INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 27, 179–192 (1998)

FINITE ELEMENT ADAPTIVE MESH ANALYSIS USING A CLUSTER OF WORKSTATIONS

K. P. WANG AND J. C. BRUCH JR.*

Department of Mechanical and Environmental Engineering, University of California, Santa Barbara, CA 93106–5070, U.S.A.

SUMMARY

Parallel computation on clusters of workstations is becoming one of the major trends in the study of parallel computations, because of their high computing speed, cost effectiveness and scalability. This paper presents studies of using a cluster of workstations for the finite element adaptive mesh analysis of a free surface seepage problem. A parallel algorithm proven to be simple to implement and efficient is used to perform the analysis. A network of workstations is used as the hardware of a parallel system. Two parallel software packages, P4 and PVM (parallel virtual machine), are used to handle communications among networked workstations. Computational issues to be discussed are domain decomposition, load balancing, and communication time.

Int. J. Numer. Meth. Fluids, **27**, 179–192 (1998)

KEY WORDS: parallel computing; cluster of workstations; domain decomposition; load balancing; free surface seepage; finite element adaptive meshes

1. INTRODUCTION

For the past few years, with the advanced technology in the computer industry, workstations have been produced with high computing speed and low cost. Because of their high computing speed, cost effectiveness and scalability, parallel computation on clusters of workstations is becoming one of the major trends in the study of parallel computations. Also, with the availability of easily accessible parallel software packages to create a parallel virtual machine, researchers who are unable to access parallel computers can create one with their resident workstations, thus expanding their computing power.

This paper is organized such that each aspect necessary to make the parallel computations feasible on a cluster of workstations will be discussed separately with appropriate references. Thus the next section presents the free boundary seepage problem that is used as the test case. The following section discusses the adaptive mesh finite element scheme used. Then ways of performing domain decomposition necessary for the parallel computations will be analysed, followed by the numerical algorithm to be used in the parallel environment. Finally a short discussion of the two popular parallel software packages P4 and PVM will be presented before results of using all these elements are given.

*Correspondence to: J. C. Bruch, Jr., Department of Mechanical and Environmental Engineering, University of California, Santa Barbara, CA 93106-5070, U.S.A.

Contract grant sponsor: National Science Foundation; Contract grant number: ECS-9006516

CCC 0271–2091/98/010179–14 $17.50 *Received May 1996*

2. FREE BOUNDARY PROBLEM

For the numerical implementation the free surface seepage problem to be studied is shown in Figure 1. Because of the assumptions made, the problem is described by the velocity potential function φ, whose governing differential equation and boundary conditions are also shown in Figure 1. The relevant dimensions are taken to be $x_1 = 40$, $y_1 = 10$ and $y_2 = 3$. In Figure 1, Ω is the seepage region 'abdf'. The location of the curve $\widehat{fd}$, $y = \bar{f}(x)$, is unknown *a priori*.

A fixed domain formulation for this problem can be obtained by using the Baiocchi method and transformation. Since the early 1970s these have been used on a large number of seepage problems.[1–5] In this approach the *a priori* unknown solution region is extended across the free surface into a known region. The dependent variable is also continuously, similarly extended. Then a new dependent variable is defined using Baiocchi's transformation within these regions. The resulting problem formulation leads to a 'complementarity system' associated with its respective variational or quasi-variational inequality formulation. This method has proven effective not only from the purely theoretical point of view but also from the point of view of yielding new, simple and efficient numerical solution schemes.

Figure 2 shows the governing equations and boundary conditions that describe the fixed domain formulation of the problem presented in Figure 1. D is the region 'abef' The variable w is the Baiocchi transformation of the extended potential function, i.e.

$$w(x, y) = \int_y^{y_1} [\tilde{\varphi}(x, \bar{\eta}) - \bar{\eta}] \, \mathrm{d}\bar{\eta}, \tag{1}$$

where

$$\tilde{\varphi}(x, y) = \varphi(x, y) \quad \text{in } \bar{\Omega}, \qquad \tilde{\varphi}(x, y) = y \quad \text{in } \bar{D} - \bar{\Omega}. \tag{2}$$

The detailed derivations of these equations are given in Reference 4.

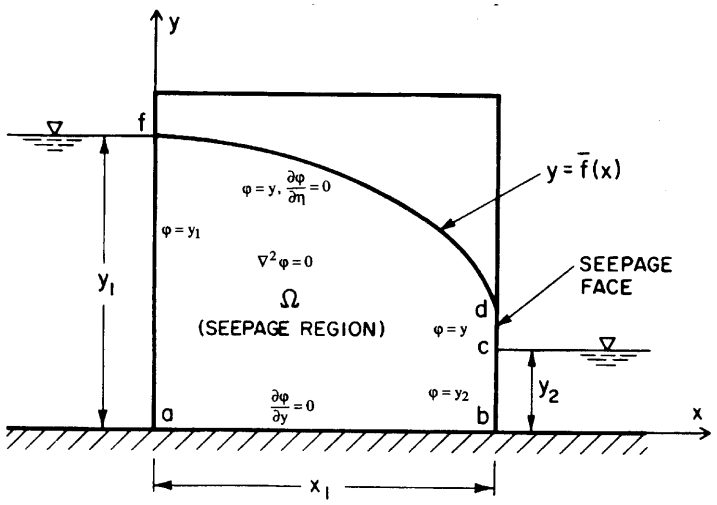


Figure 1. Example physical problem (free boundary seepage) for numerical implementation

