Engineering Analysis with Boundary Elements **19** (1997) 41–48
© 1997 Elsevier Science Ltd
All rights reserved. Printed in Great Britain
0955-7997/97/$17.00

ELSEVIER PII: S0955-7997(97)00024-6

Parallel MR-BEM using ScaLAPACK

Jacques Lobry[a] **& Pierre Manneback**[b]

[a]*Power Systems Laboratory,* [b]*Computer Science Laboratory, Faculté Polytechnique de Mons, 7000 Mons, Belgium*

The multiple reciprocity method is a recent generalisation of the well-known Boundary Element Method. It allows the numerical analysis of Poisson's problem in an efficient and elegant manner since it converts the classical domain integral coming from excitation to a summation of boundary ones. However, the method leads to the computation of higher order kernels so that the assembly of the linear systems is significantly increased. In this paper, it is shown that parallel computing allows a substantial reduction in CPU time. Different data distribution strategies have been implemented and compared using standard ScaLAPACK computational kernel. Tests have been successfully run on a 24-processor Intel Paragon. © Elsevier Science Ltd

Key words: Multiple reciprocity method, boundary element method, parallel computing, ScaLAPACK, Gaussian elimination method.

1 INTRODUCTION

When the classical boundary element method is applied to the numerical analysis of Poisson's problem, the source term leads to a domain integral. Generally, it can be treated without introducing further unknowns. However, it requires the introduction of internal nodes and cells so that the domain integration may be time consuming. Therefore, some investigators have focused their research on the transformation of the domain integral into boundary ones. The first method of this type is the so-called dual reciprocity method[1,2] (DRM) that was invented in 1982 by Nardini and Brebbia for elastodynamic problems. The source term is approximated by a combination of known coordinate functions of which the particular solution is easy to find. Internal points have to be chosen in the domain and their number and position determine the accuracy of the results. More recently, Nowak and Brebbia have introduced the multiple reciprocity method[3–5] (MRM or MR-BEM) that is the most recent technique for converting domain integrals. Its principle lies in the use of higher order fundamental solutions that lead to a recurrent summation of boundary integrals that models the excitation. The new formulation is made by a series of boundary only integrals without any approximation. Those integrals have the same form as the one related to the left-hand side of Poisson's equation. Successive Laplacians of the source term and their normal derivatives have the same position as potential and flux whereas higher order kernels are at the same level as the Green's function. Thus resulting influence matrices are built in a manner similar to the classical ones. The number of terms in the summation depends on the complexity of the excitation but it should vanish rapidly provided that the problem has been properly scaled. However execution time is relatively important in the assembly part of the linear system in BEM and the MR-BEM technique still increases the computational effort.

The emergence of a mature parallel computing market now offers a convenient environment for solving computing intensive applications. It is therefore a natural and cost effective choice to consider parallel implementation of BEM methods. First parallel implementations of BEM were conducted in the early 1980s. A seminal paper is due to Symm[6] and other researchers, such as Davies,[7,8] Bryant *et al.*,[9] Lobry *et al.*[10] have brought their contributions to the area. Distributed DRM has also been recently studied.[11] An excellent account of the state-of-the-art has been written in Ref. 7. Our paper is devoted to the parallel implementation of MR-BEM on a coarse-grained distributed memory MIMD (multiple instruction stream, multiple data stream) architecture. Different data distributions on a grid of processors are considered and compared, with respect to the problem size, the number of processors and the number of higher order terms. The implementation is performed in standard Fortran 77, using BLACS[12] and ScaLAPACK[13] libraries for communication and computation, respectively. Therefore, the code is scalable and compatible on different distributed platforms, such as Cray T3D, IBM SP2 or Intel Paragon. Experiments are conducted on this last machine, showing that the assembly part of the code is relatively expensive but is parallelisable with excellent efficiency.

2 MR-BEM FORMULATION

Consider Poisson's problem defined on a 2-D domain Ω:

$$\nabla^2 u = b_0 \tag{1}$$

where u and b_0 are potential and source terms respectively. Classical BEM treatment consists of using the Green's kernel w_i that is the fundamental solution to Laplace's equation, i.e.

$$\nabla^2 w_i = -\delta_i \tag{2}$$

where δ_i is Dirac's distribution acting at definition point i. The two-dimensional kernel, w_i, is equal to $(-1/2\pi)\ln r$ where r is the distance from the point i to any point under consideration. Multiplying both sides of (1) and (2) by w_i and u respectively, performing integration over domain Ω and applying the second of Green's theorem, we obtain a formulation that involves only boundary unknowns:[14]

$$c_i u_i + \oint_{\partial\Omega}\left(u\frac{\partial w_i}{\partial n} - w_i\frac{\partial u}{\partial n}\right)\mathrm{d}\Gamma = -\int_\Omega \mathrm{w_i b_0 d}\Omega \tag{3}$$

where c_i is a geometric factor. From now on, a subscript '0' will complete the fundamental solution w_i since higher order kernels are introduced by MRM.

