Contemporary Mathematics
Volume **218**, 1998
B 0-8218-0988-1-03023-5

Solutions of Boundary Element Equations by a Flexible Elimination Process

Choi-Hong Lai and Ke Chen

1. Introduction

Field methods such as finite difference, finite volume or finite element methods are usually applied to solve partial differential equations. Such methods reduce either linear partial differential equations or linearized partial differential equations to a large sparse set of linear equations. However for certain kinds of boundary value problems, boundary element methods have proven to be effective alternatives, especially when dealing with exterior problems. One well-known advantage of boundary element methods is that the dimension of the original problem is reduced by one. The reason is because a differential problem in a domain $\Omega \subset \boldsymbol{R}^m$ can be reformulated as an integral equation problem [**2**] over the underlying boundary $\Gamma = \partial\Omega \subset \boldsymbol{R}^{m-1}$. A number of approximations can then be applied that may lead to boundary element equations.

This paper has two objectives. Firstly, a brief description is given of the sequential boundary element method followed by a possible conversion and its requirements to a distributed algorithm. Secondly, a flexible elimination method [**10**] is used with the distributed algorithm to solve the set of boundary element equations. Such an elimination method has the advantage of not following the usual ordering of the system of equations in a classical elimination procedure such as Gaussian elimination. The technique greatly enhances the intrinsic parallelism of solving dense linear systems of equations. This paper also compares the accuracy of the flexible elimination method with the classical Gauss-Jordan elimination method for two potential flow problems and provides some timing results of the flexible elimination method on a network of SUN workstations using MPI as distribution directives. The paper concludes with an extension of the algorithm to complex systems of linear equations.

1991 *Mathematics Subject Classification.* Primary 65Y05; Secondary 65R20, 65F05.
Key words and phrases. Boundary Elements, Flexible Elimination, Parallel Algorithms, Distributed Computing.
This research is supported by a London Mathematical Society Scheme 4 Grant (Ref 4222).

2. Boundary Element Methods

For certain kinds of boundary value problems such as Laplace's equation, the Helmholtz equation and the linear diffusion equation, boundary element methods have proved to be effective alternatives to field methods. It is particularly true for exterior problems. One typical technique similar to the boundary element method known as the panel element method [**5**], which is a well established practice in aeronautical engineering industry for the design of steady and unsteady subsonic compressible flows over airfoils and other airframe structures, is proved to be a successful tool for engineers.

Consider the exterior Neumann problem in two dimensions using a simple layer logarithmic potential [**6**]. Let $\partial\Omega$ denote the surface of a body which is sufficiently smooth and Ω denote the exterior of the body where a harmonic function is defined in it. Suppose the outward normal derivative ϕ' along $\partial\Omega$ is known, then the solution can be written as

$$\phi(p) = \int_{\partial\Omega} \ln|p-q|\sigma(q)ds \ , \quad p \in \Omega \cup \partial\Omega \tag{1}$$

Suppose the source density σ is distributed on $\partial\Omega$, then it must satisfies the integral

$$\int_{\partial\Omega} \frac{\partial}{\partial n_i} \ln|q_i - q|\sigma(q)ds \ + \ \pi\sigma(q_i) \ = \ \phi'(q_i), \ q_i \in \partial\Omega \tag{2}$$

where n_i is the outward normal at q_i. Suppose now the boundary $\partial\Omega$ is subdivided into elements $\partial\Omega_i$, $i = 1, \cdots, n$, the above integral can be approximated by

$$\sum_{j=1}^{n} \sigma_j \int_{\partial\Omega_i} \frac{\partial}{\partial n_i} \ln|q_i - q_j|ds \ + \ \pi\sigma_i \ = \ \phi_i' \tag{3}$$

where $\sigma_j = \sigma(q_j)$ and $\phi_i' = \phi'(q_i)$. The discretized replacement results in the set of dense linear equations $A\underline{\sigma} = \underline{b}$ where $\underline{\sigma} = [\sigma_1 \cdots \sigma_n]^\top$ and $\underline{b} = [\phi_1' \cdots \phi_n']^\top$. It involves far fewer unknowns than any field method such as finite difference or finite volume methods. Hence a direct method such as Gaussian elimination is usually sufficient for moderate n.