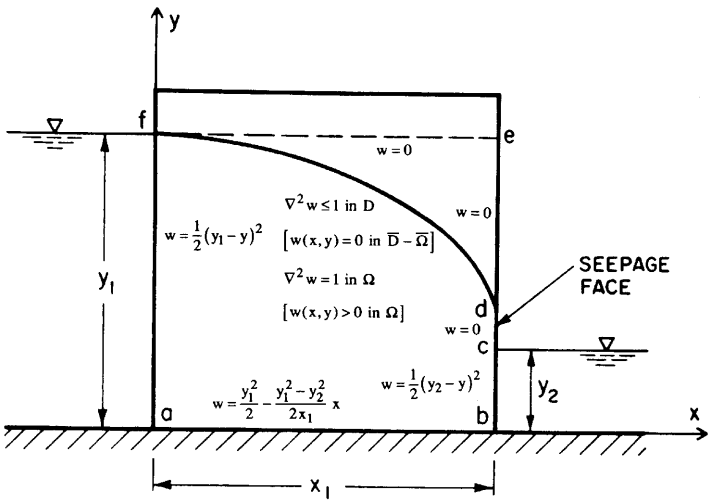


Figure 2. Fixed domain formulation for example physical problem

The problem shown in Figure 2 can be written as a 'complementarity system' and its corresponding variational inequality formulation. Then the following theorem can be stated. Let $\bar{p}(x, y)$ be the Dirichlet data in Figure 2 and define

$$K = \{v(x, y) | v \in H^1(D),\ v|_{\partial D} = \bar{p},\ v \geqslant 0 \text{ a.e. on } D\},$$

a closed convex set, $K \subset H^1(D)$.

Theorem

If $w \in K$ satisfies the governing equations and boundary conditions shown in Figure 2, then it also satisfies the variational inequality

$$a(w, v - w) \geqslant L(v - w) \quad \forall v \in K, \tag{3}$$

where

$$a(w, v - w) = \iint_D \nabla w \cdot \nabla(v - w)\, \mathrm{d}x\, \mathrm{d}y = \iint_D [w_x(v_x - w_x) + w_y(v_y - w_y)]\, \mathrm{d}x\, \mathrm{d}y, \tag{4}$$

$$L(v - w) = -\iint_D (v - w)\, \mathrm{d}x\, \mathrm{d}y. \tag{5}$$

The finding of $w \in K$ is equivalent to solving the minimization problem

$$J(w) \leqslant J(v) \quad \forall v \in K, \tag{6}$$

where

$$J(v) = a(v, v) + 2(f, v), \tag{7}$$

in which $a(v, v)$ is a bilinear form, continuous, symmetric, positive definite on $\mathbb{R}$ and $f \in \mathbb{R}$, i.e.

$$a(v, v) = \iint_D \nabla v \cdot \nabla v \, dx \, dy, \tag{8}$$

$$(f, v) = \iint_D f v \, dx \, dy. \tag{9}$$

For this example problem, $f = 1$. The functional J has one and only one minimum in a closed convex set.

The minimum is found using the finite element algorithm

$$u_i^{(n+1/2)} = -\frac{1}{a_{ii}} \left(\sum_{j=1}^{i-1} a_{ij} u_j^{(n+1)} + \sum_{j=i+1}^{N} a_{ij} u_j^{(n)} + f_i \right), \tag{10}$$

$$u_i^{(n+1)} = P_i(u_i^{(n)} + \alpha(u_i^{(n+1/2)} - u_i^{(n)})) = \max(0, u_i^{(n)} + \alpha(u_i^{(n+1/2)} - u_i^{(n)})), \tag{11}$$

where $a_{ij} = a(N_i, N_j)$, $f_i = (f, N_i)$, N_i is the canonical basis of $\mathbb{R}^N$, P_i is the projection on the convex set, $i = 1, \ldots, N$, N is the number of nodal points and α is the relaxation factor. Linear triangular elements will be used in the discretization. It should be noted that the projection operation in the numerical scheme must be applied during the iteration process. It cannot be applied after the iteration process has been completed, since if it were, an incorrect solution would be obtained. For the numerical results given herein, the SOR relaxation factor was 1·85, while the stopping error criterion was 10^{-4} for the maximum absolute difference between iterates at a mesh point.

3. ADAPTIVE MESH FINITE ELEMENT ANALYSIS

Error estimation and local mesh refinement are two major concepts of adaptive mesh finite element analysis. In addition, a mesh refinement algorithm is required to perform remeshing after obtaining the error of a finite element system. The error estimate decides how the computed results deviate from the exact solution. Local mesh refinement testing is performed to determine how the mesh is to be refined. The remeshing algorithm is then used to automatically generate a refined mesh according to the error obtained.