The domain integral on the right-hand side of (3) can be transformed if we consider a new function $w_{1,i}$ defined as:

$$\nabla^2 w_{1,i} = w_{0,i}$$

Thus, again from Green's theorem, the domain integral becomes:

$$\begin{aligned}\int_\Omega b_0 w_{0,i}\mathrm{d}\Omega &= \int_\Omega b_0\nabla^2 w_{1,i}\mathrm{d}\Omega \\ &= \oint_{\partial\Omega}\left(b_0\frac{\partial w_{1,i}}{\partial n} - w_{1,i}\frac{\partial b_0}{\partial n}\right)\mathrm{d}\Gamma \\ &\quad + \int_\Omega w_{1,i}\nabla^2 b_0\mathrm{d}\Omega \end{aligned}\tag{4}$$

where $\nabla^2 b_0$ may be denoted by b_1, i.e.

$$\int_\Omega w_{1,i}\nabla^2 b_0\mathrm{d}\Omega = \int_\Omega w_{1,i}b_1\mathrm{d}\Omega \tag{5}$$

Domain integral (5) is of the same form as in (3), hence the same decomposition can be applied by considering the new function $w_{2,i}$ defined in a similar way as $w_{1,i}$, i.e.

$$\nabla^2 w_{2,i} = w_{1,i}$$

Iterating the transformation process, a recurrent expression arises. The domain integral is then converted into the following series of boundary integrals:

$$\int_\Omega b_0 w_{0,i}\mathrm{d}\Omega = \sum_{p=0}^{\infty}\oint_{\partial\Omega}\left(b_\mathrm{p}\frac{\partial w_{p+1,i}}{\partial n} - w_{p+1,i}\frac{\partial b_p}{\partial n}\right)\mathrm{d}\Gamma \tag{6}$$

where higher order fundamental solutions $w_{p,i}$ and functions b_p and defined as:

$$\nabla^2 w_{p+1,i} = w_{p,i}$$

$$b_{p+1} = \nabla^2 b_p$$

for $p \geq 0$.

Substituting (6) into original expression (3) yields:

$$\begin{aligned} c_i u_i + \oint_{\partial\Omega}\left(u\frac{\partial w_{0,i}}{\partial n} - w_{0,i}\frac{\partial u}{\partial n}\right)\mathrm{d}\Gamma &= \\ -\sum_{p=0}^{\infty}\oint_{\partial\Omega}\left(b_p\frac{\partial w_{p+1,i}}{\partial n} - w_{p+1,i}\frac{\partial b_p}{\partial n}\right)\mathrm{d}\Omega \end{aligned}\tag{7}$$

Notice that no approximation was made here, this new boundary only formulation is an exact form of the original Poisson's problem.

Successive Laplacians of the source term are assumed to be known, as b_0 is a known function of space. The b_p functions can be calculated either analytically or numerically by using symbolic manipulators. The higher order kernels are easy to derive for both 2-D and 3-D cases. The two-dimensional expressions for $w_{p,i}$ and its normal derivative are given here:

$$\begin{aligned} w_{p,i} &= -\frac{r^{2p}}{2\pi}(A_p\ln r - B_p) \\ \frac{\partial w_{p,i}}{\partial n} &= -\frac{r^{2p-1}}{2\pi}\left[A_p(2p\ln r + 1) - 2pB_p\right]\frac{\partial r}{\partial n} \end{aligned}\tag{8a}$$

where coefficients A_p and B_p are obtained from the following recurrence relationships:

$$\begin{aligned} A_{p+1} &= \frac{A_p}{4(p+1)^2} \\ B_{p+1} &= \frac{1}{4(p+1)^2}\left[\frac{A_p}{p+1} + B_p\right] \end{aligned}\tag{8b}$$

for $j > 0$, $A_0 = 1$ and B_0 is arbitrary due to the non-uniqueness of the solution of the integral equation,[5] B_0 is often chosen equal to 0 or $(1/2\pi)$ ln $r_{\max}$, where $r_{\max}$ stands for the maximum distance between nodal points.

3 DISCRETISATION

The numerical evaluation of the integrals is obtained by subdividing the boundary $\partial\Omega$ into a number of boundary elements $\partial\Omega_j$. Discretisation of the integrals associated with the potential and its normal derivative in (7) requires a suitable approximation of those variables within each element. Linear boundary elements and linear interpolation on them is generally a good compromise between accuracy and simplicity of implementation. There results the assembly of the well-known influence matrices that we write with the subscript '0', i.e. $\mathbf{H}_0$ and $\mathbf{G}_0$. The recurrent boundary integrals of the right-hand side are of the same form as those related to the variables u and $\partial u/\partial n$. Therefore, a similar treatment can be conducted on them. Successive Laplacians of excitation b_0 are able to be directly calculated on the nodes of the boundary since b_0 is known. Thus discretisation of those integrals leads to higher order influence matrices[14] denoted as $\mathbf{H}_{p+1}$ and $\mathbf{G}_{p+1}$. Finally, the set of equations is of the matrix form:

$$\mathbf{H}_0\mathbf{u} - \mathbf{G}_0\mathbf{q} = \sum_{p}^{\infty} = 0\mathbf{H}_{p+1}\mathbf{d}_p - \mathbf{G}_{p+1}\mathbf{b}_p \tag{9}$$

where $\mathbf{u}$ and $\mathbf{q}$ represents the vectors of boundary potential and normal derivative respectively. Vectors $\mathbf{b}_p$ and $\mathbf{d}_p$ contain vertex values of b_p and $\partial b_p/\partial n$ respectively. Boundary conditions are considered by rearranging the system as usual.

