the CFD fire model. Detailed analysis of SMARTFIRE predictions can be found in a variety of other publications (see for example References [25, 26]). However, it is important to note that the results generated by the standard serial version of the software and the parallel implementation in its various network configurations are virtually identical for all the cases examined in this paper.

The maximum difference in the two predictions was found to be less than $10^{-4}\%$. This was found to be typical of all the cases examined and is consistent with the convergence criteria (10^{-4}) used.

DOI: 10.1002/fld

Table I. Effect of network usage on parallel processing network.

Tests	Case 1	Case 2
No extra network traffic	138 ± 1 s	135 ± 1 s
Test 1	138 ± 1 s	135 ± 1 s
Test 2	158 ± 3 s	156 ± 2 s
Test 3	239 ± 3 s	241 ± 4 s

Effect of network usage with no DLB

Several tests were conducted to see the effect of additional network traffic on the performance of the parallel code. The tests were carried out on three 733 MHz Pentium III computers connected to a 100 Mbps Ethernet which were running parallel SMARTFIRE while one of the following three tests was performed:

(1) Browsing the internet (primarily the BBC news site (news.bbc.co.uk)) on one of the machines (<5 Mbytes).
(2) Downloading a large file from the internet onto one of the machines ($\sim$ 50 Mbytes).
(3) Uploading a large directory structure via the intranet onto remote disk storage ($\sim$ 500 Mbytes).

These tests were repeated five times each for the two different cases previously described. Case 1 utilized 1 time step and 50 outer iterations, while case 1 utilized 1 time step and 20 outer iterations. These settings were chosen to ensure that the time taken for the simulations was shorter than the period required for downloading test 2 and uploading test 3. This ensured the maximum detrimental network effect on the simulation runtimes.

While browsing the Internet had no measurable effect on the performance of parallel SMARTFIRE, it can be seen from Table I that downloading a 50 Mbytes file from the Internet added approximately 20 s to the overall runtime and uploading a directory structure via the Ethernet added approximately 100 s to both the simulations considered. These results must be considered in the context of a true simulation that may take several hours. The additional 100 s in this context has minimal impact on the overall runtime and an engineer is generally unlikely to be using the network to move around such large amounts of data. A far greater impact on the runtime will be caused by the engineer running extra computationally intensive jobs on one or more of the computers involved in the parallel processing.

Effect of additional computational load with no DLB

Using the same configuration that was used above one of the nodes was selected to run an additional computational load. The additional computational load is serial SMARTFIRE running the same case but with no interaction with the parallel case.

It can be seen from Table II that running the additional computational loads had an adverse effect on the overall runtime of a fire model prediction. Unlike the effect of the network traffic, the effect of the additional computational load could possibly last the duration of the parallel simulation undertaken leading to runtimes increasing by a factor of $\sim$3 in this case and completely removing the advantages of parallel processing. With more processors in the parallel network the problem

DOI: 10.1002/fld

Table II. Effect of additional computational load on parallel processing network.

Test	Case 1	Case 2
No additional load	138 ± 1 s	135 ± 1 s
Additional load	333 ± 19 s	355 ± 14 s

Table III. Results of testing of DLB with an additional serial computational load.

	2 Processing nodes		3 Processing nodes	
Performance index	Case 1	Case 2	Case 1	Case 2
Ideal	2.0	2.0	3.0	3.0
Actual (P)	1.91	1.95	2.67	2.82
Ideal with load (P)	1.05	1.05	2.05	2.05
Actual with load (P)	0.85 ± 0.1	1.02 ± 0.02	1.61 ± 0.13	1.80 ± 0.05
Ideal with load (S)	0.95	0.95	0.95	0.95
Actual with load (S)	0.93 ± 0.01	0.92 ± 0.01	0.94 ± 0.01	0.93 ± 0.01
Combined (P + S)	1.78 ± 0.09	1.95 ± 0.01	2.56 ± 0.12	2.73 ± 0.04

is compounded as all the processors are limited by the performance of the slowest processor e.g. if a parallel network had a speedup of 12 this could be reduced to a speedup of 4.

Testing of load balancing scheme with additional computational load

In a real office environment it is possible that one or more of the machines that an engineer chooses to use as part of their parallel processing system may be used by other engineers or users in that office at some point during the simulation. The parallel implementation needs to be capable to cope with such an event. As previously described, the performance of the busy processor is reduced to 5% of the maximum processor performance to ensure that the local user of the computer has a large amount of processing power available to them.

Two network configurations were used to test the ability of the DLB scheme to react to an additional serial load being placed on one of the processors. The first network configuration was composed of two Pentium III 733 MHz processors. The second configuration was composed of three Pentium III 733 MHz processors. Both of these configurations were connected via a 100 Mbps Ethernet. The testing has been conducted on homogeneous networks to demonstrate the load balancing scheme but does not preclude the possibility of using the DLB with an additional load on a heterogeneous network. The parallel cases ran for a couple of time steps and then the additional serial load would be started on one of the processors.

