SAND2000-0146C

Massively parallel boundary integral element method modeling of particles in a low Reynolds Number Newtonian fluid flow

M. S. Ingber[1], S. R. Subia[2], & L. A. Mondy[3]

[1] *Department of Mechanical Engineering, University of New Mexico, email: ingber@me.unm.edu*
[2] *Thermal/Fluid Computational Engineering Sciences Department*
[3] *Multiphase Transport Processes Department Sandia National Laboratories Albuquerque, New Mexico, USA*

Abstract

The analysis of many complex multiphase fluid flow systems is based on a scale decoupling procedure. At the macroscale continuum models are used to perform large-scale simulations. At the mesoscale statistical homogenization theory is used to derive continuum models based on representative volume elements (RVEs). At the microscale small-scale features, such as interfacial properties, are analyzed to be incorporated into mesoscale simulations. In this research mesoscopic simulations of hard particles suspended in a Newtonian fluid undergoing nonlinear shear flow are performed using a boundary element method. To obtain an RVE at higher concentrations, several hundred particles are included in the simulations, putting considerable demands on the computational resources both in terms of CPU and memory. Parallel computing provides a viable platform to study these large multiphase systems. The implementation of a portable, parallel computer code based on the boundary element method using a block-block data distribution is discussed in this paper. The code employs updated direct-solver technologies that make use of dual-processor compute nodes.

1 Introduction

Many modern day manufacturing processes involve the use of multiphase fluid suspensions (particles suspended in a carrier fluid) to achieve certain characteristics in a product. Suspensions are also prevalent in many industrial processes where the particulate phase serves some active role in the functionality of the process. In either event the suspension flows into place, and the resulting distribution of volume fraction of the suspended particulate phase often dictates the effectiveness of the fluid suspension in the process. Owing to the complex nature of fluid-particle interaction in a suspension, the planned use of suspensions in a process often relies heavily upon design experience rather than systematic analytic design procedures. In an effort to better understand suspension flows, researchers have experimentally studied the overall characteristics of the flow [1, 2]. These efforts have produced new insights and phenomological continuum models of suspension flow behavior [3]. Using these phenomological models as part of numerical models, it has been demonstrated that this approach has some merit for predictive design [4].

Mesoscopic analyses of multiphase systems are used to determine parameters used in continuum models, such as diffusion coefficients or hindered settling correlations. In these analyses individual elements of the dispersed phase are explicitly included in the models. A number of different methods for mesoscopic simulations have been used for this purpose including the method of Stokesian dynamics [5], the finite element method, and the boundary element method. In this research mesoscopic simulations of hard particles suspended in a Newtonian fluid undergoing nonlinear shear flow are performed using a boundary element method.

The advantage of using a boundary element method for modeling the particle motions in a suspension is that the difficulty in generating a volume grid for the fluid/particle system, as in a finite difference or finite element approach, is avoided since the boundary element method uses only a surface discretization. Although some finite element simulations of this type have been carried out [6], they are generally limited to dilute systems and use empirical models to address the issue of particle-particle interactions. Furthermore, these applications often require specialized grid generation tools that might not be available to many analysts.

Boundary integral formulations have been used for several years to study systems comprised of solid particles suspended in low Reynolds number fluid flows. Youngren and Acrivos [7] presented a direct boundary integral element method (DBIEM) to model the inertialess flow about single particles. Higdon [8] used a similar formulation to study shear flows over ridges and cavities, and Pozrikidis [9] later used the DBIEM to study some creeping flow problems in channels. Tran-Cong *et al.* [10] used the DBIEM to study the effective viscosity of a few particles in arrays of periodic cells, and Vincent *et al.* [11] modeled the flow about particles of arbitrary shape.