The error estimation procedure used herein was introduced by Zienkiewicz and Zhu.[6] It allows an accurate assessment of errors while remaining so simple that it can readily be implemented as a postprocessor involving minimal computation. This computationally simple error estimator with the modification introduced by Burkley and Bruch[7] and Burkley *et al.*[8] is used in the existing finite element code for the solution of a free surface flow through an earth dam using a parallel computer. The modification used by Burkley and Bruch[7] was that, instead of using the Zienkiewicz–Zhu procedure (a projection method) to calculate the nodal estimates for the exact nodal fluxes, they performed a simple averaging technique. That is, for each node they added up, say, the x-fluxes (which were constants since linear elements were used) from the elements that contained that node and divided that sum by the number of contributing elements. The same was done for the y-fluxes. They also made one other minor change in the nature of the denominator for the calculation of the predicted value of the relative percentage error. A mesh generator and a mesh refiner were used to perform the finite element analysis and the postprocessing of the mesh refinement. Isosceles right triangle elements were used for the purpose of this study. A simple mesh generator and refiner for these elements is implemented to generate the initial mesh. The concept behind the mesh generation and mesh refinement is simple: divide an element into two by refining across its longest side.

In this study, incompatible elements are not allowed. Therefore a recursive process proposed by Rivara[9,10] is used to avoid having such elements. For the refining process, if the longest side of a refining element (target element) is shared with another element (neighbour element), the neighbour element must be refined first. However, once the shared side is not the longest side of the neighbour element, the neighbour element is refined first into two elements, then the new element sharing the same longest side with the target element is refined. Since the same situation may happen to the neighbour element, the refinement process will be propagated until no element shares the longest side of the element being refined, or the shared side is the longest side for both elements. Therefore refining an element may cause a recursive refinement in the neighbourhood of that element. Since triangles are being divided in half, this procedure will cause elements to be similar triangles without there being any incompatible elements.

Accordingly, to create an initial mesh block, we start with two large triangular elements in a square or rectangle. Then these two large elements can be subdivided (refined) into four smaller triangular elements. Thus a mesh can be obtained by repeating this process until the area of every element is smaller than a common area criterion. Similarly, for the refinement of the mesh after the error estimate the same procedure is applied. In this stage the desired area for each element calculated from the error estimate process is used as the new area criterion for an element.

4. DOMAIN DECOMPOSITION

For the past decade, domain decomposition techniques have been studied intensively. When domain decomposition techniques are used, the complicated geometries can be dealt with in a simpler manner. Also, domain decomposition can be used to handle a system with different types of equations in different parts of the physical domain. Moreover, when a problem is solved on a parallel computer, domain decomposition techniques are used to obtain parallelism for the problem. The general concept to carry out the domain decomposition is to subdivide the domain of definition into a set of subdomains and then obtain solutions of related equations on each subdomain. These solutions of all the subdomains are then put together in some way to obtain the approximation of the whole computation domain.

A schematic diagram is shown in Figure 3 which demonstrates the early researchers who have theoretically dealt with domain decomposition. There are basically two types of domain decomposition techniques. Some domain decomposition techniques require overlap between two adjoining subdomains,while others do not. The two categories of problems highlighted in Figure 3 are those having all fixed or known boundaries or those having some free (or moving) or unknown boundaries. In the former category, there is the overlapping domain decomposition Schwarz

Boundary / Domains	Fixed	Free
Overlapping	"Classical" H.A. Schwarz (1843-1921) [11]	Bruch and Sloss (1985) [14]
Non-Overlapping	Funaro, Quarteroni, and Zanolli (1988) [12]	Papadopoulos, Sloss, and Bruch (1991) [15]

Figure 3. Schematic diagram of references to Schwarz methods and extensions

alternating method.[11] This approach starts with a guessed value of the dependent variable on one of the overlapping domain boundaries whose solution is then obtained in that domain. The other overlapping domain boundary now has values for the dependent variable along it since it is interior to the domain whose dependent variable values were just obtained. Using these boundary data, the values of the dependent variable in this respective domain are obtained. After this calculation, new updated values on the first overlapped boundary are considered. Again, values of the dependent variable are calculated for the domain having this boundary. This procedure is continued until convergence of the values of the dependent variable on the overlapped boundaries.

For fixed boundary problems, Funaro *et al.*[12] proposed a domain decomposition technique without overlapping boundaries. See Figure 4 for a schematic diagram of the iterative relaxation process for a problem whose governing differential equation was $\nabla^2\phi = f$ in D, the domain of solution, with $\phi = 0$ on ∂D. This problem is split into the two non-overlapping domains D_1 and D_2, shown with common boundary Γ and corresponding governing equations and boundary conditions. An interface relaxation factor θ is introduced to speed up the convergence of the iterative procedure. Iterative processes are then performed between the two adjacent subdomains by imposing on the interfaces the continuity of the solution on one side and the continuity of the normal derivative in a weak sense on the other side. At the limit of the convergence process, transmission conditions at the interface are therefore satisfied. Marini and Quarteroni[13] proved the convergence, for the two-subdomain case, of this iterative scheme for both the differential problem and its finite element approximation.

For an application of the Schwarz alternating procedure to free boundary value problems, Bruch and Sloss[14] demonstrate how these problems, too complicated for formulation as a variational inequality, are broken up into two problems on overlapping regions. On one region the problem is treated as an ordinary boundary value problem; on the second region the 'free boundary part' of the problem is reduced to a variational inequality. By solving the two problems successively, it is shown that under certain conditions the successive solutions converge to a single function that gives a solution of the original problem. They also present an application of their results to a free surface seepage problem.

