Abstract

This paper shows how parallel processing may improve the computational efficiency of the transient dynamic non-linear analysis of reinforced concrete plates subjected to blast loading. The results presented in the paper are based on a parallel scheme for a time marching procedure using the explicit Newmark's algorithm. It is shown that very high computational efficiency may be obtained by decomposing the finite element mesh into a number of sub-domains for distributed analysis on multiple processors. Through examples it is demonstrated how this efficiency depends on the problem size (*i.e.* the level of refinement of the problem idealisation), the number of sub-domains and the status of the analysis with regards to the states of the material.

Keywords: parallel processing, non-linear analysis, reinforced-concrete, plates

1 Introduction

Application of parallel and distributed computing to finite element analysis requires the development of parallel algorithms [1]. For these algorithms to be applied effectively then considerable pre-processing of the finite element model is required to ensure that the analysis is efficiently undertaken in parallel [1].

The present paper discusses a parallelisation scheme of the program for the temporal analysis of reinforced concrete plates subjected to severe dynamic loads. This program utilises an explicit Newmark iteration scheme [2, 3] as a temporal discretisation of the finite element formulation with simple six noded triangular facet elements which account for a complex non-linear material model [4] represented in a layered manner [5]. The example runs show very high speed-ups and efficiency results which proves the importance of using parallelisation for efficient solution of the problem. The paper also demonstrates the virtues of the domain decomposition tools developed

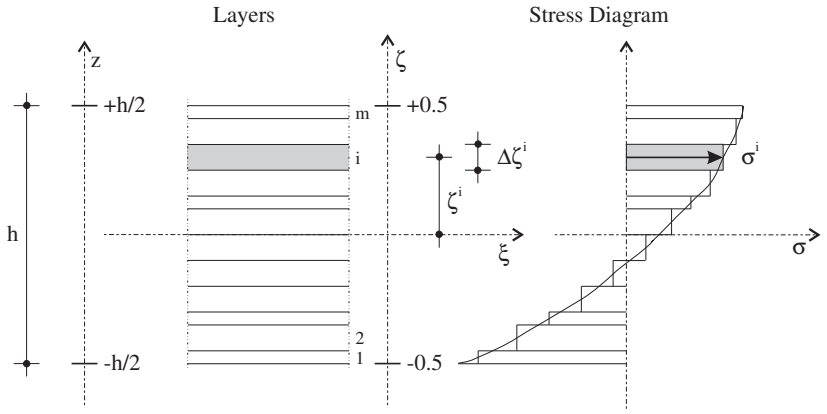

Figure 1: A layered model and the corresponding stress representation

by the authors [1, 6, 7] for the pre-processing of the finite element mesh.

2 Sequential Non-linear Finite Element Analysis

For the solution of this transient engineering problem a finite element spatial discretisation has been used to establish dynamic equilibrium equation for the structure. The resulting time dependent sets of differential equations were further discretised in time to obtain the numerical solution. An elasto-viscoplastic constitutive relationship for the concrete and steel is adopted and built into the discretised system. Without considering all the alternatives the theoretical base of the program is briefly discussed in the following sections. This is part of a larger project concerned with non-linear finite element analysis [8].

In the program a flat triangular element, which was originally developed for linearly elastic plates is used in the temporal analysis. This element is obtained by combining the constant strain plane stress element and the constant moment plate bending element (due to Morley [9]) to form the simplest facet element.

The element *generalised stresses* (or in this case the stress resultants) $\{\sigma_g\}$ are obtained by integrating the corresponding stress components with respect to the thickness co-ordinate, then they may be used to determine the internal element forces $\{q\}$ for the temporal integration:

$$\{q\} = \{q(d, v)\} = \int_A [B]^T \{\sigma_g\} dA \tag{1}$$

For further details of the evaluation of Equation (1) the reader should consult reference [2]. The global internal resisting force vector $\{N\}$ may then be calculated by summing the contributions from all the elements in the standard manner for each of the global degrees of freedom:

$$N(d,v) = \sum_{i=1}^{nelem} \{q^i\} \tag{2}$$

The material model used is as described in reference [8].

3 Parallel Non-linear Finite Element Analysis

3.1 Parallelisation Scheme

The scheme is a special type of parallelisation called *geometric parallelism*. With this type of scheme each processor executes the same code on data corresponding to a sub-region or sub-domain of the whole structure being simulated or processed. Inter-processor communication is possible to permit the exchange of boundary data between processors representing connections between neighbouring sub-domains. This is an *explicit domain decomposition* approaches [1].