For large n, a direct method can be expensive. Our work here will be a good starting point towards achieving speed up. Iterative methods are alternative approaches that have achieved a varied level of success. That is, efficient iterative solvers can be problem-dependent and preconditioners dependent; refer to [**1**].

2.1. Sequential Algorithm. It is clear that a sequential boundary element method involves two computational functionals, namely, (a) the construction of the dense matrix A and (b) the solution of $A\underline{\sigma} = \underline{b}$, which cannot be computed concurrently. However, one can easily parallelize functional (a) because of the intrinsic parallelism existing in (3). Early work in the parallelization of these integrals can be found in [**3, 8**] and the references therein. The parallelization essentially involves the sharing of the computation of the above integrals amongst a number of processors within a distributed environment. Then functional (b) may be started once functional (a) is completed. There are a large number of literatures on parallel solvers for dense matrices. However on some parallel machines Gauss-Jordan elimination is preferred. One common feature of these parallel implementations is that they primarily rely on the extraction of parallelism

to Gaussian or Gauss-Jordan elimination. The intrinsic sequential behaviour of the two functionals has not been removed to suit modern days distributed computing.

Suppose t_a and T_a denote the sequential and parallel times respectively for computing the integrals and t_s and T_s denote the sequential and parallel times respectively for solving $A\underline{\sigma} = \underline{b}$, then the total sequential computing time, t, is given by

$$t = t_a + t_s \tag{4}$$

and the corresponding parallel computing time, t_p, is given by

$$t_p = T_a + T_s. \tag{5}$$

2.2. A Distributed Algorithm. Divided and conquer type algorithms are usually used to tackle discretized problems in distributed and/or parallel computing environments. There are notably three major classes of divide and conquer type of algorithms, namely, domain decomposition [**7**], problem partitioning [**9**] and functional decomposition [**4**]. The first two type of algorithms concern geometric partitioning of computational domains according to either load balancing or regional physical/numerical behaviour and the latter concerns parallelism in computational functionals. For the present problem, if functionals (a) and (b) could be performed concurrently, then it is possible to achieve

$$t_d = \max\{T_a \ , \ T_s\} \tag{6}$$

It is certainly true that $t_p > t_d$. Hence the key requirement of a new dense matrix solver is that it does not rely on the natural ordering of the equations as required by the classical Gaussian elimination. In other words, the matrix solver should be able to eliminate any equations that have been constructed by the parallel or distributed processing of functional (a) and are made available to the matrix solver in a random ordering. It is important that such distributed algorithm should not affect the accuracy of the solution compared to that obtained by means of the classical Gaussian elimination.

Now suppose the obstacle surface is subdivided into n_p sub-domains such that n/n_p is an integer for simplicity. It is also assumed that each sub-domain is mapped to a processor within the distributed computing environment. Then a distributed algorithm can be given as follow.

Algorithm: A distributed boundary element method.
Notation:- n (number of elements), $\partial\Omega$ (shape of the body),
n_p (number of processors and n/n_p is an integer for simplicity),
i_r (maps the local element numbering r to the global element numbering),
g_i (denotes the arrival ordering of the equations at sub-task 2).

```
sub-task 1 {
    parallel-for p = 1, ··· , n_p
        for r = 1, ··· , n/n_p
            Compute row i := i_r of matrix A;
            Compute element [b]_i of the r.h.s. vector b;
            Non-blocking send of row i and element [b]_i;
        end for
    end parallel-for
```

