# Massively Parallel Computation for Acoustical Scattering Problems using Boundary Element Methods

P. Geng, J. T. Oden, and R. A. van de Geijn

May 11, 1995

Texas Institute for Computational and Applied Mathematics
The University of Texas at Austin
Austin, Texas 78712

**Abstract**

This paper is concerned with the parallel implementation of the boundary element method and its application to exterior acoustical scattering problem.

The numerical implementation is based on a Galerkin-type formulation of a Burton-Miller integral equation proposed in [4, 10]. This scheme can avoid sensitive properties of hypersingular integration, simplify the numerical implementation, and improve the quality of the numerical solutions, especially in the range of high frequencies. An *a posteriori* error estimate proposed in [4] is implemented.

In essence, a complete parallel implementation of the boundary element method, including the parallel formation of the linear system, a parallel dense solver, and a parallel algorithm for postprocessing (error estimation), are presented in this paper.

# 1 Introduction

In this study, the possibility of using parallel computing to reduce the time required to solve large-scale structural acoustic problems by boundary element methods is explored. As a major numerical scheme used in engineering analysis, boundary element methods have the potential of reducing the dimensionality of problems and thereby of significantly reducing the number of equations to be solved numerically. For the problem defined on exterior domains, the boundary element method can convert an exterior boundary-value problem into integral equations defined on the interior boundary. Thus, the exterior boundary issues are avoided and the computational cost is further reduced.

A major problem in the application of the boundary element methods is that the system of linear equations generated by this method is usually full and non-symmetric and has to be solved by a direct linear solver. Consequently, $O(n^3)$ cost of dense solvers, in combination with the $O(n^2)$ memory required to store a matrix of dimension $n$ limits the size of problem that can be solved using this method. Furthermore, because of the involvement of singular integrals, the formation of the linear system in the boundary element model is often much more expensive than the formulation of linear systems when other numerical techniques are used. These facts have limited the range of sizes of problems to which the boundary element method can be applied.

The range for which boundary element methods can be used can be greatly expanded through the utilization of massively parallel supercomputers. Dense linear solvers lend themselves particularly well to parallel computing, achieving very good efficiency even when very large numbers of processors are used [3, 6, 19]. The parallel computation can also significantly reduce the computational time in the formation of the linear system.

In this paper, we show how a publically available dense linear solver packag can be used in conjunction with a parallel implementation of the generation of the linear system arising in the boundary element method to solve a practical acoustic scattering problem. A parallel error estimation algorithm is also implemented to estimate the quality of the solution obtained.

This paper is organized as follows. We first describe the mathematical model for the problem of interest and some basic principles in the problem discretization; next we discuss the parallel algorithm on each part of computation; and finally, the experimental results are presented.

# 2 Problem Statement

The problem under consideration is the solution of the exterior acoustical scattering problem. To be precise, we wish to find the pressure distribution $p = p(\mathbf{x})$ on a domain $\Omega^e$ exterior to a closed bounded surface $\Gamma$ satisfying the Helmholtz equation

$$\Delta p + k^2 p = 0 \qquad\qquad \text{in } \Omega^e \tag{1}$$

and the boundary conditions

$$\frac{\partial p}{\partial n}|_\Gamma = Gp + g \tag{2}$$

$$\lim_{\rho\to\infty} \rho\, [\frac{\partial p^s}{\partial \rho} - ikp^s(\mathbf{x})] = 0 \tag{3}$$

where $k = \omega/c$ is the acoustic wavenumber with c and $\omega$, the speed and angular frequency of sound, $\rho = |\mathbf{x}|$ represents the distance between origin and the position $\mathbf{x}$, $n$ denotes the outward normal unit vector on the surface $\Gamma$, $p^s = p - p^{inc}$ is referred by the scattering wave pressure, and $p^{inc}$ is the pressure of incident wave. The boundary condition (3) is also called the Sommerfeld condition and in (2), $G$ and $g$ are two given functions on the boundary $\Gamma$.

The problem defined above can be converted into an integral equation on the boundary $\Gamma$, i.e.,

$$\frac{1}{2}p(\mathbf{x}) - \int_\Gamma [p(\mathbf{y})\frac{\partial \Phi(kr)}{\partial \mathbf{n}(\mathbf{y})} - \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}(\mathbf{y})}\Phi(kr)]ds(\mathbf{y}) = p^{inc}(\mathbf{x}), \tag{4}$$

In (4), $\mathbf{x}$ and $\mathbf{y}$ are two points on the boundary, $r = |\mathbf{y} - \mathbf{x}|$ and $\mathbf{n}(\cdot)$ denotes the outward normal unit vector at the corresponding position, $\Phi$ represents the fundamental solution of Helmholtz equation and is of the form

$$\Phi(kr) = \frac{i}{4}H_0^1(kr) \qquad\qquad \text{in two dimensions}$$

and

$$\Phi(kr) = \frac{\exp(ikr)}{4\pi r} \qquad \text{in three dimensions}$$

where $H_0^1$ is the Hankel function of the first kind of zero order and $i = \sqrt{-1}$. Replacing (1) and (3) with the boundary integral equation can reduce the original problem to one in which only the solutions on the interior boundary $\Gamma$ need to be considered. Unfortunately, the integral formulation of the Helmholtz problem is not perfectly equivalent to the Helmholtz equation (1) on the exterior domain and, at certain frequencies, the so-called forbidden frequencies, the integral operator in (4) becomes singular and the equation is not uniquely solvable.