3.2 Pre-processing

When considering the scheme of geometric parallelism to finite element analysis the idealisation has to be sub-divided into a number of sub-domains. Each of these sub-domains will be subjected to the same instruction set by the worker tasks concurrently on each processor. To make the parallel analysis procedure efficient the following conditions have to be considered when creating the partitions:

- The processors must all complete their task at the same time if processors are not to be left idle waiting for the others to finish. That means the computational load has to be balanced between the processors which are part of the parallel system. In the case of a homogeneous network of processors (*i.e.* identical hardware nodes) the number of finite elements have to be equal to the number of sub-domains. This is based on the assumption that the computational load is equal for each of the finite elements. This would be the case most of the time, especially at the beginning of a non-linear analysis procedure. If the computational load is not equal for each element then some weighting facility has to be implemented.

- The physical partitions should be made such that interprocessor communication is minimised. Parallel machines usually have relatively slow communication capacity between the computing elements compared to their processing capabilities. During the solution procedure the amount of inter-processor communication has to be minimised. As it was discussed in Section 3.1, in the case of an explicit dynamic finite element technique the amount of this communication is proportional to the sum of the degrees of freedom of the finite element

nodes which lie on the inter-sub-domain boundary. Thus to make the parallel analysis procedure faster and more efficient the number of these nodes must be minimised.

These two conditions define a special optimisation problem which can be solved in a number of ways [1]. It is not particularly difficult when the finite element mesh is regular. For the present work the meshes were created using a parallel adaptive unstructured mesh generator described in References [1, 11]. The decomposition of the generated irregular meshes were done using the Enhanced Sub-domain Generation Method (ESGM) detailed in Reference [6].

3.3 Parallel Algorithm

The parallel algorithm for the analysis program is well defined by the parallelisation scheme described in Section 3.1. The master task reads in the description of the finite element idealisation, *i.e.* the mesh definition and the material specifications. The pre-processing then prepares and delivers the sub-domains to the worker tasks with the information they need concerning their connectivity. This procedure described in detail in Reference [6].

For the non-linear analysis each worker task executes the same program code but on different sub-domains of the problem. The difference between this code and the sequential version is detailed as follows. The actual instructions to carry out the steps of the analysis are the same as the finite element sub-domains are stored within the program in a same structured way as the original domain.

Those components of the vectors of masses, external loads and internal forces, *i.e.* $\{M\}$, $\{F^{ext}\}$ and $\{N\}$ respectively, which correspond to nodes on the boundary interface between adjacent sub-domains have to be exchanged between these sub-domains. This exchange communication is the same for all of these three vectors with the difference that for the vector of masses, $\{M\}$, the exchange is done only once at the beginning of the analysis after the calculation of the element mass contributions have been carried out. The masses are constants throughout the iteration procedure and hence only have to be calculated once. For the vectors of external loads and internal forces, *i.e.* $\{F^{ext}\}$ and $\{N\}$ the same communication has to be repeated in each iteration step when calculating the residuals. To carry out this communication a function has been written in Parallel C [10] which is used in the assembly of all three vectors.

At the end of each iteration step the energy balance check is carried out. But in a distributed environment the energy values for the whole system have to be calculated by summing up all the contributions of each sub-domain. But the fact that the inter sub-domain nodes are represented within more than one sub-domains at the same time their contribution to the total energy would be duplicated. To avoid this once in the beginning of the program, in the pre-processing phase the worker tasks determine from which sub-domain each inter-sub-domain node will 'contribute' to the energy

values. The worker tasks do not therefore consider all the degrees of freedom of their sub-domain. In the summation the shared nodes are added only if it is appropriate. Determining the sub-domain contributions to the energy calculations in such a way, ensures that the energy values that are sent to the master task, form the total values. The master task now is in the position of being able to carry out the energy balance check and reports its result to the users terminal.

On completion of the iteration all the worker tasks send their results which include stresses, strains, displacements, *etc.* to the master task which then assembles the messages received into results corresponding to the whole structure and saves these to disk. Naturally these results are numerically identical to those of the sequential program, but they are achieved in considerably shorter time.