The same techniques used in boundary element models of systems with a few particles can be extended to fluid suspensions containing a large number of particles. To obtain an RVE at higher concentrations, several hundred particles must be included in the simulations, thus placing considerable demands on the computational resources both in terms of CPU and memory. Parallel computing provides a viable means to study these large multiphase systems. Reported parallel implementations of boundary element method computer codes for the analysis of low Reynolds number fluid flows include the indirect BIEM implementation of Karilla *et al.* [12] and the direct BIEM implementation of Ingber *et al.* [13]. Both of these implementations have been somewhat specialized in nature, as they target a specific compute platform or a specific message-passing procol.

In this paper a new implementation of a portable, parallel boundary element computer code is discussed for the analysis of suspension flows. The goal in developing this code is to provide a nearly platform-independent implementation of the boundary element method that can be used in both sequential and parallel computing. This new implementation draws heavily upon previous experience with the DBIEM [13, 14]. Portability of the code for parallel platforms is enhanced by employing the MPI message-passing library [15]. The code employs updated parallel direct-solver technologies which can take advantage of dual-processor compute nodes. Preliminary performance timings obtained, with the code on a platform with dual-processor compute nodes, are reported.

2 Governing Equations

The continuity equation for incompressible flow about suspended particles is given by

$$\frac{\partial u_i(\mathrm{x})}{\partial x_i} = 0 \ , \tag{1}$$

where u_i are the components of velocity. For steady-state low Reynolds number creeping flow of a Newtonian fluid with viscosity μ, the momentum equation becomes

$$\mu \frac{\partial^2 u_i(\mathrm{x})}{\partial x_j \partial x_j} = \frac{\partial p(\mathrm{x})}{\partial x_i} \ . \tag{2}$$

For limiting values $u_i \to 0$ and $p \to 0$ at infinity, the fundamental solutions of the above equations are

$$u^*_{ij}(\mathrm{x}, \mathrm{y}) = \frac{1}{8\pi\mu r} \left(\delta_{ij} + r_{,i} r_{,j}\right) \ , \tag{3}$$

and

$$p^*_j(\mathrm{x}, \mathrm{y}) = \frac{r_{,j}}{4\pi r^2} \ , \tag{4}$$

where $r = |\mathrm{x} - \mathrm{y}|$. The shear stress tensor, T_{ij}, in terms of the physical variables is given by

$$T_{ij}(u_i, p) = -p\delta_{ij} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \quad . \tag{5}$$

Further considering an additional scalar field, q, a vector field, v_i, and a corresponding stress tensor, $T_{ij}(v_i, q)$, it can be shown using the divergence theorem [16] that the governing velocity boundary integral equation is given by

$$c_{ij}(\mathrm{x})u_j(\mathrm{x}) + \int_\Gamma q^*_{ijk}(\mathrm{x},\mathrm{y})n_j(\mathrm{y})u_k(\mathrm{y})\mathrm{d}\Gamma(\mathrm{y}) = -\int_\Gamma u^*_{ij}(\mathrm{x},\mathrm{y})f_j(\mathrm{y})\mathrm{d}\Gamma(\mathrm{y}), \tag{6}$$

where $c_{ij}(\mathrm{x})$ is a coefficient tensor function, $q^*_{ijk}(\mathrm{x},\mathrm{y})$ is a fundamental solution

$$q^*_{ijk}(\mathrm{x},\mathrm{y}) = -\frac{3}{4\pi}\frac{r_{,i}\,r_{,j}\,r_{,k}}{r^2} \quad , \tag{7}$$

f_j are the traction components, and n_j are components of the unit normal vector along $\Gamma(\mathrm{y})$.

The above equations are generally sufficient to permit solution of fluid flow problems on a given domain. However, for problems of suspension flow where neither the velocity nor traction components on the particle surface are known, the boundary integral equations must be supplemented with additional equations to provide closure. To this end the velocities at the particle surface can be related to the particle translational and angular velocities, u^p_i and ω^p_j, as

$$u_i = u^p_i + \epsilon_{ijk}\omega^p_j|x_k - x^p_k| \quad , \tag{8}$$

where ϵ_{ijk} is the alternating tensor, and $|x_k - x^p_k|$ is the distance from the particle surface to the particle center.