When one of the processors had an additional serial load placed upon it, the new speedups for the processors were calculated using the load balancing algorithm. The performance of the processor with the additionally serial load was reduced to 0.05. In the networks tested, this left 95% of the processing power available for the additional serial job on the additionally loaded processor. The additional serial computational load was an instance of serial SMARTFIRE.

DOI: 10.1002/fld

The test results are summarized in Table III. (P) is the parallel performance and (S) is the performance of the additional serial load, SMARTFIRE.

It can be seen from Table III that the effect of the small parallel load on the processor with the additional serial computational load was quite small with a 6–8% loss of serial performance compared to a processor with no parallel load. The user of the remote computer used for the parallel processing should hardly notice the small parallel load.

The final row (Combined) of Table III gives an indication of the total amount, of computational power extracted from the processors on the parallel network. From Table III it can be seen that the overall processing power extracted from the network was only slightly less than the processing performance extracted from a plain parallel load.

Intermediate changeover behaviour

The worst performance of both the serial and the parallel job occurs during the transition phase when the additional serial job is started. The transition phase occurs before rebalancing can be performed to take account of the additional serial load. The performance of both the parallel job and the additional serial job will be compromised by using the wrong load balance and will behave similarly to that obtained by running a serial and parallel case without DLB (Table II). This results in the parallel performance being approximately 25–50% of the anticipated parallel performance. The performance of the additional serial load is approximately 50–70% of the normal serial performance. This problem is relatively short lived and should generally only last for a single time step. In some circumstances it may last for two time steps.

When the additional serial job finishes then the parallel performance is slightly compromised as only 5% of the maximum processing power of the processor, which also has the additional serial load, is used. This is worst for two processors where only about half the total available power is used. This problem gets progressively better as more processors are used. Theoretically for a set of n homogeneous processors the parallel speedup is $(n - 0.95)$. This problem is only short lived and should generally last for a single time step. In some circumstances it may last for two time steps. However, this effect is far less severe than the effect of starting an additional serial load as described in the previous paragraph.

The effect of the changeover behaviour could be alleviated by checking the processor performance indices more frequently and therefore allowing a problem partition rebalancing to be performed earlier. However, this process incurs its own time penalty and may be activated prematurely for short-lived additional serial loads. It is difficult to determine an optimal strategy for choosing when to rebalance the problem partition. The strategy adopted here, i.e. checking processor usage and if required rebalancing the computational load at the end of each time step, seems to perform reasonably well.

CONCLUDING COMMENTS

In this paper, we have attempted to address the excessive computational resource required to perform CFD fire simulations through the use of parallel computing techniques. However, unlike standard approaches used in parallel computing which make use of specialist hardware and software, a main aim of this work was to achieve an efficient parallel implementation using equipment typically found

DOI: 10.1002/fld

in an engineering office, namely standard commodity PC computers using Microsoft WINDOWS connected using a standard LAN.

The fire simulation software SMARTFIRE was parallelized using the domain decomposition technique and it has been previously demonstrated [9–11] that good speedups can be achieved for a moderate number of computers (<12).

The robustness of the parallelization was demonstrated by its ability to deliver good speedups on a heterogeneous/homogeneous network of PCs using a DLB scheme. This scheme helps ensure that engineers utilizing an individual PC within the parallel network would not be unduly affected by the parallel fire simulation and vice versa.

A potential difficulty in utilizing standard office PCs connected via a LAN for parallel processing concerns the risk of individual computers failing or other users inadvertently electing to switch off or reboot computers while they are being used for parallel processing. To address this problem a fault-tolerant version of the parallel implementation is being developed. In the first instance this will be based on a check pointing method which regularly saves restart data and automatically restarts the parallel processing job by assessing which machines are 'safe' to operate on.

By harnessing the power of standard office-based PCs in a parallel network, CFD-based fire modelling can be made more attractive to fire engineering practices. Tapping into under-utilized computer power that already exists within their offices, will allow fire engineers to tackle large fire simulations in more practical time-scales at almost no additional cost. Furthermore, this technology could be easily applied to other CFD-based application.

ACKNOWLEDGEMENTS

Professor E. R. Galea is indebted to the UK CAA for their financial support of his personal chair in Mathematical Modelling at the University of Greenwich.