3.3.1 Communication Routine

With respect to their communication requirements all the finite element nodes held within one sub-domain are divided into three groups: to those which appear in one other sub-domain, these are referred to as P-nodes; the ones which appear in more than two sub-domains, are referred to as M-nodes; and the ordinary nodes which are not common with any other sub-domain. Figure 2 describes this classification. This classification was neccessary to ensure that the message passing in the program is efficient. If the domain consists of ordinary and P-nodes alone then the message passing would only be between pairs of sub-domains. For M-nodes the communication is more complex; the relevant values have to be collected from each participating sub-domain and then the sum of these values have to be re-distributed back to all the connected sub-domains. Thus the overall communication scheme is divided into two stages; first the P-communication deals with all the P-nodes then the M-communication collects all the values for the remaining, usually very few, M-nodes.

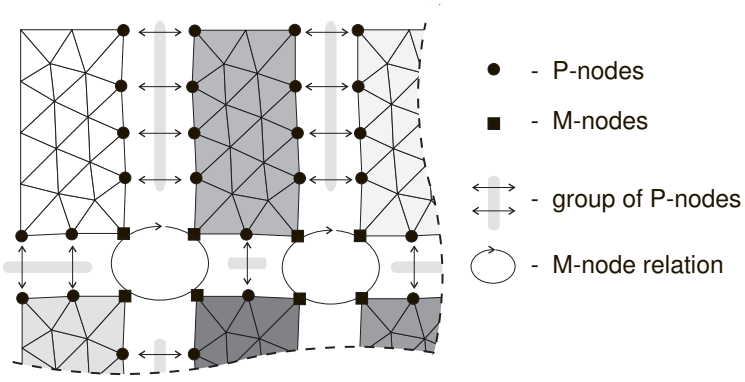

Figure 2: The different type of FE nodes with respect to communication

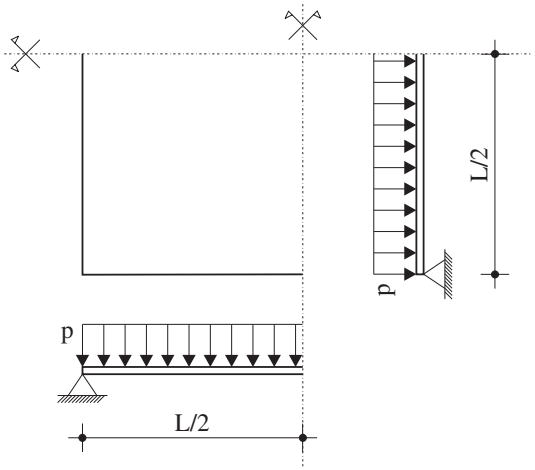

Figure 3: The quarter of the microconcrete plate model

4 Example

To test the result of the parallelisation a blast load experiment published in Section 4.1 of Reference [12] as *Test 1: Bare Panel Subjected to Uniform Blast Loading* was simulated. In this experiment a thin microconcrete slab, 391mm square and 15.2mm thick, was supported on the edges of an open steel box (sunk into the ground) so that the upper face of the slab was flush with the ground. The corners were held down with clamps attached to the box, and the unsupported span was 366mm. The lower face of the slab was reinforced to a level of 0.75% each way with 1.63mm wires at 22.1mm centres. The quarter of the plate model is shown in Figure 3. The microconcrete panel with concrete compressive strength of approximately 28MPa was subjected to peak blast pressure loading of 138kPa.

By considering the symmetrical conditions, a quarter of the plate model was idealised using 208 triangular finite elements. Each element comprised thirteen layers of which two represented the two orthogonal sets of reinforcement bars. This mesh is referred to as the 'A2' model. This original idealisation was decomposed into two subdomains of 104 elements, four sub-domains of 52 elements and eight sub-domains of 26 elements. The idealisation is shown in Figure 4 with the three decompositions illustrated. Both the 'A2' and 'A3' decompositions were undertaken using the enhanced subdomain generation method [6].

The plate was re-meshed using the 280 of the same triangular finite elements. This mesh is referred to as the 'A3' model. This idealisation was decomposed into two subdomains of 140 elements, four sub-domains of 70 elements and eight sub-domains of 35 elements. The idealisation is shown in Figure 5 with the three decompositions illustrated.