Closure for particle suspension flows can then be achieved by introducing the static equilibrium relations for the particles. For force and moment equilibrium, the total forces on a particle surface, Γ^p, are equal to the external forces, b^p_i, applied to the particle

$$\int_{\Gamma^p} f_i d\Gamma^p(\mathrm{y}) + b^p_i = 0 \, , \tag{9}$$

and no net torque acting on the particle implies

$$\int_{\Gamma^p} \epsilon_{ijk}|x_i - x^p_i| f_i d\Gamma^p(\mathrm{y}) = 0 \quad . \tag{10}$$

3 Code Implementation

To obtain a numerical boundary element method implemention, discrete interpolants are introduced for the field variables components, u_i, and f_j, and for the boundary geometry, Γ, with local support over elemental subdivisions of the surface boundary. In particular, superparametric elements are used in which the geometry is interpolated quadratically and the field variables are considered constant within the element. Substituting these interpolations into (6,9,10), along with the fundamental solutions (3,7), the discretized boundary integral equations are obtained. These discretized equations can then be integrated numerically to obtain a system of equations

$$\begin{bmatrix} [G_{ij}] & [H_{il}]\,[K_{lj}] \\ [M_{ij}] & [0] \end{bmatrix} \begin{Bmatrix} f_j \\ u_j \end{Bmatrix} = \begin{Bmatrix} 0 \\ b_i^p \end{Bmatrix} . \tag{11}$$

Here G_{ij} and H_{il} represent the integrations in (6), M_{ij} represent the integrations in (9) and (10), and K_{lj} represents possible modifications in (6) owing to (8). It is important to note that unlike most boundary element formulations which yield dense fully-populated matrices, the partitioned system (11) contains dense upper blocks and sparsely populated lower blocks.

Given a set of proper boundary conditions along the problem boundary, Γ, removed from the particle surfaces, the system of equations can then be solved for the unknown surface tractions and velocities as well as the particle translational and angular velocities. In the present code implementation, provisions for user-defined boundary conditions and periodic boundary conditions are provided similar to those used by Tran-Cong *et al.* [10] in addition to the standard velocity and traction boundary conditions.

Once the particle translational and angular velocities have been determined, the quasi-static approach of Dingman *et al.* [14] can be used to simulate particle motions. In this approach the equations of particle motion are integrated using a set of Euler parameters [17].

As in most BIEM codes, there are two portions of the code which account for most of the CPU time, namely, assembling the system of equations (11) by performing the numerical integrations to obtain the components G_{ij}, H_{il}, and M_{ij}, and solving the assembled system of equations. For a small number of unknowns, the computation cost of matrix assembly can be on the same order of the solution phase. However, as the number of unknowns increases, the cost of direct solution greatly outweighs the cost of assembly. For sequential applications the system of linear equations (11) is solved using the LAPACK [18] direct solver. For parallel platforms, additional strategies are adopted as discussed below to exploit different aspects of parallelism.

The assembly phase of the code can be efficiently parallelized since all integrations required to assemble (11) are independent. It is, however, important that the workload be balanced for each of the processors, and to this end a block-block data decomposition is adopted that assigns specific integrations in the discrete form of (6), (9), and (10) to be performed on certain processors. In this scheme each processor is responsible for populating submatrices of the same general form as (11) and is equivalent to the *owner-computes* approach described by Natarajan and Krishnaswamy [19].

Many parallel solver packages have appeared over the past several years, but most of these packages are dedicated to providing iterative solutions for sparse systems of equations. Generally speaking, iterative solvers work best when the system matrix is somewhat diagonally dominant but because of the underlying structure of (11), that may not be the case when using the DBIEM to analyze suspension flows. Nevertheless, there has been some success in using iterative solvers for parallel boundary element simulations [20, 21, 22]. The use of iterative solvers often requires experience in their operation, since some of the solver parameters may require adjustments appropriate to the application. Since the motivation here in the development of this code is to provide a tool that is easy to use and requires less skill on the part of the analyst, a parallel direct solver is implemented.