Several boundary integral formulations have been proposed to overcome the nonuniqueness problem of eq. (4), see [2, 8, 14, 16, 17, 18]. The CHIEF (Combined Helmholtz Integral Equation Formulation) method, proposed by Schenck [16], is the most widely used method in practical computation. In the CHIEF method, the system of linear equations is combined with a few additional equations generated from the Helmholtz integral equation with the source points in the *interior* of the closed boundary. The method can be used to analyze a class of acoustical scattering problems, but because the stability of the method depends strongly on the number and locations of the source points chosen in the computation, the CHIEF method may not be reliable and efficient at high wave numbers.

A method which seems to be valid for all cases was proposed by Burton and Miller in 1971, and in [2], they proved that a complex linear combination of (4) and and a so-called hypersingular integral equation:

$$\frac{1}{2}\frac{\partial p(\mathbf{x})}{\partial \mathbf{n}(\mathbf{x})} - \frac{\partial}{\partial \mathbf{n}(\mathbf{x})}\int_{\Gamma_C} [p(\mathbf{y})\frac{\partial \Phi(kr)}{\partial \mathbf{n}(\mathbf{y})} - \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}(\mathbf{y})}\Phi(kr)]ds(\mathbf{y}) = \frac{\partial p^{inc}(\mathbf{x})}{\partial \mathbf{n}(\mathbf{x})} \qquad (5)$$

is equivalent to the original differential equation. By using (2), the Burton-Miller formulation can be written

$$\frac{1}{2}[1+\alpha G(\mathbf{x})]p(\mathbf{x}) - \int_{\Gamma}[\frac{\partial \Phi(kr)}{\partial \mathbf{n}(\mathbf{y})} - G(\mathbf{y})\Phi(kr)]p(\mathbf{y})ds(\mathbf{y})$$

$$-\alpha\frac{\partial}{\partial \mathbf{n}(\mathbf{x})}\int_{\Gamma}[\frac{\partial \Phi(kr)}{\partial \mathbf{n}(\mathbf{y})} - G(\mathbf{y})\Phi(kr)]p(\mathbf{y})ds(\mathbf{y}) = f(\mathbf{x}) \qquad (6)$$

where $\alpha$ is a complex constant whose imaginary part cannot be zero and

$$\begin{aligned} f(\mathbf{x}) &= p^{inc}(\mathbf{x}) + \int_\Gamma g(\mathbf{y})\Phi(kr)ds(\mathbf{y}) \\ &+ \alpha\frac{\partial}{\partial n(\mathbf{x})}[\, p^{inc}(\mathbf{x}) + \int_\Gamma g(\mathbf{y})\Phi(kr)ds(\mathbf{y})\,] - \frac{\alpha}{2}\, g(\mathbf{x}) \end{aligned}$$

Equation (6) defines the mathematical model used in this investigation to solve the problem of the acoustic wave scattering on the exterior domain $\Omega^e$.

# 3 Discretization of the Integral Equation

A special problem arises in the implementation of Burton-Miller formulation is that the integral

$$\frac{\partial}{\partial \mathbf{n}(\mathbf{x})}\int_\Gamma \frac{\partial\Phi(kr)}{\partial \mathbf{n}(\mathbf{y})}p(\mathbf{y})ds(\mathbf{y}) \tag{7}$$

in (6) cannot be evaluated easily in a meaningful way. Symbolically, it is customary to write (7) in the form

$$\int_\Gamma \frac{\partial^2\Phi(kr)}{\partial \mathbf{n}(\mathbf{x})\partial \mathbf{n}(\mathbf{y})}ds(\mathbf{y}). \tag{8}$$

The integral kernel $\partial^2\Phi(kr)/\partial\mathbf{n}(\mathbf{x})\partial\mathbf{n}(\mathbf{y})$ contains hypersingular term and it is easily verified that as $r \to 0$ in two dimensions

$$\frac{\partial^2\Phi(kr)}{\partial \mathbf{n}(\mathbf{x})\partial \mathbf{n}(\mathbf{y})} = O(\frac{1}{r^2}),$$

and in three dimensions

$$\frac{\partial^2\Phi(kr)}{\partial \mathbf{n}(\mathbf{x})\partial \mathbf{n}(\mathbf{y})} = O(\frac{1}{r^3}).$$

Although the hypersingular integrals can be computed numerically in the sense of the finite part integral ( see [11] and [4] for details ), the implementation for such procedures is complicated and accuracy of its result is usually very poor.

The method used in this study to avoid hypersingular integral kernels associated with (8) was proposed in [10] and [5]. The method requires that the problem be formulated in a weak (Galerkin) approach and is completed in two steps. The first step, proposed by Burton and Miller [2], is based on an important identity derived by Maue [12] and Mitzner [13]:

$$\int_\Gamma p(\mathbf{y}) \frac{\partial^2 \Phi(kr)}{\partial \mathbf{n}(\mathbf{x}) \partial \mathbf{n}(\mathbf{y})} \, ds(\mathbf{y}) =$$

$$\int_\Gamma \{ \, [\, \mathbf{n}(\mathbf{y}) \times \nabla_{\mathbf{y}} p(\mathbf{y}) \,] \cdot [\, \mathbf{n}(\mathbf{x}) \times \nabla_{\mathbf{x}} \Phi(kr) \,]$$

$$+ \, k^2 \, \mathbf{n}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{y}) \, \Phi(kr) \, p(\mathbf{y}) \, \} \, ds(\mathbf{y}). \tag{9}$$

For the second step, multiplying both sides of (9) by a test function $\overline{q}(\mathbf{x})$ ( the conjugate of $q(\mathbf{x})$ ), and integrating the whole equation again over $\Gamma$, yields

$$\int_\Gamma \overline{q}(\mathbf{x}) \int_\Gamma p(\mathbf{y}) \frac{\partial^2 \Phi(kr)}{\partial \mathbf{n}(\mathbf{x}) \partial \mathbf{n}(\mathbf{y})} ds(\mathbf{y}) ds(\mathbf{x}) =$$

$$\int_\Gamma \overline{q}(\mathbf{x}) \int_\Gamma \{ \, [\, \mathbf{n}(\mathbf{y}) \times \nabla_{\mathbf{y}} p(\mathbf{y}) \,] \cdot [\, \mathbf{n}(\mathbf{x}) \times \nabla_{\mathbf{x}} \Phi(kr) \,]$$

$$+ \, k^2 \, \mathbf{n}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{y}) \Phi(kr) p(\mathbf{y}) \, \} \, ds(\mathbf{y}) \, ds(\mathbf{x}) \tag{10}$$