}end sub-task 1

sub-task 2 {
for $i = 1, \cdots, n$
Block receive row g_i and $[\underline{b}]_{g_i}$ from sub-task 1;
Flexible elimination step (see next Section);
end for
}end sub-task 2

end *Algorithm*

3. A Flexible Elimination Method

The method is based on the concept of orthogonality of vectors. Suppose the coefficients of the ith equation of the system $A\underline{\sigma} = \underline{b}$, where $[A]_{ij} = a_{ij}$, $[\underline{b}]_i = b_i$ and $[\underline{\sigma}]_i = x_i$, are written as the augmented vector $A_i = [a_{i1}a_{i2}\cdots a_{in} - b_i]^\top$, $i = 1, 2, \cdots, n$, then a vector V is said to be the solution of the system provided that the last component of V is unity and that $A_i^\top V = 0$, for all $i = 1, 2, \cdots, n$. Let $\mathcal{C} = \{A_1, A_2, \cdots, A_n\}$. Define the set $\mathbf{C}^{(i)} = \{C_1, C_2, \cdots, C_i \mid \text{a selection of } i \text{ different vectors from } \mathcal{C}\}$ such that $\mathbf{C}^{(i)} = \mathbf{C}^{(i-1)} \bigcup \{C_i\}$ and the set $\mathbf{R}^{(i)}$ as the subspace of dimension $n + 1 - i$ which consists of vectors orthogonal to the vectors in $\mathbf{C}^{(i)}$, for $i = 1, 2, \cdots, n$. It is intuitively obvious that the basis for the $(n+1)$-dimensional subspace $\mathbf{R}^{(0)}$ may be chosen as the natural basis, i.e.

$$\mathbf{V}^{(0)} = \{V_1^{(0)} := [10\cdots0]^\top, \cdots, V_{n+1}^{(0)} := [0\cdots01]\} \tag{7}$$

For each i from 1 to n, linear combinations of a chosen vector from the basis $\mathbf{V}^{(i-1)}$ and one of the remaining vectors from that basis are performed. Such linear combinations are subject to the condition that the resulting vectors are orthogonal to C_i. Therefore for any $C_i \in \mathcal{C} \setminus \mathbf{C}^{(i-1)}$, it is equivalent to the construction of the basis

$$\mathbf{V}^{(i)} = \left\{ V_k^{(i)} \in \mathbf{R}^{(i)} \mid V_k^{(i)} := \alpha_k V_{s(k)}^{(i-1)} + V_{m(k)}^{(i-1)}, C_i^\top V_k^{(i)} = 0 \right\} \tag{8}$$

where $1 \le k \in \mathbf{N} \le n + 1 - i$, $s(k)$ and $m(k) \in \mathbf{N}$ and $s(k) \ne m(k)$. Here $\mathbf{C}^{(0)} = \{\emptyset\}$ is empty. It can be easily shown that

$$\alpha_k = -\frac{C_i^\top V_{m(k)}^{(i-1)}}{C_i^\top V_{s(k)}^{(i-1)}} \tag{9}$$

and that the vector $V_k^{(i)}$ is orthogonal to each of the vectors in $\mathbf{C}^{(i)} \subset \mathcal{C}$. In order to avoid instability of the orthogonalization procedure, the condition $C_i^\top V_{s(k)}^{(i-1)} \ne 0$ must be satisfied. Usually a check may be incorporated in the algorithm to ensure the stability of the method. The dimension of the subspace $\mathbf{R}^{(n)}$ is 1 and the basis $\mathbf{V}^{(n)}$ is orthogonal to every vector in $\mathbf{C}^{(n)} = \mathcal{C}$. Thus the solution of the system $A\underline{\sigma} = \underline{b}$ is constructed [**10**]. It should be noted here that when C_i is chosen as A_i and that if $s(k) = 1$ and $m(k) = k + 1$ then the method is equivalent to a Gauss-Jordan elimination [**10**].

However the choice of $s(k)$ and $m(k)$ can be as flexible as it could be, provided that the condition $s(k) \ne m(k)$ is satisfied. From (8), $n + 1 - i$ pairs of vectors are