Although there are a number of different approaches that can be used in developing parallel direct solvers [23], a family of solvers utilizing a two-dimensional torus-wrapped mapping appears to provide the best performance. Use of such a solver in the context of the boundary element method has previously been reported by Natarajan and Krishnaswamy [19]. In the current work an updated version of the torus-wrapped solver of Hendrickson and Womble [24] is used. The solver requires that the system matrix be mapped into a two-dimensional block-block array, and the block-block data decomposition used in the current DBIEM implementation is compatible with this requirement.

4 Test Problem

A simple pressure-driven flow in a square conduit with suspended particles as shown in Figure 1, is chosen as a demonstration benchmark test problem. Boundary conditions used in the simulation are no-slip velocity conditions along the boundaries parallel to the flow direction and traction conditions on the boundaries perpendicular to the flow direction. Two variations of the problem are studied. One to demonstrate speedup and the other to examine scaled parallel efficiency.

All simulations are carried out on the Intel ASCI Red Teraflops supercomputer at Sandia National Laboratories. The compute platform includes 4,536 333MHz Pentium II Xeon dual-processor compute nodes with 256Mb of memory available on each node. In normal operation program execution utilizes only one of the CPUs on an individual compute node. More recently the Intel computational scientists have released math libraries written specifically to take advantage of the other CPU. This paper compares the performance of the current benchmark problem with and without the use of the second CPU.

The test problem concerning speedup serves a dual purpose in the anal-

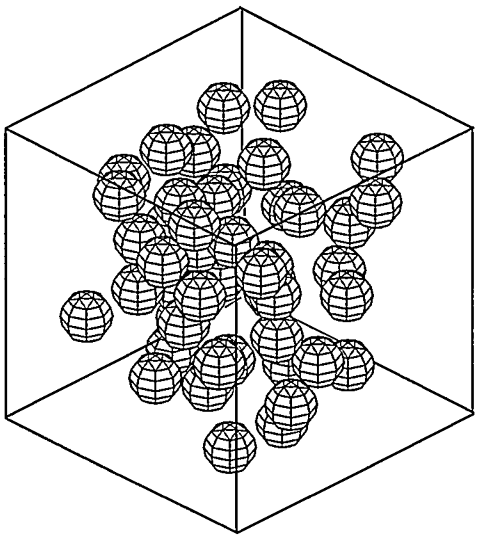

Figure 1: Model cell geometry.

ysis. This problem is used to provide working estimates of speedup and is also used to quantify any improvement in performance contributed by the secondary processor. To demonstrate speedup, a problem of fixed size is considered with 48 particles and 4,512 degrees of freedom using a solver blocksize of 32. After analyzing the problem on a single node, the same problem is analyzed on an increasing number of nodes, namely, 4, 16, 32, and 64 nodes both with and without the second processor active. In all cases, the torus-wrap mapping used a square processor array. Timings for the individual analyses are given in Table 1. The use of two processors per node is seen to yield lower overall and solver CPU times than when using only one processor. Further, the matrix assembly time is seen to be essentially perfectly scalable. This is a result of the fact that all integrations are independent, and the only communication required during the assembly phase is a gather of the load vector.