By Stokes' theorem, it can be verified ( see [5] for details ) that

$$\int_\Gamma \overline{q}(\mathbf{x}) \int_\Gamma [\, \mathbf{n}(\mathbf{y}) \times \nabla_{\mathbf{y}} \, p(\mathbf{y}) \,] \cdot [\, \mathbf{n}(\mathbf{x}) \times \nabla_{\mathbf{x}} \Phi(kr) \,] ds(\mathbf{y}) \, ds(\mathbf{x}) =$$

$$- \int_\Gamma \int_\Gamma \Phi(kr) [\, \mathbf{n}(\mathbf{y}) \times \nabla_{\mathbf{y}} p(\mathbf{y}) \,] \cdot [\, \mathbf{n}(\mathbf{x}) \times \nabla_{\mathbf{x}} \overline{q}(\mathbf{x}) \,] \, ds(\mathbf{y}) \, ds(\mathbf{x}),$$

then, (10) can be written as

$$\int_\Gamma \overline{q}(\mathbf{x}) \int_\Gamma p(\mathbf{y}) \frac{\partial^2 \Phi(kr)}{\partial \mathbf{n}(\mathbf{x}) \partial \mathbf{n}(\mathbf{y})} ds(\mathbf{y}) ds(\mathbf{x}) =$$

$$\int_\Gamma \int_\Gamma \Phi(kr) \, \{ - [\, \mathbf{n}(\mathbf{y}) \times \nabla_{\mathbf{y}} p(\mathbf{y}) \,] \cdot [\, \mathbf{n}(\mathbf{x}) \times \nabla_{\mathbf{x}} \overline{q}(\mathbf{x}) \,]$$

$$+ \, k^2 \, \mathbf{n}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{y}) \, \overline{q}(\mathbf{x}) \, p(\mathbf{y}) \, \} \, ds(\mathbf{y}) \, ds(\mathbf{x}). \tag{11}$$

In two dimensions, (11) reduces to

$$\int_\Gamma \overline{q}(\mathbf{x}) \int_\Gamma p(\mathbf{y}) \frac{\partial^2 \Phi(kr)}{\partial \mathbf{n}(\mathbf{x}) \partial \mathbf{n}(\mathbf{y})} ds(\mathbf{y}) ds(\mathbf{x}) =$$

$$\int_\Gamma \int_\Gamma \Phi(kr)[-\frac{\partial \overline{q}(\mathbf{x})}{\partial \tau(\mathbf{x})} \frac{\partial p(\mathbf{y})}{\partial \tau(\mathbf{y})} + k^2 \mathbf{n}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{y}) \overline{q}(\mathbf{x}) p(\mathbf{y})] \; ds(\mathbf{y}) ds(\mathbf{x}) \quad (12)$$

where $\tau(\mathbf{x})$ and $\tau(\mathbf{y})$ are the positive tangential directions at $\mathbf{x}$ and $\mathbf{y}$.

Multiplying both sides of (6) by $\overline{q}(\mathbf{x})$, integrating the whole equation over $\Gamma$ and replacing the corresponding terms by (11) or (12), we obtain the following two- and three-dimensional versions of the variational formulation of (6):

*Find* $p \in H^{\frac{1}{2}}(\Gamma)$ *such that in two dimensions,*

$$\frac{1}{2} \int_\Gamma [\; 1 + \alpha G(\mathbf{x}) \;] \; p(\mathbf{x}) \; \overline{q}(\mathbf{x}) \; ds(\mathbf{x})$$

$$- \int_\Gamma \int_\Gamma [\; \frac{\partial \Phi(kr)}{\partial \mathbf{n}(\mathbf{x})} - G(\mathbf{y}) \Phi(kr) \;] \; p(\mathbf{y}) \; \overline{q}(\mathbf{x}) \; ds(\mathbf{y}) \; ds(\mathbf{x})$$

$$-\alpha \int_\Gamma \int_\Gamma \{ \; -\Phi(kr) \frac{\partial p(\mathbf{y})}{\partial \tau(\mathbf{y})} \frac{\partial \overline{q}(\mathbf{x})}{\partial \tau(\mathbf{x})} + [\; k^2 \tau(\mathbf{x}) \cdot \tau(\mathbf{y}) \Phi(kr) - \frac{\partial \Phi(kr)}{\partial \mathbf{n}(\mathbf{x})} G(\mathbf{y}) \;]$$

$$p(\mathbf{y}) \; \overline{q}(\mathbf{x}) \; \} \; ds(\mathbf{y}) \; ds(\mathbf{x}) = \int_\Gamma f(\mathbf{x}) \; \overline{q}(\mathbf{x}) \; ds(\mathbf{x}), \quad (13)$$