The numerical analysis was carried out for a small part of the simulation process using the original sequential version of the program and the execution time was mea-

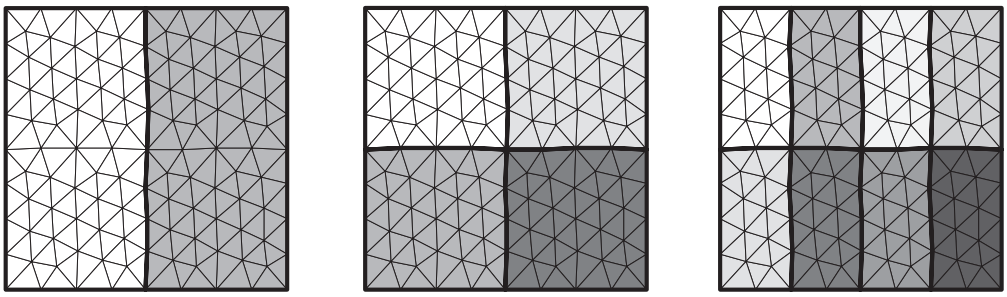

Figure 4: The 'A2' finite element idealisation divided into two, four and eight sub-domains

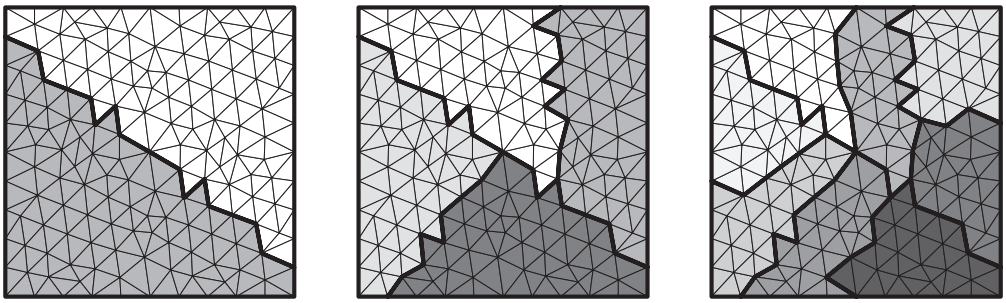

Figure 5: The 'A3' finite element idealisation divided into two, four and eight sub-domains

sured, t_{seq}. Subsequently the parallel program executed the same number of iterations on the decomposed meshes. The parallel execution times, t_{par}, for the three different decompositions were measured and the speed-ups, S, and efficiencies E calculated as described in Reference [1]. In the case of this parallel program the number of processors are always equal to the number of worker tasks thus to the number of sub-domains.

Table 1 shows the execution times, the speed-ups and the parallel efficiencies achieved by running the program for 25 iterations as a benchmark for the 'A2' mesh and Table 2 shows the same for mesh 'A3'. These timing results do not include the time required for the pre-processing of the finite element mesh (*i.e.* the domain decomposition $\approx 20 - 30$sec), the preparation of sub-domain connectivity data for the communication routine ($\approx$ 1sec), the delivery of the sub-domains to the actual processing nodes ($\approx 2 - 3$sec) nor the setting up time before the iteration commences ($\approx$ 1sec). The number of iterations required to run the non-linear simulation for a sufficient time in real terms ($\approx 2 - 3$msec), is large because of the very small time steps ($\Delta t \approx 10^{-6}$sec). The number of iterations is usually in the $10^3 - 10^5$ range, rendering the execution of the program very long. Compared to this, the amount of time spent on the pre-processing of the finite element mesh and the setting-up of the calculations is negligible.

Num. of Proccessors N	exec. time t_{par}	Speed-up S	Efficiency E
1 - Sequential run	257s (=t_{seq})	n/a	n/a
2	168.9	1.53	76.4%
4	84.3	3.05	76.2%
8	42.3	6.07	75.9%

Table 1: Speed-up values and efficiencies for the parallel system using mesh 'A2' based on 25 (linear) iterations

Num. of Proccessors N	exec. time t_{par}	Speed-up S	Efficiency E
1 - Sequential run	346s (=t_{seq})	n/a	n/a
2	223.5	1.55	77.4%
4	112.4	3.08	77.0%
8	56.8	6.09	76.1%

Table 2: Speed-up values and efficiencies for the parallel system using mesh 'A3' based on 25 (linear) iterations

Table 3 summarises the nodal connectivity between the sub-domains for both 'A2' and 'A3' idealisations. This table list the number of P and M nodes which determine the communication load on the parallel system.