Similarly to Funaro *et al.*,[12] Papadopoulos *et al.*[15] suggested a domain decomposition technique having two regions requiring no overlapping area to solve a free boundary seepage problem of a slanted-faced dam with a toe drain. In addition, Papadopoulos *et al.*[16] used techniques for domain decomposition on a problem involving a free surface which had more than two non-overlapping regions.

The various approaches to domain decomposition that have been presented have also been used to solve problems on parallel computers. For example, the parallel Schwarz alternating procedure (see references cited in Reference 17) equates the information on the overlapped region of two subdomains and solves each subdomain concurrently with data exchange between two adjacent subdomains. White[18] proposed several multisplitting techniques to divide the control domain into

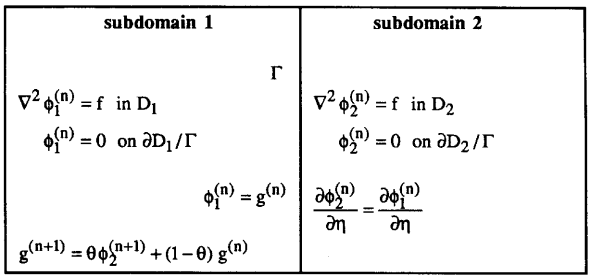


Figure 4. Schematic diagram of iterative relaxation process for non-overlapping method

vertical, horizontal or block subdomains with overlapped regions between adjacent subdomains. In addition, Rodrigue and Shah[17] imposed pseudo-boundary conditions to accelerate the parallel Schwarz method. Neumann and Dirichlet boundary conditions were imposed on each subdomain to ensure the continuity of the solutions. Combinations of these two boundary conditions were studied.

For fixed domain problems and using non-overlapping of two neighbouring subdomains, Farhat and Wilson[19] presented substructural concepts in finite element analysis (FEA) using concurrent processing. Their basic approach was the automatic splitting of an arbitrary spatial domain, with processors being dynamically reassigned during the several phases of an analysis. Cecchi *et al.*[20] present a domain decomposition method for shallow water equations. Their application is a finite element model of the linear shallow water equation model. Their method uses a non-overlapping approach by employing a generalized Schur-complement matrix for treating the subdomain interfaces.

In regard to solving free boundary value problems, using the overlapping domain decomposition approach on a parallel computer, Wang and Bruch[21] present results for a steady state free surface seepage problem which they obtained using a Hypercube concurrent computer. In addition, Hoffmann and Zhou[22] discuss the parallel solution of variational inequalities on a general closed convex set using the Schwarz-type decomposition method. They present algorithms which work for parallel computations on general closed convex sets. Then they investigate the parallel solution of linear complementarity systems related to variational inequalities. Finally they present numerical experiments solving a free surface seepage problem to show the convergence of the algorithms.

Herein a domain decomposition technique requiring no overlapping area, derived from that of Marini and Quarteroni,[13] is used for the free surface seepage problem previously discussed. As stated previously, Marini and Quarteroni[13] discuss fixed boundary problems and convergence of the numerical solutions. Papadopoulos *et al.*[15] provide a theorem for convergence of a numerical solution for a free boundary problem separated into two subdomains, one of which includes the complete free boundary. Although the assumptions are true in a physical sense, no mathematical proof of the convergence of the iterative procedures can be found for more than two non-overlapping subdomains.

5. PARALLEL ITERATIVE SCHEME

Wang and Bruch[23] proposed parallel iterative Gauss–Seidel and SOR iterative schemes. Conventional Gauss–Seidel and SOR iteration schemes need to be performed sequentially. Reordering the equations alters these two schemes into fully parallel iterative schemes. Wang and Bruch[23] used this intuitive idea and implemented it on a free boundary seepage problem. Speed-ups were obtained that were superlinear (speed-up larger than the number of processors used).

The basic essence of the approach is as follows: after the computation domain is subdivided into subdomains, the problem domain boundary remains a boundary and the interfaces of a subdomain become new boundaries. Thus the computation of values at interior mesh points for one subdomain is uncoupled from the other subdomains. Also, the computation of values at interface mesh points of an interface is uncoupled from the other interfaces. The iterative schemes use a combination of newly computed values and old values at mesh points surrounding a mesh point to compute the new value at that mesh point. Therefore the iterative process can be performed for the interior mesh points of a subdomain using the old values at interface mesh points. Moreover, the values at interface mesh points can be updated using the newly computed values at interior mesh points by the iterative process. An example and explanation are given in Reference 23.

Accordingly, this parallel iterative scheme simply reorders the computing sequence such that the values at interior mesh points are computed first, then those at the interface mesh points are computed. With this parallel scheme, all processors can compute concurrently for the new values at

the interior mesh points. Also, all processors update the values at mesh points on the interface in parallel.

6. PARALLEL SOFTWARE PACKAGES

P4[24] and PVM[25] are message-passing libraries for a cluster of workstations and parallel computers. With P4 or PVM a cluster of workstations can be used as if it were a single parallel computing resource.

P4 was developed at Argonne National Laboratory. It is a library of macros and subroutines for programming a variety of parallel machines. It is intended to be portable, simple to install and use, and efficient. It can be used to programme networks of workstations, advanced message-passing parallel computers and single shared-memory multiprocessors. The version of P4 used in this study is 1.4.