*and in three dimensions,*

$$\frac{1}{2} \int_\Gamma [\; 1 + \alpha G(\mathbf{x}) \;] \; p(\mathbf{x}) \; \overline{q}(\mathbf{x}) \; ds(\mathbf{x})$$

$$- \int_\Gamma \int_\Gamma [\; \frac{\partial \Phi(kr)}{\partial \mathbf{n}(\mathbf{x})} - G(\mathbf{y}) \Phi(kr) \;] \; p(\mathbf{y}) \; \overline{q}(\mathbf{x}) \; ds(\mathbf{y}) \; ds(\mathbf{x})$$

$$-\alpha \int_\Gamma \int_\Gamma \{ \; -\Phi(kr) \; [\; \mathbf{n}(\mathbf{y}) \times \nabla_{\mathbf{y}} p(\mathbf{y}) \;] \cdot [\; \mathbf{n}(\mathbf{x}) \times \nabla_{\mathbf{x}} \overline{q}(\mathbf{x}) \;]$$

$$+ [\; k^2 \; \mathbf{n}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{y}) \Phi(kr) - \frac{\partial \Phi(kr)}{\partial \mathbf{n}(\mathbf{x})} G(\mathbf{y}) \;] \; p(\mathbf{y}) \overline{q}(\mathbf{x}) \; \} \; ds(\mathbf{y}) \; ds(\mathbf{x})$$

$$= \int_\Gamma f(\mathbf{x})\,\overline{q}(\mathbf{x})\,ds(\mathbf{x}) \tag{14}$$

*for any* $q \in H^{\frac{1}{2}}(\Gamma)$.

Equations (13) and (14) contain only the weak singular and regular integrals which can be treated easily in numerical implementations. Approximation of $p$ and $q$ in (13) or (14) by the linear combinations of the same basis functions:

$$p(\mathbf{x}) \;=\; \sum_{i=1}^{N} p^i\ \eta_i(\mathbf{x}) \tag{15}$$

$$q(\mathbf{x}) \;=\; \sum_{i=1}^{N} q^i\ \eta_i(\mathbf{x}) \tag{16}$$

leads to a system of linear equations for $p^i(i = 1, \ldots, N)$, i.e.,

$$\sum_{i=1}^{N} a_{li}\ p^i = b_l, \qquad\qquad l = 1, .., N \tag{17}$$

or

$$\mathbf{Ap} = \mathbf{b} \tag{18}$$

where $\mathbf{A} = \{a_{li}\}_{N\times N}$ is a non-symmetric, complex and dense matrix and $\mathbf{b} = \{b^l\}_{N\times 1}$ are complex vectors. Solving (17) or (18) gives the approximate pressure solution vector $\mathbf{p}$ on the boundary $\Gamma$.

# 4 Parallel Implementation

A complete parallel implementation, including the formation of the linear system and error estimation, and a parallel dense solver package, is discussed in this section.

An important index used in this study to measure the effectiveness or efficiency of a parallel algorithm is $E_{eff}$—the parallel efficiency, which is normally computed by

$$E_{eff} = \frac{\text{speedup}}{P} = \frac{\widehat{T}_1/T_P}{P} \tag{19}$$

where $P$ is the number of processors used, $\widehat{T}_1$ is execution time for the best sequential algorithm on one processor, and $T_P$ is the execution time for the parallel algorithm on $P$ processors. Because of the restriction of memory and other reasons, the value of $\widehat{T}_1$ usually can not be obtained directly by numerical experiment, and therefore, we often have to estimate a value for $\widehat{T}_1$ based on the measured peak execution rate of an individual processor for the equivalent serial computation.

## 4.1 Formation of the Linear System

We here consider the parallel computation of the following two integral operations

$$\int_\Gamma \int_\Gamma \mathbf{B}_1\, p(\mathbf{y})\Phi(\mathbf{x},\mathbf{y})\mathbf{B}_2\, \overline{q}(\mathbf{x}) ds(\mathbf{y}) ds(\mathbf{x}), \tag{20}$$

and

$$\int_\Gamma f(\mathbf{x})\overline{q}(\mathbf{x}) ds(\mathbf{x}). \tag{21}$$

The integrals given in (20) and (21) represent the general computations encountered in the formation of stiffness matrices and load vectors on the boundary element methods with $p$ trial function, $q$ test functions; and $\mathbf{B_1}$, $\mathbf{B_2}$ linear operators applied on trial and test functions; $\Phi$ in (20) is the integral kernel function and the exact form of this function depends on the type of the problem solved.

In practice, these integrals are computed locally on each element i.e.,

$$\int_\Gamma \int_\Gamma \mathbf{B}_1\, p(\mathbf{y})\Phi(\mathbf{x},\mathbf{y})\mathbf{B}_2\, \overline{q}(\mathbf{x}) ds(\mathbf{y}) ds(\mathbf{x}) = \sum_{i,j=1}^{N_1} \int_{\Gamma_i} \int_{\Gamma_j} \mathbf{B}_1\, p(\mathbf{y})\Phi(\mathbf{x},\mathbf{y})\mathbf{B}_2\, \overline{q}(\mathbf{x}) ds(\mathbf{y}) ds(\mathbf{x}), \tag{22}$$

$$\int_\Gamma f(\mathbf{x})\overline{q}(\mathbf{x}) ds(\mathbf{x}) = \sum_{i=1}^{N_e} \int_{\Gamma_i} f(\mathbf{x})\overline{q}(\mathbf{x}) ds(\mathbf{x}) \tag{23}$$

where $\Gamma_i$ $(i = 1, \cdots, N_e)$ represents the element on the boundary and $N_e$ is the total number of elements. To simplify the future discussion, we define

- the integral operation,

$$\int_{\Gamma_i}\int_{\Gamma_j} B_1 p(\mathbf{y})\Phi(\mathbf{x},\mathbf{y})B_2\overline{q}(\mathbf{x})ds(\mathbf{y})ds(\mathbf{x}) \tag{24}$$

  in (22) as **Elemsb**( $i$, $j$, $A^{ij}$ ), and

- the integral operation,

$$\int_{\Gamma_i} f(\mathbf{x})\overline{q}(\mathbf{x})ds(\mathbf{x}) \tag{25}$$

  in (23) as **Elemv**( $i$, $b^i$ )

Here, $A^{ij}_{n_i\times n_j}$ and $b^i_{n_i\times 1}$ are the local stiffness matrix and the local load vector which represent a part of global stiffness matrices and global load vector contributed by the integrals on one or a pair of elements. As noted earlier, the trial and test functions are linear combinations of a group of basis functions, and here we use $n_i$ or $n_j$ to denote the numbers of basis functions which are non-zero on the element $i$ or $j$. Because the special property (local support) of the basis functions used in the boundary element methods, we have

$$n_i \text{ and } n_j \ \ll \ N$$

for any element $i$ or $j$, where $N$ represents the total number of degrees of freedom in the boundary element methods.