Computation of the entries of $\mathbf{H}_0$ and $\mathbf{G}_0$ can be carried out in either a numerical or an analytical way. The latter case relates to the situation where source point i belongs to the boundary element under consideration. The singularity of the zero-order kernels necessitates this particular treatment for a better accuracy in integration. In this other case, Gauss quadrature integration is normally used. Expressions (8a) of the higher order kernels show that there is no singularity as distance r vanishes so that the recurrent integrals can all be evaluated numerically by using Gauss quadrature again. Thus MRM is easy to implement since the routines for numerical integration are similar to the classical ones.

The number of terms in the series (6) depends on the complexity of the source function b_0. In many cases, e.g. a polynomial definition of b_0, the summation is finite with a relative small number of terms. When the excitation gives rise to an infinite series, the summation has to be truncated by using a suitable convergence criterion that is described in detail by Nowak.[15] Generally, the terms vanish rapidly due the introduction of factorials into the denominators of coefficients A_p and B_p given in (8b).

4 PARALLEL IMPLEMENTATION OF MR-BEM

4.1 Introduction

Finite element methods (FEM) generally require large computational efforts. Although the boundary element method reduces the calculations of a problem, it can be time consuming when the geometry is three-dimensional and complex. Moreover, the related linear system involves a dense matrix whereas finite element methods yield to sparse systems, though the size of the systems are smaller for BEM than FEM. Parallel computing offers an interesting way of reducing the related computational cost.

Numerous types of parallel machines exist depending on the number and the type of processors, the connection and the memory topologies, etc. Our study is devoted to the use of a distributed architecture as it seems the top end topology for supercomputers in the future. A coarse-grained parallelism is exploited so that a small number of high-performance processors are used. On the other hand, Davies[7,8] works on massively parallel computers using small-grained implementation. Computation and communication complexities are studied with the model of Saad and Schultz.[16] The communication time of n consecutive data items between two processors is assumed to be of the form $\beta + n\tau$ where β is the latency and τ is the time to transfer one data item. The elementary computation time is ω. Two main steps should be considered for the MR-BEM: the assembly of the linear system and its solution. The second can only start when the first has been completed but each one may be decomposed into parallel tasks. Several strategies are possible for both the assembly and solving steps but they should be closely arranged in order to get the best global performance. Gaussian elimination with partial pivoting is used for the solution since it appears to be an efficient stable parallel solver for linear dense systems.[17]

Discretisation of boundary formulation (7) leads to a linear system of equations expressed by (9) in matrix form. The expanded form is now necessary in order to understand the strategy used in the parallel implementation presented in this section. Each equation is associated with a source point i that is chosen as the successive node of the boundary mesh. Let n be the number of nodes and, at the same time, the number of boundary elements $\partial\Omega_j$. Suppose that the series (6) is truncated so that $N+1$ terms are considered in the series. Thus, expansion of (9) is:

$$\begin{aligned} c_i u_i &+ \sum_{j=1}^{n}\left[\left(h^{(1)}_{0,i,j}u_{j,1} + h^{(2)}_{0,i,j}u_{j,2}\right) - \left(g^{(1)}_{0,i,j}q_{j,1} + g^{(2)}_{0,i,j}q_{j,2}\right)\right] \\ &= \sum_{p=0}^{N}\sum_{j=1}^{n}\left[\left(h^{(1)}_{p+1,i,j}b_{p,j,1} + h^{(2)}_{p+1,i,j}b_{p,j,2}\right)\right. \\ &\quad \left. -\left(g^{(1)}_{p+1,i,j}d_{p,j,1} + g^{(2)}_{p+1,i,j}d_{p,j,2}\right)\right] \end{aligned} \tag{10}$$

where $u_{j,k}$, $q_{j,k}$, $b_{p,j,k}$ ($k = 1, 2$, $p = 0$–N) are the values of u, $\partial u/\partial n$, b_p and $\partial b_p/\partial n$ respectively on the nodes

k of the element $\partial\Omega_j$. Coefficients $h^{(k)}_{p,i,j}$ and $g^{(k)}_{p,i,j}$ are deduced from integration of respective p-kernels $\partial w_{p,i}/\partial n$ and $w_{p,i}$ with the interpolation functions. Assuming that $\partial\Omega_j$ is enclosed between nodes j and $j+1$, (10) may be rewritten as:

$$\sum_{j=1}^{n}(H_{0,i,j}u_j - G_{0,i,j}q_j) = \sum_{p=0}^{N}\sum_{j=1}^{n}(H_{p+1,i,j}b_{p,j} - G_{p+1,i,j}d_{p,j}) \tag{11}$$

where

$$\begin{aligned} u_j &= u_{j-1,2} = u_{j,1} \\ q_j &= q_{j-1,2} = q_{j,1} \\ b_{p,j} &= b_{p,j-1,2} = b_{p,j,1} \\ d_{p,j} &= q_{p,j-1,2} = q_{p,j,1} \\ H_{p,i,j} &= h^{(1)}_{p,i,j} + h^{(2)}_{p,i,j-1}(+c_i \ \ if \ \ i=j \ for \ p=0) \\ G_{p,i,j} &= g^{(1)}_{p,i,j} + g^{(2)}_{p,i,j-1} \end{aligned} \tag{12}$$

Numerical integration is carried out by using an n_g-point Gauss quadrature. In practical implementations, higher order influence matrices are not stored in an array. Once calculated, coefficients $h^{(k)}_{p,i,j}$ and $g^{(k)}_{p,i,j}$ ($p>0$) are immediately multiplied by the right source function b_{pj} or d_{pj} and the result is stored in the right-hand side vector of the linear system.

4.2 Parallel assembling of the linear system

This section is devoted to the parallel assembling of the n by n linear system (11) which will be rewritten in the matrix form:

$$\mathbf{Sx} = \mathbf{f} \tag{13}$$

Each entry S_{ij} of matrix S is either a term $H_{0,ij}$ or $G_{0,ij}$, depending on the boundary condition at node j. From (12), S_{ij} may be decomposed as follows:

$$S_{ij} = s^{(1)}_{ij} + s^{(2)}_{ij} \tag{14}$$

where $s^{(k)}_{ij} = h^{(k)}_{0,ij}$ or $g^{(k)}_{0,ij}$ ($k = 1,2$).

We assume working on a grid of p_r by p_c processors, the total number of processors is then equal to $p = p_r p_c$. Data are block-cyclically distributed on this grid so that each coefficient S_{ij} is calculated by processor P_{kl}, where $k = (i-1)/b \bmod p_r$ and $1 = (j-1)/b \bmod p_c$. Parameter b represents the size of squared data block. This standard block cyclic data allocation is illustrated in the case of a 12 by 12 matrix distributed on a 2 by 3 grid with blocksize 2, the allocation is then as follows:

$$\begin{bmatrix} 00&00&01&01&02&02&00&00&01&01&02&02 \\ 00&00&01&01&02&02&00&00&01&01&02&02 \\ 10&10&11&11&12&12&10&10&11&11&12&12 \\ 10&10&11&11&12&12&10&10&11&11&12&12 \\ 00&00&01&01&02&02&00&00&01&01&02&02 \\ 00&00&01&01&02&02&00&00&01&01&02&02 \\ 10&10&11&11&12&12&10&10&11&11&12&12 \\ 10&10&11&11&12&12&10&10&11&11&12&12 \\ 00&00&01&01&02&02&00&00&01&01&02&02 \\ 00&00&01&01&02&02&00&00&01&01&02&02 \\ 10&10&11&11&12&12&10&10&11&11&12&12 \\ 10&10&11&11&12&12&10&10&11&11&12&12 \end{bmatrix}$$

where each entry of the above matrix gives the kl-index of the processor that holds the coefficient written at the same position in matrix S. The advantage of such a distribution is that it is extremely flexible, ranging from a classical column interleaved storage ($p_r = b = 1$) to a row one ($p_c = b = 1$). A particular case is the block grid distribution ($b = n/p_r$, $p_r = p_c$).

For the assembly of (14), some communication between neighbouring processors is generally required, depending on the column index. Indeed, the computation of $s^{(1)}_{ij}$ will always be local to a processor whereas $s^{(2)}_{ij}$ has to be transferred from processor P_{kl} to processor P_{kl+1} (P_{k0} if $1 = p_c$) only if column j is a border column, i.e. $j \bmod b = 0$. Figure 1 illustrates the assembly of a part of the matrix in processor P_{kl}, the transfers from and to this processor are shown by a bold line. In the case of the cyclic row distribution, no transfer is needed. As said above, the right-hand side f is computed by rearranging system (9) in order to take boundary conditions into account and by summing the higher order contributions. Each processor builds a part of this vector without any data transfer. A global accumulation is performed just before the solution step. Communication cost of the assembly part is evaluated as ($p_c > 1$):

$$\begin{aligned} T_{\text{comm}} &= \beta + \frac{n}{p_r}\frac{n}{bp_c}\tau \\ &= \beta + \frac{n^2}{pb}\tau \end{aligned} \tag{15}$$

since communication of all the coefficients can be done for each processor with one single message, as soon as the computation is performed is performed. Computation time is approximated as:

$$\begin{aligned} T_{\text{com}} &= 16\frac{n}{p_r}\frac{n}{p_c}Nn_g\omega \\ &= 16\frac{n^2}{p}Nn_g\omega \end{aligned} \tag{16}$$