The n-node speedup, S_n, is defined as the ratio of the time taken to complete the analysis on a single node, T_1, to the time taken to complete the analysis on multiple nodes, T_n, and is given by

$$S_n = \frac{T_1}{T_n} \; . \tag{12}$$

procs (n)	DOF	matrix assembly time	1 proc solver time	2 proc solver time	1 proc total time	2 proc total time
1	4512	72.8	285.8	164.1	358.9	238.7
4	2256	18.6	77.9	48.2	97.1	67.5
16	1128	6.5	23.1	16.0	30.3	23.3
36	752	3.9	13.2	10.1	19.5	14.8
64	564	1.8	8.8	6.9	11.2	9.4

Table 1: Timings for boundary element code speedup study.

procs (n)	DOF	matrix assembly time	BLK 32 solver time	BLK 40 solver time	BLK 32 total time	BLK 40 total time
1	3510	44.3	86.4	112.3	131.1	157.3
4	7032	41.9	164.5	205.2	207.3	247.8
16	14076	40.5	363.3	465.8	405.1	507.4
64	28088	39.4	725.8	925.3	768.0	967.9

Table 2: Timings for boundary element code scaled parallel efficiency study.

A plot of the speedup ratio for these runs shown in Figure 2 indicates that the use of two processors per node yields a slightly lower speedup than obtained with a single processor per node. This is the result of the communication time being a larger portion of the overall wall clock time when using two processors. Nevertheless, CPU times are substantially reduced when using the second processor on each node.

The scaled parallel efficiency of the code is next studied by considering a sequence of problem sizes chosen so that as the number of nodes is increased, the problem size also increases so that the size of the coefficient matrix assembled and reduced on each node remains nearly constant. The different combinations of number of nodes and problem sizes characterized by the degrees of freedom are shown in Table 2. Timing results for these problems are also shown in Table 2, where the results for different blocksize are denoted as BLK 32 for a blocksize of 32 and BLK 40 for a blocksize of 40. From these results the assembly portion of the code is again seen to be almost perfectly scalable, since the assembly time is nearly constant for all cases. Based upon these timings, the scaled parallel efficiency is seen to be close to 100 percent for the torus-wrap direct linear equation solver showing the efficiency of the block-block data distribution in minimizing the communication overhead.

From the solver timings and the overall execution times, it is seen that in all cases using a blocksize of 32 works better than blocksize of 40. This is

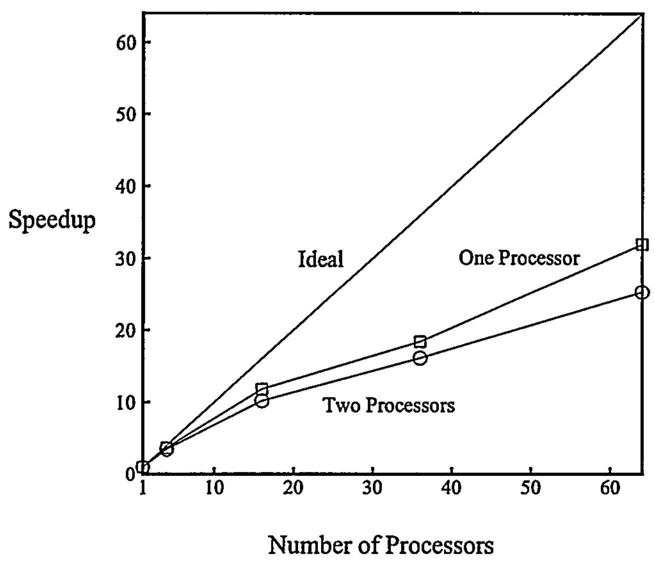

Figure 2: Stokes flow boundary element code speedup.

also shown graphically in Figure 3, where it is interesting to note that based upon the slope of the curves, the combined benefit of the second processor and a blocksize of 32 seems to be greater (CPU times are lower) for a larger number of processors.

5 Discussion

The development of a portable boundary element code for the analysis of low Reynolds number suspension flows is outlined in this paper. Testimony to the portability of the code is given by its successful use on different compute platforms. To date the code has been compiled and run on several Unix workstation serial platforms that include HP, SGI, Digital, and Sun. Although the computer code can be used exclusively for sequential calculations, its full power is best appreciated in a parallel computing environment. The parallel code has been successfully run on several multiprocessor platforms that include Alta, SGI, IBM, Sun, and the Intel Teraflops.