In most cases, the integrals involved in **Elemsb** and **Elemv** are regular, the normal Gaussian quadrature method is used to evaluate the integrals on local elements, and the implementation is standard. The singular integrations are involved in the operation of **Elemsb**( $i$, $j$, $A^{ij}$ ) only in case where $i \ = \ j$. In three dimensions, the *Duffy triangular coordinate method* proposed in [5] is used to evaluate the singular integrals. The procedure for the triangular element is briefly described in Fig. 1. Supposing for any singular point $\mathbf{x}$ in the element $K$, $\eta$ is the corresponding point in the master element $\widehat{K}$. The triangle $\widehat{K}$ is divided into three subtriangles with the point $\eta$ constituting the common vertex. Each of the subtriangles is further mapped into a right triangle as shown Fig. 1, and finally the right triangular is mapped into a square S through the transformation of ( the map 3 in Fig. 1 )

$$\begin{aligned} y_1 &= \mu_1 \\ y_2 &= \mu_1\,\mu_2 \end{aligned} \tag{26}$$

The Jacobian of the transformation (26) is $\mu_1$ ( $= r\cos\ \theta,\ , 0^o \ \le\ \theta\ \le\ 45^o$ ) which eliminates the $1/r$ singularity in the original function. Consequently, normal Gauss quadrature integrations over the square S can be used. One point that should be indicated is that because the method involves integration over three triangles rather than one, the computational cost of **Elemsb** with $i\ =\ j$ is much more expensive than that of **Elemsb** with $i\ \neq\ j$. This fact may affect load balancing in parallel computation and will be discussed later.

With the integral operations defined in (24) and (25), the conventional sequential algorithm used in the boundary element methods to form linear systems is as follows:

do 10 $i = 1,\ N_e$ ( first loop through elements )

call **Elemv**( $i,\ b^i$ ) ( computing local load vector )

assemble local vector $b^i$ ( adding $b_i$ to global vector )

do 20 $j$=1, $N_e$ ( second loop through elements )

call **Elemsb**( $i,\ j,\ A^{ij}$ ) ( computing local stiffness matrix )

assemble $A^{ij}$ ( adding $A^{ij}$ to global matrix )

20 continue

10 continue

Clearly, it is not necessary to perform **Elemv** and **Elemsb** in a sequential order. Thus, we can consider the integrals on the local elements as the basic subtasks to be scheduled on the processors and apply parallelism within the loop over the elements.

To perform the computation in parallel, the $N \times N$ matrix $\mathbf{A}$ must be partition among all processors, allowing full utilization of the aggregate memory. To achieve this, $P$ processors are viewed as a $R \times C$ logical mesh, and each processor is identified by a pair of integers $(s, t)$ where $P = R \times C$ and $0 \le s \le R-1$ and $0 \le t \le C-1$. As shown in (27), we now assign $m$ by $n$ submatrix $\mathbf{A}_{st}$ to each processor $(s,\ t)$ where $m = N/R$ and $n = N/C$.

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{0,0} & \cdots & \mathbf{A}_{0,C-1} \\ \vdots & & \vdots \\ \mathbf{A}_{R-1,0} & \cdots & \mathbf{A}_{C-1,R-1} \end{bmatrix} \tag{27}$$

Two algorithms for parallelizing the filling of matrix block among processors are given by

- the parallel algorithm based on division of the number of degrees of freedom (the first algorithm), and
- the parallel algorithm based on division of the number of elements (the second algorithm).

In the first, each processors is assigned the responsibility of forming one block of $A$, looping over all elements to find elements that contribute. In the second algorithm, it is the elements that are divided among the processors and the matrix is viewed as a global matrix to which each element contributes.

The first method has some clear advantages and disadvantages: Blocking the matrix as described is exactly how the parallel dense solver, described in the next subsection, expects the data to be distributed. Moreover, no communication is necessary. However, considerable redundant computation can be expected, especially for 3D problems. Moreover, the work associated with blocks around the diagonal of the matrix requires more operations of **Elemsb**( $i$, $j$, $A^{ij}$ ) with $i = j$ than the work associated with blocks far from the diagonal does. As a result, if the logic mesh is not in the form of $1 \times P$ or $P \times 1$, the first method can create some load imbalance.

The second method avoids redundant computation as well as load imbalance, if approximately the same number of elements are assigned to each processor. However, a global matrix is required, which automatically means communication is necessary. In practice, we can avoid this problem by storing the matrix on disk, allowing each processor to contribute. The price paid is a considerable expense in I/O overhead, however this is more than offset by the convenience of the method, and the savings in redundant computation.

The first algorithm can usually yield a very high parallel efficiency for the problem on one-dimensional mesh, i.e., one-dimensional FEM problem or two-dimensional BEM problem. On an one-dimensional mesh, the degrees of freedom for one element are shared by at most two processors. On a $P$ processor machine, if the computational cost of **Elemsb**( $i$, $j$, $A^{ij}$ ) for different $i$ and $j$ is roughly same, or the logic mesh is in the form of $1 \times P$ or $P \times 1$, the parallel efficiency can be estimated as

$$
\begin{aligned}
E_{eff} &= \frac{(\text{No. of Elements})^2}{P(\frac{\text{No. of Elements}}{R} + 2R) \times (\frac{\text{No. of Elements}}{C} + 2C)}. \\
&\approx \frac{1}{1 + \frac{2(R^2 + C^2)}{\text{No. of Elements}}}
\end{aligned}
$$

Usually, we have No. of Elements $\gg 2(R^2 + C^2)$, and the parallel efficiency can be very close to 100 percent.