PVM (parallel virtual machine) was developed at Oak Ridge National Laboratory. It is a software package that allows a heterogeneous network of parallel and serial computers to appear as a single concurrent computational resource. It consists of two parts: (i) a daemon process which any user can install on a machine; (ii) a user library which allows initiating processes on other machines, communicating between processes and changing the configuration of machines. The version of PVM that is used is 3.3.4.

7. RESULTS AND DISCUSSION

Wang and Bruch[26] present studies of using a cluster of workstations for finite difference and finite element analyses of the same free surface seepage problem which is analysed herein. A parallel algorithm proven to be simple to implement and efficient was used for both analyses. They used the two popular parallel software packages P4 and PVM to handle the communications among the networked workstations. Also used for comparison purposes were the Paragon and Meiko CS-2 computers. Furthermore, they give an approach to develop a portable parallel code.

Their results showed that even though they were able to obtain superlinear speed-up using a Paragon or Meiko CS-2 computer, when they used P4 and PVM on a cluster of workstations the speed-up was less than the parallel computer speed-ups. The reasons for this were that the performance was limited by the speed of the network devices and the communication management. They performed their numerical experiments with the number of degrees of freedom reaching as high as 8353 for their finite element analysis.

The results presented herein will also use the P4 and PVM software on the same cluster of workstations (seven SGI Indy workstations running the Irix 5.1 operating system) for adaptive mesh finite element analyses where the number of mesh points increases as the accuracy of the solution sought is improved. Also, two methods of applying the adaptive mesh technique are used. In one the global desired solution error is the same for each adaptive run, while in the second the desired error criterion is decreased for each following run.

In the first set of experiments the desired percentage error was held at 0·05. The finite element analysis and mesh refinement were run through several passes until the calculated percentage error was less than the desired percentage error. In the second set of experiments the desired percentage errors were given adaptively. Again the adaptive finite element analysis and mesh refinement were run through several passes until a desired calculated percentage error was obtained. To develop portable parallel programmes as stated in Reference 26, common Paragon NX interface libraries were developed and used in the implementation. The parallel finite element analysis programme needed only to re-link with the desired interface library to be portable for different parallel systems. In these

experiments it was not the intention to compare P4 and PVM. Different timing results were obtained because of the different ways of implementing the common interface libraries.

Table I shows the results for the first set of experiments. The first set of experiments started with 85 nodes in the computation domain. After three passes of finite element analysis and mesh refinement the number of nodes increased to 1587. The final calculated percentage error was 0·0351. Table II shows results for the second set of experiments. The second set of experiments started with the same number of nodes in the computation domain but only had 788 nodes when the calculated percentage error was less than 0·05. The second set of experiments had four passes to show the effectiveness of using a cluster of workstations for parallel finite element analysis. Tables III and IV show the results and parameters of using four workstations. The only place there is a difference in the parameters for P4 and PVM is in the maximum SOR time. The PVM results for this line are in parentheses.

Table I. P4 (PVM) parameters and results (first set of experiments)

No. of processors	1	1	1
Relaxation factor	1·85	1·85	1·85
No. of nodes/processor	85	612	1587
No. of iterations	69	85	103
Max. SOR time	38 (1550)	2511 (4904)	28871 (37100)
Desired % error	0·05	0·05	0·05
Calculated % error	0·1704	0·0571	0·0351

Table II. P4 (PVM) parameters and results (second set of experiments)

No. of processors	1	1	1	1
Relaxation factor	1·85	1·85	1·85	1·85
No. of nodes/processor	85	261	788	1829
No. of iterations	69	80	85	109
Max. SOR time	38 (930)	449 (1459)	4534 (5590)	36888 (38192)
Desired % error	0·10	0·06	0·045	0·03
Calculated % error	0·1704	0·0839	0·0429	0·0323

Table III. P4 (PVM) parameters and results (first set of experiments)

No. of processors	4	4	4
Relaxation factor	1.85	1.85	1.85
No. of nodes/processor	25	164	415
No. of iterations	71	85	98
Max. SOR time	693 (5295)	1225 (7751)	5253 (15971)
Desired % error	0.05	0.05	0.05
Calculated % error	0·1704	0·0571	0·0351

Table IV. P4 (PVM) parameters and results (second set of experiments)

No. of processors	4	4	4	4
Relaxation factor	1·85	1·85	1·85	1·85
No. of nodes/processor	25	72	210	478
No. of iterations	71	77	83	107
Max. SOR time	553 (5289)	643 (5941)	1800 (8492)	6757 (14933)
Desired % error	0·10	0·06	0·045	0·03
Calculated % error	0·1704	0·0839	0·0429	0·0323

Figures 5a and 5b show results from the P4 implementations. Figure 5a shows the timing results for the first set of experiments. When the number of nodes in the computation domain was small, there was no need for parallel computation, since the time for communication overhead may be much more than the computation time. When the number of nodes increased, the computation time also increased. The speed-up is measured by

$$\text{speed-up} = T_1/T_p, \tag{12}$$

where T_p is the SOR solution time used by the slowest of the p workstations and T_1 is the time used by one workstation. When at the third pass the number of nodes was 1587, superlinear speed-up