Using a simple test problem, it has been demonstrated that the parallel boundary element code solution algorithm is scalable to at least 64 processors for a fixed-size problem. Although the problems presented here

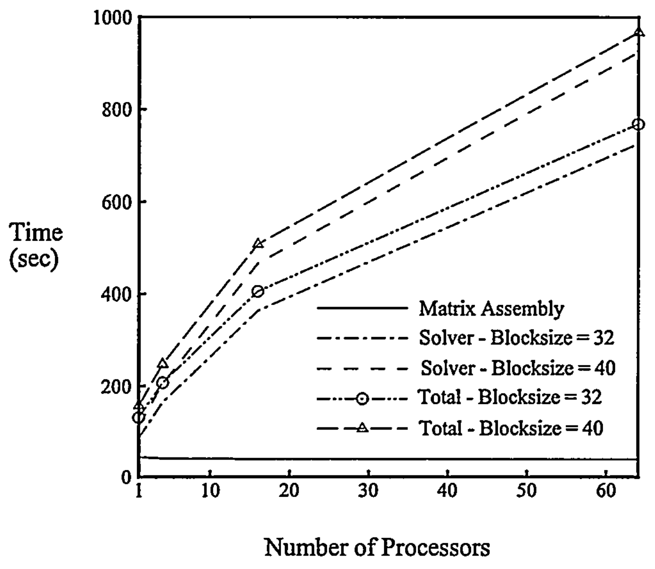

Figure 3: Overall performance timings for the scaled parallel efficiency benchmark test.

utilize but a modest number of processors, it is worth mentioning that the code has already been used in analyses employing thousands of processors. This provides some confidence that the computer code can be used to study suspension flows containing a large number of particles.

The results generated using the Intel Teraflops machine demonstrate that the computation time can be significantly reduced using the second processor on each node simply by linking the appropriate BLAS libraries. The results of the study also indicate that when using the torus-wrapped solver on this platform, the best performance is obtained by using a solver blocksize of 32, whereas Natarajan and Krishnaswamy [19] determined that a blocksize between 40-50 proved optimum for the IBM SP2.

Acknowledgements

The authors would like to thank Dr. Joseph D. Kotulski for providing the solver used in our work and for his help in implementing it at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States

Department of Energy under Contract DE-AC04-94AL85000. We gratefully acknowledge funding from the DOE's Accelerated Strategic Computing Initiative (ASCI) program and access to the Sandia/Intel Teraflop computer for this work. Partial support for this work was also provided by the US Department of Energy Applied Mathematical Sciences Division Grand Challenge Applications Program and the US Department of Energy Office of Basic Energy Sciences.

References

[1] Karnis, Goldsmith, H. L. & Mason, S. G., The kinetics of flowing dispersions I. Concentrated suspensions of rigid particles., *J. Colloid Interface Sci*, **22**, 531–553, 1966.

[2] Leighton, D. & Acrivos, A., Measurement of shear-induced self-diffusion in concentrated suspensions of spheres, *J. Fluid Mech.*, **177**, 109–131, 1987.

[3] Phillips, R. J., Armstrong, R. C., Brown, R. A., Graham, A. L. & Abbott, J. R., A constitutive equation for concentrated suspensions that account for shear-induced particle migration, *Phys. Fluids A*, 4, 30–40, 1992.

[4] Subia, S. R., Ingber, M. S., Mondy, L. A., Altobelli, S. A., & Graham, A. L., Modelling of concentrated suspensions using a continuum constitutive equation, *J. Fluid Mech.*, **373**, 193–219, 1998.

[5] Brady, J. F. & Bossis, G., Stokesian Dynamics, *Ann. Rev. Fluid Mech.*, **20**, 111–157, 1988.