For two- and three-dimensional meshes, each degree of freedom on average is shared by 4 to 20 elements. In such problems, after assigning global degrees of freedom to processors, we often find that the local degrees of freedom on a single element are shared by 5 to 6 different processors; i.e., on average, the same local stiffness matrix and load vector must be computed 5 to 6 times and the parallel efficiency then will be only about 15-20 percent. Normally, the computation of the local element stiffness matrix and load vector is relatively cheap; even with a 15 or 20 percent parallel efficiency, the total cost in formation of the linear system is still insignificant compared to the cost of solving the linear system. Thus, for distributed memory computer, the first algorithm is a viable candidate.

However, because of the involvement of hypersingular integrals, the computation of the local stiffness matrix for the three-dimensional Burton-Miller integral formulation is very expensive and a parallel efficiency much greater than 20 percent is needed in practical computation. Because it is possible to obtain a much better parallel efficiency through the second algorithm than the first one if the computation on the local stiffness matrix or load vector is very expensive, the second algorithm is also employed in this study for the generation of linear system in the three-dimensional boundary element method.

## 4.2 A Parallel Dense Matrix Solver Routine

The parallel dense solver package used for this study was originally developed to solve electromagnetic scattering problems. Implementation details can be found in Cwik, van de Geijn, and Patterson [3]. Theoretical properties are investigated in the paper by Dongarra, van de Geijn, and Walker [6]. We will restrict our discussion in this section to a description of the functionality and major properties.

Dense linear solve routines for nonsymmetric problems generally perform the following steps: Given the linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

matrix $\mathbf{A}$ is first factored into the product of lower and uppertriangular matrices $\mathbf{L}$ and $\mathbf{U}$:

$$\mathbf{P}\,\mathbf{A} = \mathbf{L}\,\mathbf{U}$$

where $\mathbf{P}$ represents the accumulation of permutations required for numerical stability. Next, intermediary result $\mathbf{y}$ is computed by solving $\mathbf{L}\mathbf{y} = \mathbf{P}\mathbf{b}$, often referred to as forward substitution, followed by the final computation of $\mathbf{x}$ from the triangular system $\mathbf{U}\mathbf{x} = \mathbf{y}$, referred to as backward substitution. Robust packages like LINPACK [7] and LAPACK [1] also provide an estimate of the condition number of the linear system, which gives insight into how much errors in the right-hand-side are amplified in the solution.

A major obstacle to efficient implemention of algorithms on high performance microprocessors that are at the heart of high performance parallel systems is the discrepancy between processor speed, which allows a high rate of computation, and main memory bandwidth, which severely limits the rate at which data can flow to the processor. Microprocessors are equipped with a small amount of high-speed memory, the cache, to overcome this bottleneck. However, exploiting the cache requires the ratio of arithmetic operations and the amount of data on which computation is performed to be large. I.e., a cache only gives an advantage if there is a lot of data reuse in a computation. A feature of many dense linear algebra algorithms, exploited by LAPACK, is to reformulate dense linear algebra algorithms in terms of matrix-matrix computations. For example, in a matrix-matrix operation involving matrices of dimension $n$, $O(n^3)$ operations are performed on $O(n^2)$ data, yielding a very favorable ratio between computation and data access.

Turning now to the parallel implementation of the LU factorization, we list this following criteria that must be met by a good parallel implementation:

**High utilization of each processor.** The algorithm must start by exploiting the efficiency of matrix-matrix operations on each processing node.

**High parallel efficiency.** When $P$ processors are used, we would like to achieve close to a factor $P$ speedup over the sequential implementation.

**Scalability.** A scalable parallel algorithm has the property that if reasonable efficiency is attained on a small number of processors, a similar efficiency can be attained when a large number of processors are used. However, it is allowed to increase the size of the problem being solved, provided the combined memories of the processors can hold the larger problem. Since total memory grows linearly with the number of processors, this condition means that it must be possible to maintain efficiency by increasing the problem size so that the amount of data involved also grows linearly with the number of processors.

**Robustness.** The parallel algorithm must exhibit the same numerical stability properties as the sequential algorithm.

We will limit our discussion of the parallel dense solver by summarizing that in the previously mentioned related papers [3, 6] we show the implementation to meet all the above listed criteria. In particular, the problem size $n$ must approximately grow with the square-root of the number of processors $p$, if efficiency is to be maintained. This meets our scalability criteria, since memory requirements are $O(n^2)$.

## 4.3 Parallel Computation of the Error Estimation

The error estimation technique studied here for the coupled problems belongs to a general family of schemes called element residual methods. In such methods, a global error estimate is computed as the sum of local error estimates which can be computed independently on each element and consequently, the computation can be trivially parallelized.

As noted in [4], the $L^2$-norm of the residual of the normal Helmholtz integral equation can produce very accurate global and local error estimates for the Galerkin approximate solution of the Burton-Miller formulation. The residual of the normal Helmholtz integral equation is computed by the formulation

$$
\begin{aligned}
r^h(\mathbf{x}) &= 2f(\mathbf{x}) - \{p^h(\mathbf{x}) \\
&-2\int_\Gamma [\frac{\partial \Phi(k|\mathbf{y}-\mathbf{x}|)}{\mathbf{n}(\mathbf{y})} - G(\mathbf{y})\Phi(k|\mathbf{y}-\mathbf{x}|)]p^h(\mathbf{y})ds(\mathbf{y})\} \qquad (28)
\end{aligned}
$$

and then the $L^2$-norm of residual $r^h$ is computed by

$$
\| r^h \|_0^2 = \int_\Gamma r^h \overline{r}^h ds(\mathbf{y}) \qquad (29)
$$

where $p^h$ is the Galerkin approximate solution of the Burton-Miller formulation (6).