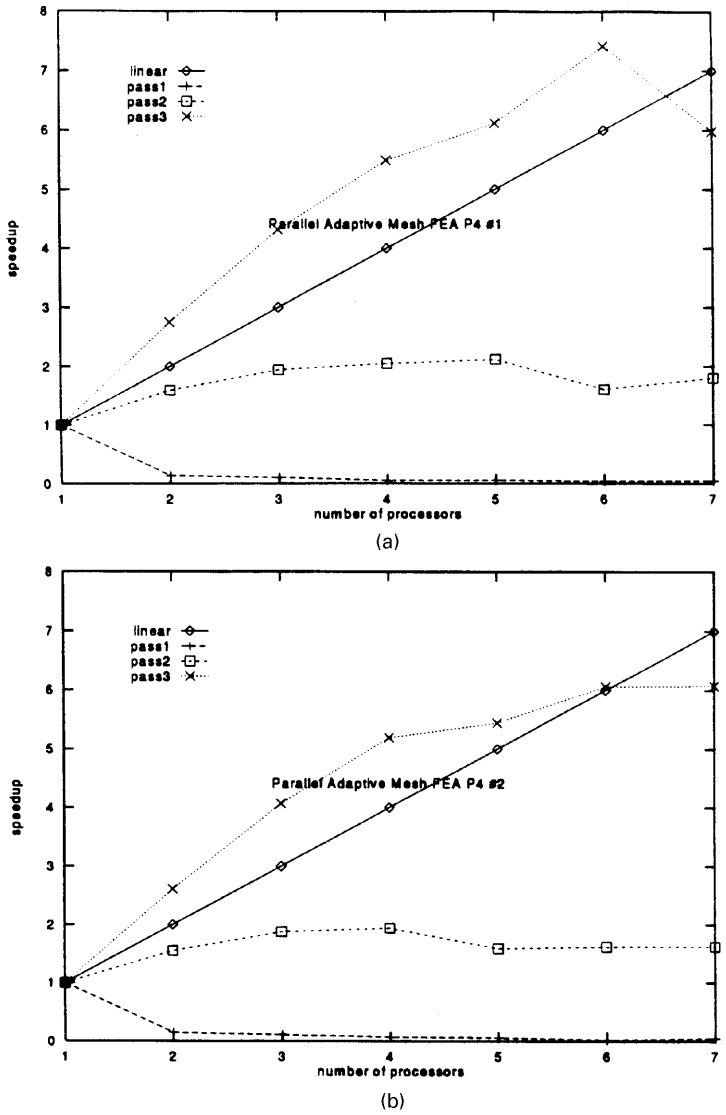


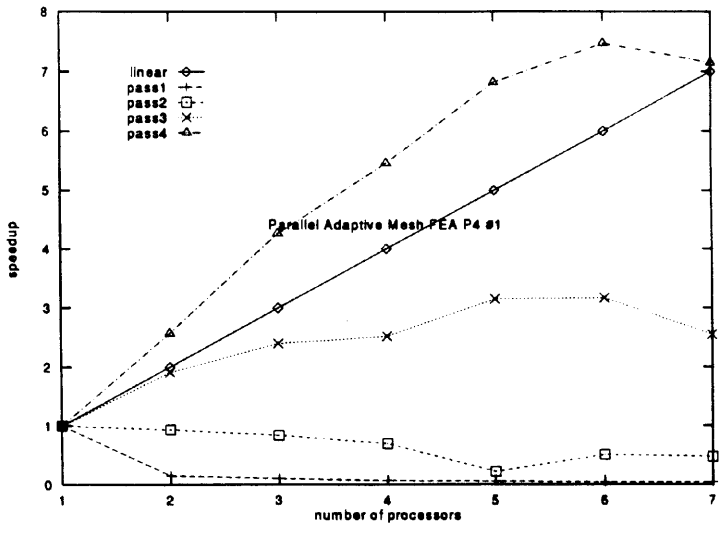
Figure 5. Speed-ups for parallel adaptive mesh finite element analysis using P4 and constant fixed desired percentage error

(speed-up larger than the number of workstations used) was obtained. The superlinear speed-up may be due to the following.

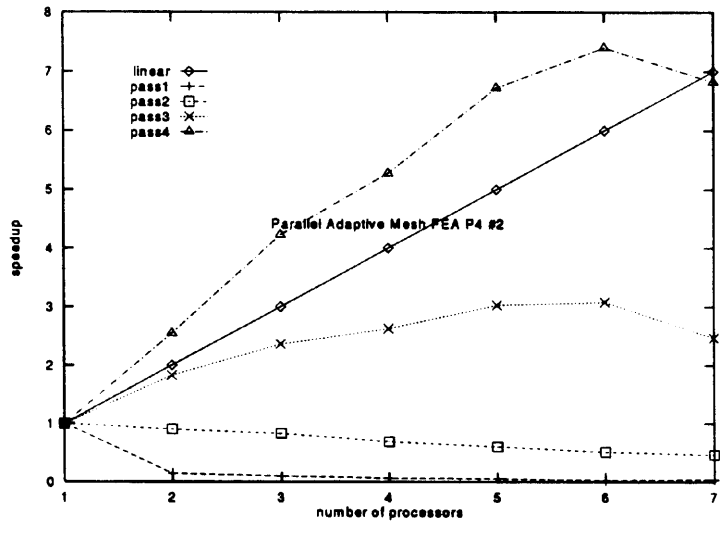
1. The memory required for each workstation was less and there was less swapping of memory required.
2. The SOR solver converged with fewer iterations.