The ratio of (15) to (16) is about:

$$\frac{\beta p}{16n^2Nn_g\omega} + \frac{\tau}{16bNn_g\omega} \tag{17}$$

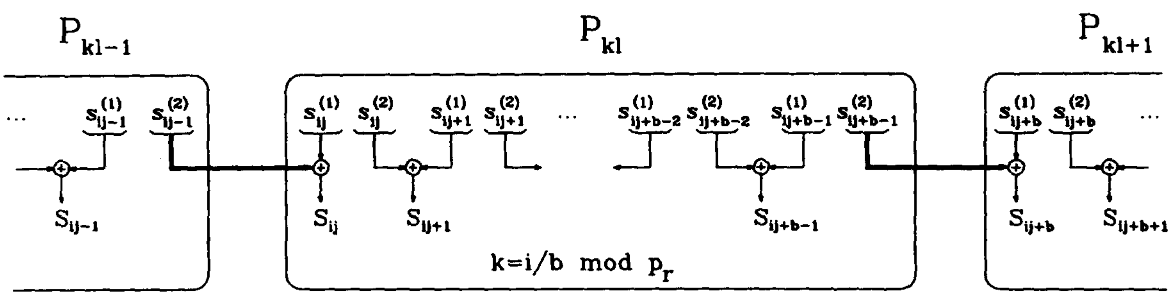

Fig. 1. Data transfer in the grid allocation.

It is clear that the communication overhead is negligible in the matrix assembly process. Therefore, this step is highly scalable and parallelisable whatever the grid topology may be.

4.3 Parallel solution of the linear system

There are numerous papers on the subject of the parallel solution of dense linear systems and it is generally admitted that LAPACK[18] sequential codes and their parallel extension to ScaLAPACK[13] are the most powerful and compatible codes for numerical linear algebra standard algorithms. Moreover, they make the conversion of a parallel code from a sequential version easy. ScaLAPACK Fortran routines PDGETRF for LU factorisation and PDGETRS for forward and backward substitution are used. They exploit the block cyclic data distribution on the grid of processors in an efficient manner. Interested readers are referred to the literature mentioned above for extensive description and test results. In Ref. 13, the complexity T_{LU} is shown to be

$$T_{\mathrm{LU}} = 2n\log_2(p_r)\beta + \frac{n^2}{2p}[2p_c + p\log_2(p)]\tau + \frac{2n^2}{3p}\omega \tag{18}$$

Although it is a cubic complexity, the real time for solving the system in parallel does not exceed the one required for the assembly step, where the complexity is $O(n^2)$, provided the size n is not too large say $n < 1000$). That is to say the best data distribution is governed on a global basis assembling + solution) rather than on the solution alone.

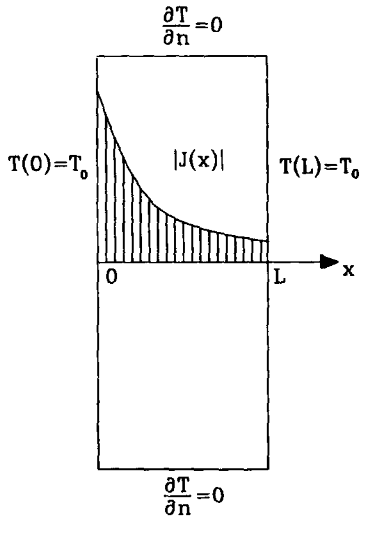

Fig. 2. Test-case for parallel MR-BEM.

5 PERFORMANCE RESULTS

In order to measure and to compare the performances of the parallel strategies, a very simple example is studied. The test case chosen consists of a 2-D heat problem described as follows. A time-harmonic current is flowing in a conductor of rectangular cross-section (Fig. 2). The frequency is sufficiently high to give rise to a skin effect so that the current density J is non-uniform. Assuming the height of the conductor is several times greater than the width, end-effects may be neglected. Then the analytical expression of J exists. It is deduced from Maxwell's equations and is of the complex form:

$$\mathbf{J}(x) = \mathbf{J}(0)e^{-(1+j)\frac{x}{\delta}}$$

where δ is the skin depth that depends on the material and decreases with the square root of the frequency. It is desirable to compute the mean temperature field T within the conductor while both sides are at Dirichlet conditions T_0. The governing equation is of Poisson type where the source term comes from the local dissipated Joule's power:

$$\nabla^2 T = -\frac{\rho}{\lambda}|\mathbf{J}|^2$$
$$= -\frac{\rho}{\lambda}|\mathbf{J}(0)|^2 e^{-2x/\delta}$$

where λ and ρ are the thermal conductivity and electrical resistivity of the material respectively. This problem has an analytical solution that can be compared with the numerical one. The field temperatures only vary with the x-direction as:

$$T(x) = T(9) + \frac{\rho\delta^2}{4\lambda}\left[1 - e^{-2x/\delta} + \left(e^{-2L/\delta} - 1\right)\frac{x}{L}\right]$$