In practice, the computation of the integral in (29) is performed on each single element, and we can rewrite (29) as

$$
\| r^h \|_0^2 = \sum_{i=1}^{N_e} \int_{\Gamma_i} r^h \overline{r^h} ds(\mathbf{y}) \qquad (30)
$$

where $\Gamma_i$ $(i = 1, \cdots, N_e)$ represents the set of boundary elements and $N_e$ is the total number of elements. If $\eta_i$ denotes the local error estimation for the element $\Gamma_i$, i.e.,

$$
\eta_i^2 = \int_{\Gamma_i} r^h \overline{r^h} ds\mathbf{y}, \qquad (31)
$$

then

$$
\| r^h \|_0^2 = \sum_{i=1}^{N_e} \eta_i^2. \qquad (32)
$$

Clearly, this computation need not be performed sequentially. In most cases, the cost of computing a single $\eta_i$ for different $i$ is roughly the same; thus simply distributing $i$ (for $\eta_i$) uniformly among processors will lead to a near-optimal efficiency, very close to 1.

# 5 Numerical Experiments

The performance of the parallel algorithms developed for the two- and three-dimensional boundary element methods is tested in this section. In particular, the parallel efficiencies for the following three parts of the computation are of interest

- formation of the linear system (stiffness matrix and load vector)
- $LU$ factorization and forward and backward substitutions, and
- an *a posteriori* error estimation.

All tests conducted in this section were performed on the Intel iPSC/860 which currently has 32 parallel processors with 8 Mbytes memory per processor and the Intel touchstone Delta which has 512 parallel processors with 16 Mbytes memory per processor. On these two machines, the dense matrix solver can solve a double precision complex linear system with a maximum size of around 3,000 and 16,000, respectively.

## 5.1 The Test Problems

The acoustic scattering problem ( see Fig.2 ) for a plane wave impinging on a cylindrical body ( in two dimensions ) or a sphere ( in three-dimensions ) with the boundary condition:

$$\frac{\partial p}{\partial \mathbf{n}} = \varepsilon p \qquad \text{on } \Gamma \tag{33}$$

is used here as test problems. Here $\varepsilon$ can be a real or complex constant. Clearly, for $\varepsilon = 0$, we have a problem of acoustic wave scattering on a rigid cylinder.

According to [9], the analytic solution of the test problem at a point $\widehat{x}$ ( $\rho = |\widehat{x}| \geq a$ ) is of the form

$$p = p^{inc} + p^{s} \tag{34}$$

where the incident pressure is

$$p^{inc} = P_{inc} \exp(ik\,\rho\, cos\,\theta) = P_{inc} \exp(ikx) \tag{35}$$

with a constant $P_{inc}$, and the scattered pressure $p^s$ is evaluated by the series

$$p^s(\rho,\theta) = \begin{cases} \sum_{m=0}^{\infty} B_m H_m(k\rho)\cos(m\theta) & \text{in two dimensions} \\ \sum_{m=0}^{\infty} B_m h_m(k\rho) P_m(\cos(m\theta)) & \text{in three dimensions} \end{cases} \tag{36}$$

and $a$ is the radius of the cylinder or sphere. In (36), $H_m$ and $h_m$ represent the Hankel function and spherical Hankel function of the first kind of order m, respectively, $P_m$ is Legendre polynomials, and as noted earlier, $k$ is the wave number. In two dimensions, the coefficients $B_m$ are given by

$$B_m = -P_{inc} b_m (-1)^{\frac{m}{2}} \frac{kJ'_m(ka) - \varepsilon J_m(ka)}{kH'_m(ka) - \varepsilon H_m(ka)} \tag{37}$$

where $J_m$ is the Bessel function of the first kind of order $m$, and

$$b_0 = 1 \text{ and } b_m = 2 \ (m = 1, 2, 3, \ldots); \tag{38}$$

and in three dimensions, the coefficients $B_m$ are computed through

$$B_m = -P_{inc}(2m+1)(-1)^{\frac{m}{2}} \frac{kj'_m(ka) - \varepsilon j_m(ka)}{kh'_m(ka) - \varepsilon h_m(ka)} \tag{39}$$

where $j_m$ is the spherical Bessel function of the first kind of order $m$. The special functions, $H_m$, $h_m$, $J_m$ and $j_m$ are computed by the routines provided in [15].

## 5.2 Experiments on 2D BEM Problems

**Performance**

The test problem is solved with different numbers of processors $P$ to test the performance of the parallel algorithms. The number of processors, elements and degrees of freedom are listed in Table 1. In all cases, a uniform mesh of quadratic elements is used. For all tests performed in this example, the input data are:

$$\epsilon = 1, P_{inc} = 1, a = 1, \text{ and } ka = 100.$$

The formation of the linear system, error estimation, and solution of linear triangular system are $O(N^2)$ computations, thus $\widehat{T}_1$ in (19) is estimated by

$$\widehat{T}_1 = T_1 \frac{N_P^2}{N_1^2}$$

and so

$$E_{eff} = \frac{T_1}{PT_P}\frac{N_P^2}{N_1^2} \tag{40}$$

where, $T_P$ and $N_P$ represents the elapsed time and the size of problem when the problem is solved in $P$ processors, and $T_1$ is the corresponding computational time in the single processor with the problem size $N_1 = 512$ (256 quadratic elements). In this experiment, we choose $N_p^2/PN_1^2 = 1$, and so the parallel efficiencies can simply estimated by