Figure 5b shows another set of results, using P4, for the first set of experiments. Somewhat different speed-ups were obtained. The reason for the different timing results is that there was an unexpected user logged on one of the workstations in use for the parallel computation.

Figures 6a and 6b show the speed-ups for the second set of experiments. In these two figures,



(a)



(b)

Figure 6. Speed-ups for parallel adaptive mesh finite element analysis using P4 and variable desired percentage error

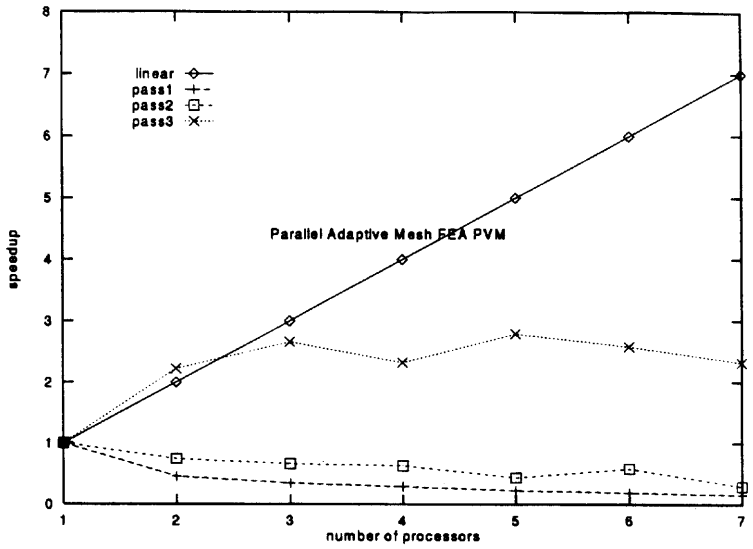


Figure 7. Speed-ups for parallel adaptive mesh finite element analysis using PVM and constant fixed desired percentage error

superlinear speed-ups were also obtained when there were 1829 nodes in the computation domain. When there were only 788 nodes in the computation domain, the speed-ups were 1·8 for the run using two workstations and 2·6 for the run using three workstations. Results from these two figures were consistent, since there were no unexpected users logged on any machine in use.

Figures 7 and 8 show PVM implementation results for only one set of runs. Although the speed-ups are not as good as those from P4 implementations, faster solutions than using only one workstation were still obtained. Inconsistent results due to unexpected users were also observed.

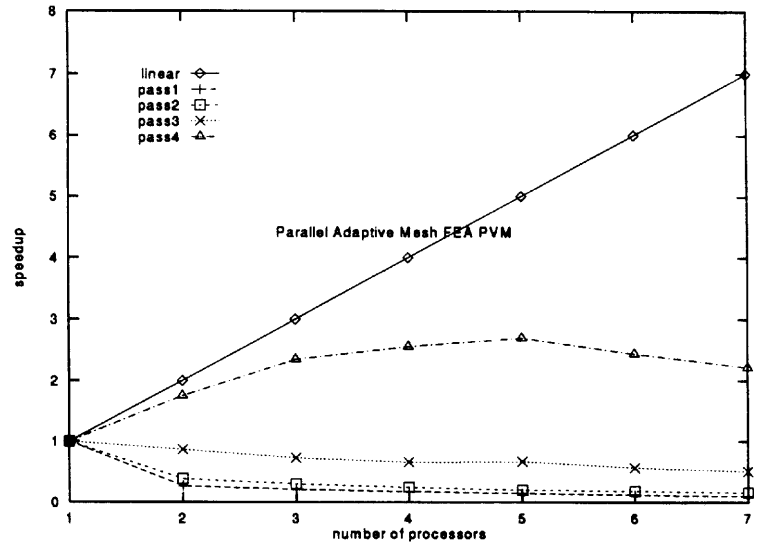


Figure 8. Speed-ups for parallel adaptive mesh finite element analysis using PVM and variable desired percentage error

These experiments and their results demonstrate the feasibility of using a cluster of workstations to perform parallel finite element analysis. Workstations have slower communication speed slower than dedicated parallel computers. However, they are still capable of performing parallel finite element analysis effectively. Accordingly, with its low cost/performance ratio a cluster of workstations can be a good alternative to expensive parallel computers. Today, workstations are widely used by many organizations. An organization can use its idle workstations to perform parallel computing without extra cost for parallel computers.

To further study parallel computation using a cluster of workstations, one has to consider that the performance of using a cluster of workstations for parallel computing may be affected by other users. As shown above, if one or more users run other CPU/memory-intensive applications on at least one of the workstations in use, the performance can be degraded to an unacceptable degree. To solve this problem, one method is to limit the access to workstations in use during the parallel computation. Another solution is to develop advanced algorithms that can perform dynamic load balancing and task rescheduling during the parallel computation. With these algorithms a parallel application can adjust its computing load distribution among workstations according to the loading in each workstation in use, such that the best performance can be obtained.

ACKNOWLEDGEMENTS

Most of this material is based upon work supported by the National Science Foundation under Award ECS-9006516. The authors would also like to thank the Cornell Theory Center's Advanced Computing Research Institute for providing time on its Hypercube concurrent computer (iPSC/2), the San Diego Supercomputer Center for providing time on its Paragon parallel computer and the Computer Science Department at the University of California at Santa Barbara for providing time on its Meiko CS-2 parallel computer.