chosen from the basis of $\mathbf{V}^{(i-1)}$, such that no two pairs of such vectors are identical, in order to perform the linear combinations. Note that the linear combinations are performed by using the constant α_k, as given by (9), which involves the division of two floating point numbers. Therefore α_k will loose accuracy if the two floating point numbers are of very different orders of magnitudes. One criterion which governs the choice of $s(k)$ and $m(k)$ is to ensure similar order of magnitude of the floating point numbers $C_i^\top V_{m(k)}^{(i-1)}$ and $C_i^\top V_{s(k)}^{(i-1)}$. This involves some additional logical comparison work. It is possible to include tests in an implementation to check that either pivoting is needed or redundant equations have occurred. It can easily be seen that the data structure of the solution vector, i.e. $V_1^{(n)}$, is not affected with various choices of $s(k)$ and $m(k)$. It should be mentioned here that pivoting is equivalent to suitable choices of $s(k)$ and $m(k)$. Therefore the implication is that column pivoting strategy has no effect on the data structure of the solution vector and that the same property applies to row pivoting strategy as long as $s(k)$ is not the same as $m(k)$. More details of these properties and examples can be found in [**10**].

As far as distributed computing is concerned, the flexible choice of s and m is not the key ingredient. However, if we consider the choice of $\mathbf{C}^{(i)}$, we realize that there is no preference in the order of choosing vectors for $\mathbf{C}^{(i)}$ from $\mathcal{C}$. In fact the order of choosing C_i is not important in the present algorithm. The only two criteria governing the choice of C_i is (i) $\mathbf{C}^{(i)}$ consists of a selection of i different vectors from $\mathcal{C}$ and (ii) eqn (9) must be satisfied in order to achieve stability of the algorithm. Hence it is possible to choose $A_{g(i)}$ provided that the map $g : \mathbf{N} \to \mathbf{N}$ is one-to-one and that $1 \leq g(i) \leq n$, $i = 1, \cdots, n$ where $g(i) \neq g(j)$ if $i \neq j$. Such mapping of g implies the order of elimination process is not as rigid as that in a Gauss-Jordan elimination. At any step i, only $C_i \in \mathbf{C}^{(i)} \subset \mathcal{C}$ and $\mathbf{V}^{(i-1)}$ are required in the computation. Therefore the orthogonalization procedure can be completely separated from the knowledge of the set $\mathcal{C} \setminus \mathbf{C}^{(i)}$. In terms of functionals (a) and (b), the requirement for functional (b) to follow functional (a) in a sequential processing is removed. This particular property satisfies the concurrent processing of both functionals (a) and (b) as the necessary requirement described in the previous Section. In terms of the sub-tasks as described in the previous Section, sub-task 2 will be allowed to take and process any equation arriving at its door without jeopardizing the data structure and the stability of the elimination process.

One can also easily see that, by choosing $s(k) = 1$ and $m(k) = k + 1$, the algorithm is particularly suitable for vector calculation and the scalar products involved in the algorithm can be optimized to provided faster timings. We shall investigate the accuracy of the algorithm by using a random number generator to provide a re-ordering function $g(i)$.

4. Examples

For simplicity, potential flows past over obstacles at zero angle of attack are considered. The two obstacles under consideration are (i) an ellipse described by $x^2 + \frac{y^2}{4} = 1$ and (ii) the NACA0012 airfoil. It is assumed that the variables in (3) are normalized with respect to the far field velocity.

Test 1. The algorithm is first implemented as a Gauss-Jordan elimination method by taking $s(k) = 1$ and $m(k) = k + 1$ for the solution of the boundary

TABLE 1. Timings in seconds for solving boundary element equations.

n	$n_p = 1$	$n_p = 4$	$n_p = 8$
640	326	232	152.5
1024	1320	957	814.3

element equations with the natural ordering of the system. Having solved the system of equations for $\sigma(q)$, it is possible to evaluate $\phi(q)$ using (1) and hence the tangential velocity $\mathbf{V}$ along the surface of the obstacle. Pressure coefficients $C_p = 1 - |\mathbf{V}|^2$ [**5, 6**] along the surface of the obstacle can be evaluated. Then a random number generator is used to provide a re-ordering function $g(i)$ as described above. The re-ordering function serves the same functionality as providing rows of matrix coefficients from sub-task 1 to sub-task 2 at a different ordering from the natural ordering according to the element numbering.