From now on, we denote the source term as b_0. Successive Lablacians b_p are easy to obtain from

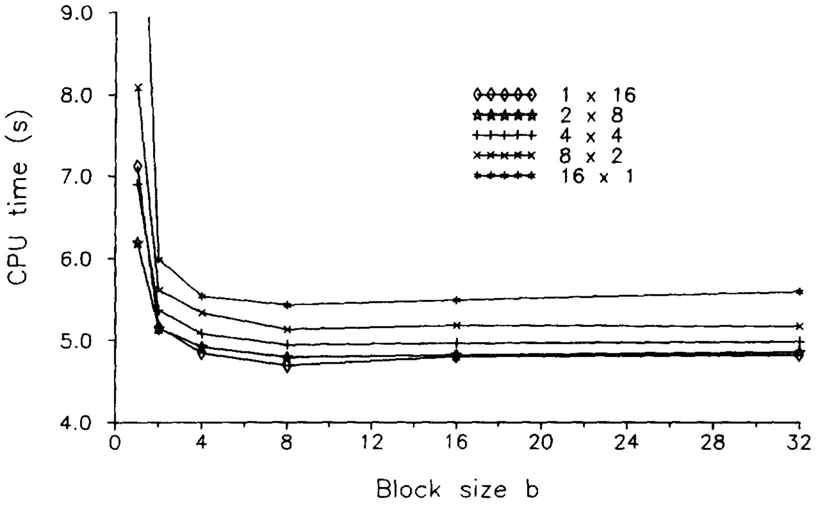

Fig. 3. CPU time vs block size ($n = 512$, $N = 5$, $p = p_r \times p_c = 16$).

differentiation:

$$b_p = \left[\frac{2}{\delta}\right]^{2p} b_0$$

Notice that the recurrent series is infinite due to the exponential function.

The implementation of the algorithm has been carried on in standard Fortran 77 using double precision. Communication library BLACS is used. The code has been run on the Intel Paragon of University of Lyon 1, consisting of 24 processors 50 MHz i860XP operating under the OSF/1 operating system. On each processor, the kernel computation for solving the linear system was performed using assembly-coded BLAS provided by Intel. Experimental tests have been conducted with several problem sizes ($n = 256$, 512, 1024, 2048). Problem size has been limited to 1024 due to the limitation of memory storage on the processors. The number of higher order terms ($N = 0$, 1, 5, 15) and the number of processors ($p = 1$, 4, 8, 16) are also parameters of the study. the assembly time is nearly independent of the grid topology since the communication overhead is negligible compared with the computation time. Tests reveal excellent efficiencies E_{ass} of this part of the MR-BEM algorithm ($E_{\text{ass}} > 95\%$). In fact, it is the solution step that governs the overall efficiency and, finally, the choice of the best processor topology. Optimal blocksize b for the block cyclic data distribution has been determined experimentally, considering the total execution time. It is found to be around 8, whatever the number of processors or the problem size may be. This is illustrated by the timings plotted in Fig. 3 for a size $n = 512$, a number of higher order terms $N = 5$ and 16 processors. This corroborates what is observed in the literature about LU factorisation using ScaLAPACK.[19] Furthermore, it is noticed that the block cyclic distribution is far better than simple cyclic wrapped interleaved allocation. It is also shown that the block column-oriented distribution ($p_r = 1$) is the best topology. Note that those conclusions may be slightly altered for very large sizes as it occurs in three-dimensional analysis but it is beyond the scope of this paper.

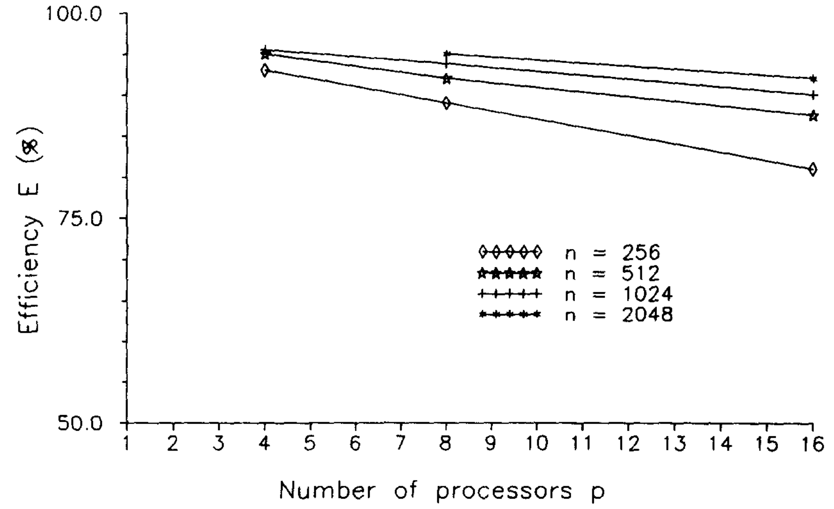

Fig. 4. Efficiency vs number of processors ($N = 5$, $b = 8$, $p_r = 1$).