Figure 6 shows the parallel efficiencies of the 'A2' mesh in the initial part of the iteration measured on 25 time step iterations (as given in Table 1). The material is mainly in a linear state here. As the iteration continues more and more layers of the finite elements start cracking and are in a non-linear material state. The cracked layers represent a much higher computational load than the un-cracked ones. The fact that in certain sub-domains there are many more cracked layers than within other ones means that there is an extra computational load in those sub-domains. This decreases the parallel efficiency because the extra computation causes a computational in-balance within the processor network. As the cracked and non-linear area propagates, thus the number of cracked layers becomes almost equal in each sub-domain, the parallel efficiency is regained because the sub-domains become computationally balanced again. This process is shown in Figure 7 for the same 'A2' mesh sampled with 25 time steps using eight sub-domains, where the variation of the parallel efficiency of the processor network is plotted against the number of iterations. This current efficiency was calculated on the basis of the computational performance during the immediate previous 25 iterations. The other graph which is plotted on the same co-ordinate system is the standard deviation (SD) of the number of cracked layers in each sub-domain at the displayed iteration number. This graph is calculated using the usual simple formula:

Idealisation	'A2'			'A3'		
No. sub-domains	2	4	8	2	4	8
No. P nodes	17	32	62	33	65	98
No. d.o.f.	35	64	122	67	129	192
No. M nodes	0	1	3	0	2	5
No. d.o.f.	0	3	9	0	6	15
No. other nodes	432	416	384	568	534	498
No. d.o.f.	656	624	560	856	788	716
Total No. nodes	449			601		
Total No. d.o.f.	691			923		
Total No. elements	208			280		
No. elements/sub-domain	104	52	26	140	70	35

Table 3: The summary of nodal connecetivity between the sub-domains for the 'A2' and 'A3' meshes

$$SD = \sqrt{\overline{nc^2} - \overline{nc}^2} \tag{3}$$

where nc is the number of cracked layers in a subdomain. Despite the fact that the efficiency is measured on a relatively wide (25 step) interval of iterations and the deviation of the number of cracked layers correspond to the actual iteration number, the correlation between the two graph is easily recognised.

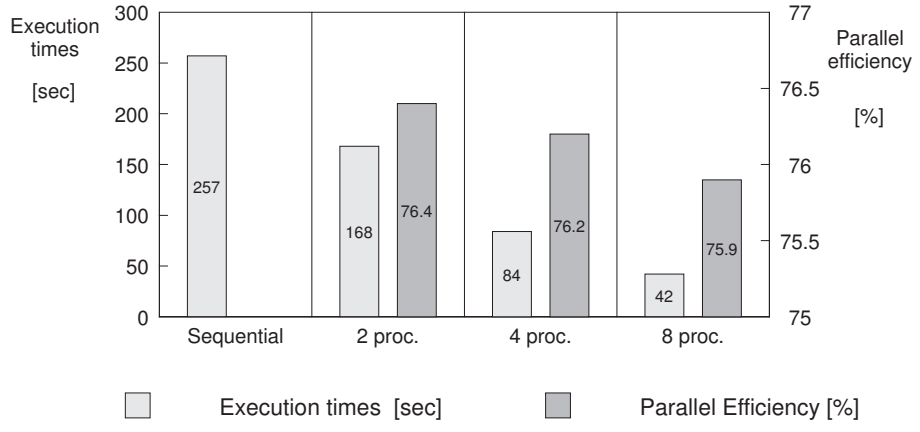

Figure 6: The chart of execution times and efficiencies for the 'A2' FE mesh

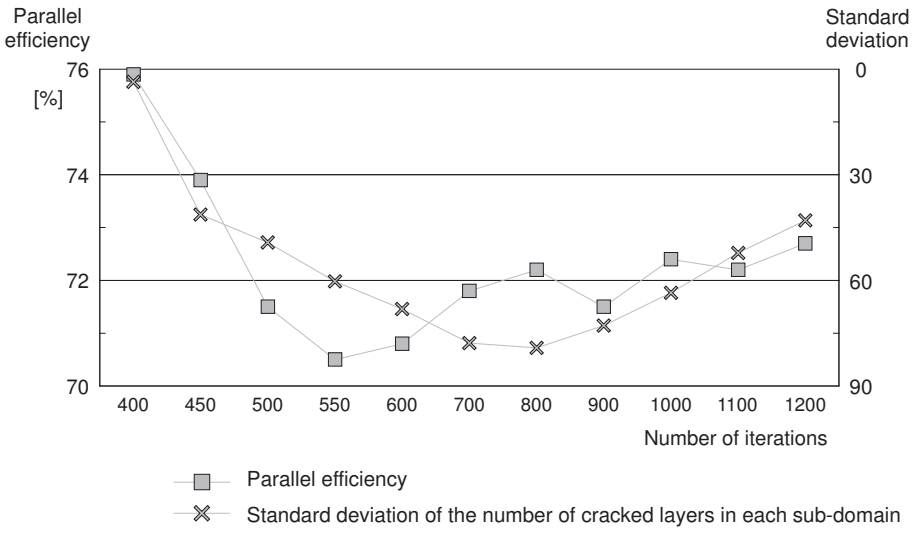

Figure 7: The variation of the parallel efficiency for the 'A2' FE mesh throughout the iteration (eight sub-domains)