Pressure coefficients are obtained for test cases (i) and (ii) by a Gauss-Jordan elimination method using the natural ordering of the equations and the re-ordering of the equations. The maximum errors that have been recorded for both cases are less than 4 decimal places.

Test 2. The algorithm is also run on a network of Sun SPARC Classic workstations at Greenwich. MPI Standard was used to implement the communication. Timings for the solutions of 640 and 1024 unknowns were recorded for both of the obstacle configurations as a sequential process and distributed processes on 4 and 8 workstations. The re-ordering function described above is used here. Table 1 shows the timings on the network. However the speedup in this test is not good because of the heavily used network.

5. Extension to Complex Systems of Equations

Suppose the system $A\underline{\sigma} = \underline{b}$ is a complex system such that $A = A_1 + iA_2$, $\underline{\sigma} = \underline{\sigma}_1 + i\underline{\sigma}_2$ and $\underline{b} = \underline{b}_1 + i\underline{b}_2$. The complex system can be re-written as the following real system

$$\begin{bmatrix} A_1 & -A_2 \\ A_2 & A_1 \end{bmatrix} \begin{bmatrix} \underline{\sigma}_1 \\ \underline{\sigma}_2 \end{bmatrix} = \begin{bmatrix} \underline{b}_1 \\ \underline{b}_2 \end{bmatrix}$$

Since the ordering of the system in the flexible elimination algorithm does not affect the solution, once the $g(i)$th equation is constructed and is made available to sub-task 2, the two equations

$$[\ [A_1]_{g(i)}\ [-A_2]_{g(i)}\ [-\underline{b}_1]_{g(i)}\ ]$$

$$[\ [A_2]_{g(i)}\ [A_1]_{g(i)}\ [-\underline{b}_2]_{g(i)}\ ]$$

can be used immediately into the construction of the new basis as described previously in (8).

6. Conclusions

A distributed algorithm for boundary element methods is discussed. In order to introduce concurrency to boundary element methods at the functional level, one has to employ a flexible elimination method as described in this paper. The accuracy of the flexible elimination method is good and its stability can be ensured easily. Early distributive computing tests show that the method is a suitable candidate for

solving boundary element equations in a distributive environment. The extension to a complex system of equation is straightforward.

References

1. K. Chen, *Preconditioning boundary element equations*, Boundary elements: implementation and analysis of advanced algorithms (W. Hackbusch & G. Wittum, ed.), no. 54, Vieweg, 1996.
2. D. Colton and R. Kress, *Integral equation methods in scattering theory*, Wiley, 1993.
3. A.J. Davies, *The boundary element method on the ICL DAP*, Parallel Computing **8** (1988), 348–353.
4. I. East, *Parallel processing with communicating process architecture*, UCL Press Ltd, London, 1995.
5. J.L. Hess and A.M.O. Smith, *Calculations of potential flow about arbitrary bodies*, Progress in Aeronautical Sciences (D. Kucheman, ed.), no. 8, 1976.
6. M.A. Jawson and G.T. Symm, *Integral equation methods in potential theory and elastostatics*, Academic Press, 1977.
7. D.E. Keyes, Y. Saad, and D.G. Truhlar, *Domain-based parallelism and problem decomposition methods in computational science and engineering*, SIAM, 1995.
8. C-H. Lai, *A parallel panel method for the solution of fluid flow past an aerofoil*, CONPAR88 (CR Jesshope and KD Reinartz, eds.), Cambridge University Press, 1989, pp. 711–718.
9. ———, *A domain decomposition for viscous/inviscid coupling*, Advances in Engineering Software **26** (1995), 151–159.
10. ———, *An extension of Purcell's vector method with applications to panel element equations*, Computers Math. Applic. **33** (1997), 101–114.

School of Computing and Mathematical Sciences, University of Greenwich, Wellington Street, Woolwich, London SE18 6PF, UK

E-mail address: C.H.Lai@greenwich.ac.uk and http://cms1.gre.ac.uk/

Department of Mathematical Sciences, University of Liverpool, Peach Street, Liverpool L69 3BX, UK

E-mail address: k.chen@liverpool.ac.uk and http://www.liv.ac.uk/maths/applied