REFERENCES

1. Galea ER. On the field modelling approach to the simulation of enclosure fires. *Journal of Fire Protection Engineering* 1989; **1**(1):11–22.
2. Cox G (ed.). *Combustion Fundamentals of Fire*. Academic Press: New York, 1995.
3. Waters RA. Stansted terminal building and early atrium studies. *Journal of Fire Protection Engineering* 1989; **1**(1):63–76.
4. Galea ER, Ierotheou C. Fire field modelling on parallel computers. *Fire Safety Journal* 1992; **19**(4):251–266.
5. Ierotheou C, Galea ER. A fire-field model implemented in a parallel computing environment. *International Journal for Numerical Methods in Fluids* 1992; **14**:175–187.
6. Galea ER, Ierotheou C. A parallel implementation of a general purpose fluid flow code and its application to fire field modelling. In *Proceedings of the Parallel Computing*, Joubert, Evans, Liddel (eds). Elsevier Press: Amsterdam, 1991, 1993.
7. Sterling T, Becker D, Savarese D *et al*. BEOWULF: a parallel workstation for scientific computation. *Proceedings of the 1995 International Conference on Parallel Processing* (*ICPP*), vol. 1, August 1995; 11–14.
8. Law RA, Turnock SR. Utilising existing computational resources to create a commodity PC network suitable for fast CFD computation. *Parallel Computational Fluid Dynamics—Trends and Applications*, Jensson CB (ed.). Elsevier Science: New York, 2001; 115–122 (ISBN 0 444 50673 X).
9. Grandison AJ, Galea ER, Patel MK, Ewer J. Parallel CFD based fire modelling on conventional office based PCs. *Proceedings of the Joint DCABES and ICPACE Meeting*, CMS Press: University of Greenwich, U.K., 2005; 43–46 (ISBN 1-904521-27-4).
10. Grandison AJ, Galea ER, Patel MK, Ewer J. The development of parallel implementation for a CFD based fire field model utilising conventional office based PCs. *Journal of Applied Fire Science* 2003–2004; **12**(2):137–157.

DOI: 10.1002/fld

11. Grandison AJ. Improving the regulatory acceptance and numerical performance of CFD based fire-modelling software. *Ph.D. Thesis*, University of Greenwich, December 2003.
12. Ewer J, Galea ER, Patel MK, Taylor S, Knight B, Petridis M. SMARTFIRE: an intelligent CFD based fire model. *Fire Protection Engineering* 1999; **10**(1):13–27.
13. Ewer J, Jia F, Grandison A, Galea E, Patel M. *SMARTFIRE V4.0 User Guide and Technical Manual*, University of Greenwich, 2004.
14. Hoffman N, Markatos NC. Thermal radiation effects on fires in enclosures. *Applied Mathematics Modelling* 1988; **12**:129–140.
15. Raithby GD, Chui EH. A finite volume method for predicting a radiant heat transfer in enclosures with participating media. *Journal of Heat Transfer* 1990; **112**:415–423.
16. Lewis MJ, Moss MB, Rubini PA. CFD modelling of combustion and heat transfer in compartment fires. *Fire Safety Science*, *Proceedings of the 5th International Symposium*, 1997; 463–474.
17. Patankar S. *Numerical Heat Transfer and Fluid Flow*. Intertext Books, McGraw-Hill: New York, 1980.
18. Van Doormal JP, Raithby GD. Enhancements of the SIMPLE method for predicting incompressible fluid flows. *Numerical Heat Transfer* 1984; **7**:147–163.
19. Baltas ND, Spalding DB. MIMD PHOENICS: porting a computational fluid dynamics application to a distributed memory MIMD computer. *Massively Parallel Processing Applications and Development*, Dekker L *et al*. (eds). Elsevier: Amsterdam, 1994.
20. McGratten K, Bouldin C. Simulating the fires in the world trade center. *Proceedings Interflam 2004*, Interscience, New York, 2004; 999–1008.
21. Gropp W, Lusk E, Skjellum A. *Using MPI*. MIT Press: Cambridge, MA, U.S.A., 1999.
22. Walshaw C. A parallelisable algorithm for optimising unstructured mesh partitions. *Technical Report P95/IM/03*, School of Computing and Mathematical Science, University of Greenwich, U.K., January 1995.
23. Karypis G, Kumar V. Multilevel *k*-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing* 1998; **48**(1):96–129.
24. Glocking JLD, Annable K, Campbell SC. Fire spread in multi-storey buildings—fire break out from heavyweight unglazed curtain wall system—run 007. *LPC Laboratories Report TE 88932-43*, 25 February 1997.
25. Grandison AJ, Galea ER, Patel MK. Development of standards for fire field models—report on SMARTFIRE Phase 2 simulations. FRD Publication 1/2003, Fire Research Division, ODPM (Office of the Deputy Prime Minister), U.K., 2002.
26. Zhang J, Ewer J, Jia F, Grandison A, Galea E. *SMARTFIRE v4.0*: *SMARTFIRE Verification and Validation Report*, University of Greenwich, U.K., 2004.

DOI: 10.1002/fld