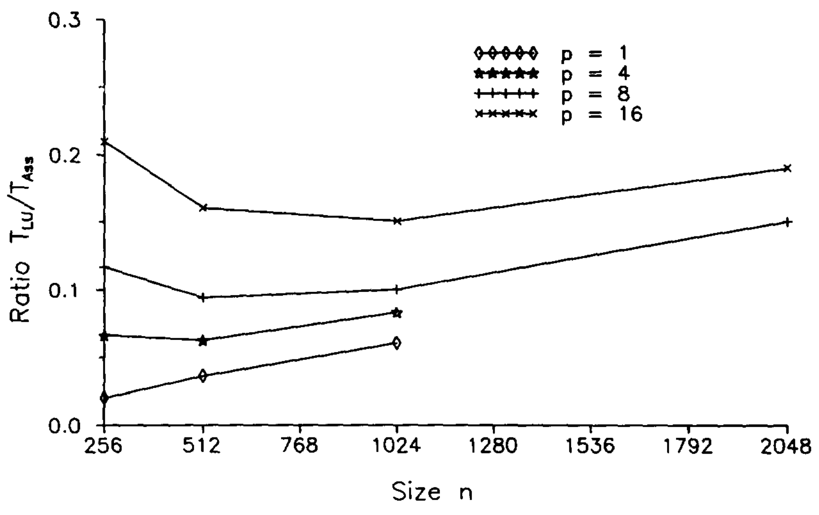

Fig. 5. Ratio T_{LU}/T_{Ass} vs size n ($N = 5$, $b = 8$, $p_r = 1$).

Figure 4 presents the global efficiency E versus the number of processors for the four sizes considered where optimal topology is considered. In the case of $n = 2048$, the implementation cannot be run on 1 and 4 processors due to memory limitation. Extrapolation of sequential timings has been necessary for the calculation of the efficiency in the case of $p = 8$ and 16. From the results, we deduce that the efficiency decreases with the number of processors whereas it increases with the problem. This is due to the parallel LU factorisation that is known to exhibit this natural behaviour. This is further demonstrated through Fig. 5 that displays the ratio of solution and assembly times T_{LU}/T_{ass} as a function of the size for different number of processors, with optimal topology.

An interesting parameter is the number N of higher order terms that is different from one problem to another. Here the same problem is used for various values of N since only the timing is of interest. It is clear that the assembly process becomes more and more important with respect to the solution step as N increases. Indeed triangular reduction only depends on the size n of the problem. Therefore the efficiency is closer to unity for problems with complex source excitation (N large). The worst case relates to $N = 0$, i.e. a constant or linear excitation. Then we are in the case of the classical boundary element method. This is in agreement with results of Fig. 6. An interesting extrapolation of the assembling execution time T_{ass}, with respect to the number of terms is the following:

$$T_{ass}(N) = N[T_{ass}(1) - T_{ass}(0)] + T_{ass}(0)$$

The results of this study lead to some interesting conclusions. First of all, the column-oriented allocation with a blocksize b around 8 seems the optimal topology. From a practical point of view, larger values of b do not significantly affect the performances but lower values must be avoided. In particular, the optimal choice should be followed with care as a low number of higher order terms is to be expected in the problem. As the problem size is still increased above 1024, the ratio T_{LU}/T_{ass} increases so that the global efficiency

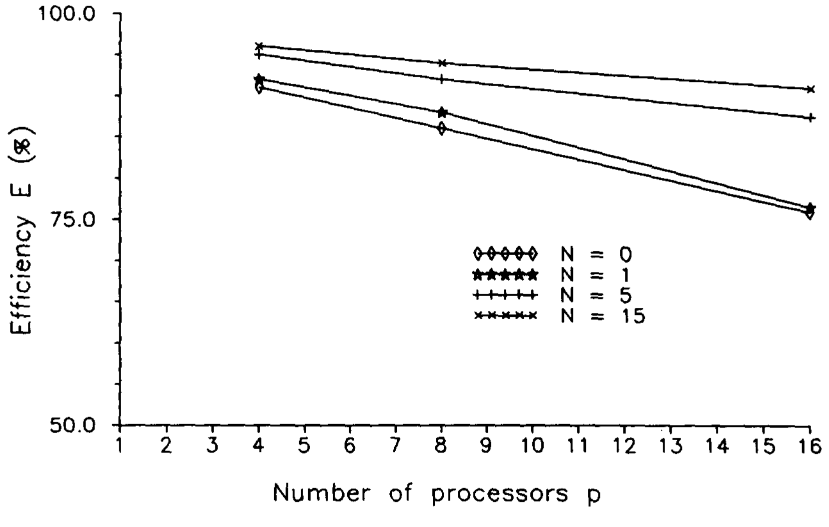

Fig. 6. Efficiency vs number of processors ($n = 512$, $b = 8$, $p_r = 1$).

becomes further influenced by the solution step. On the other hand, the LU efficiency becomes better since the size is larger.