[6] Zhu, M. & Hu, H. H., Direct numerical simulation in the mixing of particulate flow, *Proceedings of ASME Fluids Engineering Division Summer Meeting, Washington D.C.*, FEDSM98-4904, 1998.

[7] Youngren, G. K. & Acrivos, A., Stokes flow past a particle of arbitrary shape: a numerical method of solution, *J. Fluid Mech.*, **69**(2), 377–402,1979.

[8] Higdon, J. J. L., Stokes flow in arbitrary two-dimensional domains: shear flow over ridges and cavities, *J. Fluid Mech.*, **159**, 195–226, 1985.

[9] Pozrikidis, C., Creeping flow in two-dimensional channels, *J. Fluid Mech.*, **180**, 495–514, 1987.

[10] Tran-Cong, T., Phan-Thien, N. & Graham, A. L., Stokes problems of multiparticle systems: Periodic arrays, *Phys. Fluids A*, 2(5), 666–673, 1991.

[11] Vincent, J., Phan-Thien, N. & Tran-Cong, T., Sedimentation of multiple particles of arbitrary shape, *J. Rheol.*, **35**(1), 1–26, 1991.

[12] Karilla, S. J., Fuentes, Y. O., & Kim, S. Parallel computational strategies for hydrodynamic interactions between rigid particles of arbitrary shape in a viscous fluid, *J. Rheol.*, **36**(3), 413–440, 1992.

[13] Ingber, M. S., Womble, D. E. & Mondy, L. A., A parallel boundary element formulation for determining effective properties of heterogeneous Media, *Int. J. Num. Meth. Engrg.*, **37**, 3905–3919, 1994.

[14] Dingman, S. E., Ingber, M. S. & Mondy, L. A., Particle tracking in three-dimensional Stokes flow, *J. Rheol.*, **36**(3), 413–440, 1992.

[15] Snir, M., Huss-Lederman, S., Walker, D. & Dongarra, J., *MPI – The Complete Reference*, MIT Press, 1998.

[16] Brebbia, C. A., Telles, J. C. F. & Wrobel, L. C., *Boundary Element Techiques*, Springer-Verlag, Berlin and New York, 386–387, 1983.

[17] Roberson, R. E. & Schwertassek, R., *Dynamics of Multibody Systems* Springer-Verlag, Berlin and New York, 75–76, 1988.

[18] Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J. J., Du Croz, J., Greenbaum, A., Hammarling, S. & Sorenson, D., LAPACK: A portable linear algebra library for high-performance computers, *Computer Science Technical Report CS-90-105*, Univ. of Tennessee, 1990.

[19] Natarajan, R. & Krishnaswamy, D., A case study in parallel scientific computing: the boundary element on a distributed-memory multicomputer, *Engrg. Anal. Boundary Elem.*, **18**(3), 183–193, 1996.

[20] Ye, T. Q., Wang, R. P. & Zhang, X. S., Boundary element analysis by Krylov subspace method on MIMD parallel multiprocessor system, *Engrg. Anal. Boundary Elem.*, **18**(3), 183–193, 1996.

[21] Kamiya, N., Iwase, H. & Kita, E., Parallel implementation of boundary element method with domain decomposition, *Engrg. Anal. Boundary Elem.*, **18**(3), 209–216, 1996.

[22] Kamiya, N., Iwase, H. & Kita, E., Performance evaluation of parallel boundary element analysis by domain decomposition, *Engrg. Anal. Boundary Elem.*, **18**(3), 217–222, 1996.

[23] Kumar, V., Grama, A., Gupta, A. & Karypis, *Introduction to Parallel Computing*, Benjamin/Cummings, Redwood City Ca., 178–198, 1994.

[24] Hendrickson, B. A. & Womble, D. E., The Torus Wrap Mapping for dense matrix calculations on massively parallel computers, *SIAM J. Sci. Comp.*, **15**, 1201–1226, 1994.