5 Conclusion

The timing results presented in Tables 1 and 2 show that the parallelisation of the given algorithm with the presented parallelisation scheme is very efficient. The parallel program runs almost as many times faster as the number of processor employed. This is a consequence of the complexity of the problem being solved, the special structure of the explicit solution scheme and that the amount of communication overhead is relatively small.

The efficiency results in Figure 6 show that the parallel communication scheme is well organised since with an increased communication requirement the parallel program did not loose significant efficiency. This demonstrates that the communication scheme has been implemented making efficient use of the given hardware with its limitations. It has to be noted here too, that although the communication requirements are increasing in Figure 6 as the number of processors are increasing, but the special layout of the hardware connectivity handles this increase in a distributed way, rather than having to go through only one connection link. This contributes to keep the efficiency at almost the same level.

It can also be seen from Figure 6 that even though there is a decrease in efficiency if the decomposition is not properly balanced this decrease is still smaller than the disturbance caused by the non-linear material states later in the iteration as displayed in Figure 7.

As a conclusion of these examples it may be stated that for the optimisation of the initial domain decomposition it is more important to have the load balance right than

to minimise the inter sub-domain connections. The progressing non-linear areas will destroy the load balance so the most important factor is to have some dynamic load balancing procedure supporting the parallel analysis. This would correct the problems of the initial domain decomposition (if there are any) and would account for the developing in-balances resulting from non-linearity. Thus an efficient dynamic load balancing algorithm would also decrease the importance of the initial mesh partitioning technique.

The parallelisation of this algorithm is a good example that in the area of transient non-linear finite element analysis parallel techniques will have a significant and increasing role in the future.

References

[1] B.H.V. Topping, A.I. Khan, *"Parallel Finite Element Computations"*, Saxe-Coburg Publications, Edinburgh, UK, 1996.

[2] T.J.R. Hughes, T. Belytschko, *"Nonlinear Finite Element Analysis"*, Course Notes, Paris, France, 1994.

[3] N.M. Newmark, *"A Method of Computation for Structural Dynamics"*, Journal of the Engineering Mechanics Division, ASCE, USA, July, 1959.

[4] F.B.A. Beshara, *"Non-linear Finite Element Analysis of Reinforced Concrete Structures Subjected to Blast Loading"*, PhD Thesis, City University, London, April, 1991.

[5] E. Hinton, D.R.J. Owen, *"Finite Element Software for Plates and Shells"*, Pineridge Press, Swansea, UK, 1984.

[6] J. Sziveri, C. Seale, B.H.V. Topping, *"Enhanced Parallel Sub-domain Generation"*, submitted for publication, 1997.

[7] B.H.V. Topping, J. Sziveri, *"Parallel Sub-domain Generation Method"*, *in* Developments in Computational Techniques for Structural Engineering, B.H.V. Topping, (Editor), 449-457, Civil-Comp Press, Edinburgh, UK, 1995.

[8] J. Sziveri, B.H.V. Topping, *"Report on the Program JM1 for the Dynamic Non-linear Finite Element Analysis of Reinforced Concrete Plates Subject to Blast Loading"*, Technical Report to BRE, Heriot-Watt University, Edinburgh, UK, 1996.

[9] L.S.D. Morley, *"On the Constant Moment Plate Bending Element"*, Journal of Strain Analysis, 6, 20-24, 1971.

[10] *"Parallel C User Guide (compiler version 2.2.4), Reference Manual"*, 3L Ltd., Livingston, UK, 1991.

[11] Khan, A.I., Topping, B.H.V., *"Parallel Adaptive Mesh Generation"*, Computing Systems in Engineering, 2(1), 75-102, 1991.

[12] B.J. Broadhouse, A.J. Neilson, *"Modelling Reinforced Concrete Structures in DYNA3D"*, Safety and Engineering Division, United Kingdom Atomic Energy Authority, Winfrith, UK, AEEW-M 2465, 1987.