6 CONCLUSION

Our paper was devoted to the parallel implementation of MR-BEM on a coarse-grain distributed memory MIMD architecture. Different data distributions on a grid of processors were considered and compared with respect to the problem size, the number of processors and the number of higher order terms. The implementation has been performed in standard Fortran 77, using BLACS and ScaLAPACK libraries for communication and computation, respectively. Performance results have been conducted on an Intel Paragon machine with 24 processors. A simple test-case was considered. The grid topology has been the basis of the study. It is deduced that the block column-oriented allocation with a blocksize b around 8 seems the best topology whatever the size of the problem may be. The choice of optimal topology is important for the parallel treatment of problems with smooth body force, that is to say when the number of higher order terms is low.

ACKNOWLEDGEMENT

The author would like to thank the CDCSP (Centre pour le Développement due Calcul Scientifique Parallèle) of the University Claude Bernard Lyon 1 (France) for providing access and facilities to their Intel Paragon.

REFERENCES

1. Nardini, D. & Brebbia, C. A. A new approach to free vibration analysis using boundary elements. In *Boundary Element Methods in Engineering*, ed C. A. Brebbia. Springer-Verlag, Berlin, 1982.
2. Yamada, T. & Wrobel, L. C. A new approach to magnetic field analysis by the dual reciprocity boundary element method. *International Journal of Numerical Methods in Engineering*, 1993, **36**, 2073–85.
3. Neves, A. C. & Brebbia, C. A. The multiple reciprocity boundary elements method in elasticity: a new approach for transforming domain integrals to the boundary. *International Journal for Numerical Methods in Engineering*, 1991, **31**, 709–27.
4. Nowak, A. J. & Brebbia, C. A. Numerical verification of the multiple reciprocity method for linear potential problems with body forces. *Engineering Analysis with Boundary Elements*, 1992, **10**, 259–66.
5. Nowak, A. J. Recent developments in MRM — solving problems with nonlinear source term. In *Advances in Computer Methods for Heat Transfer III*, ed. L. C. Wrobel. Computational Mechanics Publication, Southampton, 1994, pp. 497–512.
6. Symm, G. T. Boundary elements on a distributed array processor. *Engineering Analysis with Boundary Elements*, 1984, **1**, 162–5.
7. Davies, A. J. Parallel algorithms for the boundary element method. In *Boundary Element Technology VIII*, ed. H. Pina & C. A. Brebbia. Computational Mechanics Publications, Southampton, 1993, pp. 303–12.
8. Davies, A. J. Massively parallel computing and the boundary element method. In *Application of Supercomputing in Engineering III*, ed. C. A. Brebbia & H. Power. Computational Mechanics Publications, Southampton, 1993, pp. 531–46.
9. Bryant, C. F., Roberts, M. H. & Trowbridge, C. W. Implementing a boundary element method on a transputer system. *IEEE Transactions on Magnetics*, 1990, **26**(2), 819–22.
10. Lobry, J., Broche, C. & Trécat, J. Parallelization of the BEM on transputers. In *High-performance Computing in Engineering, Vol. 2*, ed. H. Power & C. A. Brebbia. Computational Mechanics Publications, Southampton, 1995, pp. 281–307.
11. Davies, A. J. A parallel implementation of the dual reciprocity boundary element method. In *Boundary Element XIV, Vol. 2.*, ed. C. A. Brebbia, J. Dominguez & F. Paris. Computational Mechanics Publications, Southampton, 1992, pp. 613–26.
12. Dongarra, J. J. & Whaley, R. C. A User's guide to the BLACS v1.0, Technical Report LAPACK # 94, University of Tennessee, 1995.
13. Choi, J. *et al.* ScaLAPACK: A portable linear algebra for distributed memory computers — design issues and performances. Technical Report LAPACK # 95, University of Tennessee, 1995.
14. Brebbia, C. A. & Dominguez, J. *Boundary Elements — an Introductory Course*, 2nd edn. CMP, McGraw-Hill, New York, 1992.
15. Nowak, A. J. Convergence studies of the multiple reciprocity method. In *Boundary Element Technology VIII*, ed. H. Pina & C. A. Brebbia. Computational Mechanics Publications, Southampton, 1993, pp. 369–82.
16. Saad, Y. & Schultz, M. H. Data communication in hypercubes. *Journal of Parallel and Distributed Computing*, 1989, **6**, 115–35.
17. Kumar, V. *et al. Introduction to Parallel Computing*. Benjamin Cummings, New York, 1994.
18. Anderson, E. *et al. LAPACK User's Guide*, 2nd edn. SIAM Press, 1991.
19. Choi, J. *et al.* The design and implementation of the ScaLAPACK LU, QR and Cholesky factorization routines. Technical Report, University of Tennessee, 1994.