REFERENCES

1. C. Baiocchi, and A. Capelo, *Variational and Quasivariational Inequalities*, Wiley, New York, 1984.
2. D. Kinderlehrer, and G. Stampacchia, *An Introduction to Variational Inequalities and Their Applications*, Academic, New York, 1980.
3. J. T. Oden, and N. Kikuchi, 'Theory of variational inequalities with applications to problems of flow through porous media', *Int. J. Eng. Sci.*, **18**, 1173–1284 (1980).
4. J. C. Bruch Jr., 'A survey of free boundary value problems in the theory of fluid flow through porous media: variational inequality approach. Parts I and II', *Adv. Water Resources*, **3**, 65–80, 115–124 (1980).
5. J. Crank, *Free and Moving Boundary Problems*, Clarendon, Oxford, 1984.
6. O. C. Zienkiewicz and J. Z. Zhu, 'A simple error estimator and adaptive procedure for practical engineering analysis', *Int. j. num. methods. engr.*, **24**, 337–357 (1987).
7. V. J. Burkley, and J. C. Bruch Jr., 'Adaptive error analysis in seepage problems', *Int. j. numer. methods eng.*, **31**, 1333–1356 (1991).
8. V. J. Burkley, J. C. Bruch Jr. and O. C. Zienkiewicz, 'Adaptive meshes used in solving a free surface seepage problem', in J. R. Whiteman (ed.) *The Mathematics of Finite Elements and Applications,* Vol. VII, Academic, New York, 1991, pp. 101–110.
9. M. C. Rivara, 'Algorithms for refining triangular grids suitable for adaptive and multigrid techniques', *Int. j. numer. methods eng.*, **20**, 745–756 (1984).
10. M. C. Rivara, 'A grid generator based on 4-triangles conforming mesh-refinement algorithms', *Int. j. numer. methods eng.*, **24**, 1343–1354 (1987).
11. R. Courant and D. Hilbert, *Methods of Mathematical Physics*, Vol. II, Interscience, New York, 1962.
12. D. Funaro, A. Quarteroni, and P. Zanolli, 'An iterative procedure with interface relaxation for domain decomposition methods', *SIAM J. Numer. Anal.*, **25**, 1213–1236 (1988).
13. L. Marini, and A. Quarteroni, 'A relaxation procedure for domain decomposition methods using finite elements', *Numer. Math.*, **55**, 575–598 (1989).
14. J. C. Bruch Jr. and J. M. Sloss, 'Alternating iteration and elliptic variational inequalities', *Numer. Math.*, **47**, 459–481 (1985).

15. C. A. Papadopoulos, J. M. Sloss, and J. C. Bruch Jr., 'A domain decomposition technique applied to a free surface problem', in P. Neittaanmaki (ed.), *Proc. Conf. on Numerical Methods for Free Boundary Problems*, Jyvaskyla, 1991, Birkhauser, Basel, 1991, pp. 337–346.
16. C. A. Papadopoulos, K. P. Wang, J. M. Sloss, and J. C. Bruch Jr., 'Domain decomposition for free boundary seepage', *Fluid Flow*, **1**, 37–48 (1991).
17. G. Rodrigue, and S. Shah., *Parallel Supercomputing: Method, Algorithms and Applications*, Wiley, New York, 1989, pp. 77–88.
18. R. E. White, 'Multisplittings and parallel iterative methods', *Comput. Methods Appl. Mech. Eng.*, **64**, 567–577 (1987).
19. C. Farhat, and E. Wilson, 'A new finite element concurrent computer program architecture', *Int. j. numer. methods eng.*, **24**, 1771–1792 (1987).
20. M. M. Cecchi, A. Pica, and E. Secco, 'A study of the venice lagoon by domain decomposition of a finite element model' *Proc. Ninth Int. Conf. on Finite Elements in Fluids—New Trends and Applications*, Venice, October 1995, pp. 1417–1425. Dip di Matematica Pura ed Applicata, Università di Padova, Padova, Italy, 1995.
21. K. P. Wang, and J. C. Bruch Jr., 'Solutions of a steady state free surface seepage problem on a hypercube concurrent computer' *Eng. Comput.*, **6**, 225–236 (1989).
22. K. H. Hoffmann, and J. Zou, 'Parallel algorithms of Schwarz variant for variational inequalities' *Numer. Func. Anal. Optim.*, **13**, 449–462 (1982).
23. K. P. Wang, and J. C. Bruch Jr., 'A SOR iterative algorithm for the finite difference and the finite element methods that is efficient and parallelizable', *Adv. Eng. Softw.*, **21**, 37–48 (1994).
24. R. Butler, and E. Lush, 'User's guide to the P4 programming system', *Tech. Rep. TM-ANL/92/17*, Argonne National Laboratory, 1992.
25. A. Geist, A. Beguelin, J. Dongarro, W. Jiang, R. Manchek, and V. Sunderam, 'PVM 3 user's guide and reference manual' *Tech. Rep., ORNL/TM-12187*, Oak Ridge National Laboratory, 1994.
26. K. P. Wang, and J. C. Bruch, Jr., 'Finite difference and finite element analyses using a cluster of workstations', in H. Power (ed.), *Application of High Performance Computing in Engineering IV, Proc. Fourth Int. Conf.*, Milan, June 1995, Computational Mechanics Publications, Southampton, 1995, pp. 131–138.