$$E_{eff} = \frac{T_1}{T_P}. \tag{41}$$

The $LU$ factorization is a $O(N^3)$ computation and, thus, the parallel efficiency is estimated here by

$$E_{eff} = \frac{T_1}{PT_P}\frac{N_P^3}{N_1^3}.$$

For the formation of the linear system, the first algorithm— the parallel algorithm based on division of the number of degrees of freedom, is used. As mentioned, due to singular integrals, the computational work associated with coefficients around the diagonal of the matrix is more than those associated with blocks far from the diagonal, thus some load imbalance is created unless the logical mesh of $1 \times P$ or $P \times 1$ is used. Table 2 lists the computational time and estimated parallel efficiency on each step of computation for $P = 1 \times P$ logical mesh. As shown in Table 2, essentially perfect parallel efficiencies are obtained for the generation of the linear system.

Since the iPSC/860 is physically a hypercube, we always have that $P = 2^m$ where $m \geq 0$ is an integer and, as discussed in [6], the parallel dense solver attains a better performance if the logical mesh is approximately square, i.e.

$$R \times C = \begin{cases} 2^{\frac{m}{2}} \times 2^{\frac{m}{2}} & \text{if m is even} \\ 2^{\frac{m+1}{2}} \times 2^{\frac{m-1}{2}} \text{ or } 2^{\frac{m-1}{2}} \times 2^{\frac{m+1}{2}} & \text{if m is odd.} \end{cases} \tag{42}$$

Table 3 lists the computational time and estimated parallel efficiency on each step of computation for the same input data as Table 2 but with the logical mesh of processors given as an approximately square mesh. As shown in

Table 3, because of load imbalance, the parallel efficiency in the formation of linear system decreases a little here, but a much better parallel efficiency for the dense solver ($LU$ factorization and forward and backward substitutions) is obtained. As shown Tables 2 and 3, the parallel efficiencies in the computation of the error estimation are almost 100 percent, as expected.

For the complex linear system, the matrix factorization requires $8/3N^3$ double precision arithmetic operations and solving the linear triangular system requires around $16N^2$ double precision arithmetic operations. The computational speed per processor attained by the dense solver is thus calculated by

$$\frac{\text{Mflops}}{\text{per processor}} = \frac{\frac{8}{3}N^3 + 16N^2}{P \times \text{computational time}} \times 10^{-6}. \tag{43}$$

and is listed in Table 4. Notice that for problems that occupy the same amount of memory per processor, the performance decreases moderately with the number of processors.

The largest matrix system($N = 2,896$) took less than 2 minutes to solve on 32 processors of the Intel iPSC/860 machine. The peak performance of each individual is in the 35-40 Mflops range for this kind of problem, and the dense solver here achieve better than 40 percent of peak for this part of the computation. If larger problems were solved, e.g. on an iPSC/860 with 16 Mbytes per processor, better efficiencies would be observed.

**Physical Results**

To demonstrate the applicability of the present method to problems with high wave number, the test problem is solved for $ka = 400$ and $\epsilon = 1$ by 16 processors of the Intel iPSC/860. A uniform mesh of 1,024 quadratic elements with a total degrees of freedom of 2,048 is used in the computation.

Figure 3 compares the numerical solution and corresponding exact solution (computed through (34)-(39)). The corresponding $L^2$-norm of relative error $\| \hat{e} \|_0$ and the estimated error $\| \hat{r} \|_0$, are also given. Here, $\| \hat{e} \|_0$ and $\| \hat{r} \|_0$ are computed, respectively, by

$$\| \hat{e}^h \|_0^2 = \frac{\int_\Gamma (p^h - p_e)\overline{(p^h - p_e)}\; ds(\Gamma)}{\int_\Gamma p_e \overline{p_e}\; ds(\Gamma)}$$

and

$$\| \hat{r} \|_0^2 = \frac{\int_\Gamma r^h \overline{r^h}\; ds(\Gamma)}{\int_\Gamma p_e \overline{p_e}\; ds(\Gamma)};$$

where $p_e$ represents the exact solution, $p^h$ is the numerical solution and $r^h$ is the residual defined in (28). The result illustrated in Fig. 3 shows a good correlation between the numerical solutions and analytic solution.

Figure 4 further compares the local error $\| e^h \|_i$ with the estimated local error $\| r^h \|_i$ where $\| e^h \|_i$ and $\| r^h \|_i$ are computed, respectively, by

$$\| e^h \|_i = \int_{\Gamma_i} (p^h - p_e)\overline{(p^h - p_e)} ds\mathbf{y}$$

and

$$\| r^h \|_i = \int_{\Gamma_i} r^h \overline{r^h} ds\mathbf{y}$$

for any given element $\Gamma_i$. As shown in Fig. 4, the estimated error is almost equal to the true error and this fact indicates that the error estimation used in the present study possesses good global and local properties.

## 5.3 Experiments on 3D BEM Problems

The numerical experiments performed in the preceding section indicate that the first algorithm can produce satisfactory performance in forming the stiffness matrix for the two-dimensional BEM problem. However, because the computation of linear system for three-dimensional problem is quite expensive in this study, the second algorithm—the parallel algorithm based on division of the number of elements, should have superior performance for the 3D case. To verify this conjecture, both algorithms are implemented and their performances are compared experimentally.

The following input data are selected for the test problem in three-dimensions:

$$P_{inc} = 1, a = 1, \text{ and } ka = 10.$$

The test is still performed in the Intel iPSC/860 machine with 16 processors, three of which possess a CFS (Concurrent File System) disk I/O capacity. A uniform mesh of 1,024 quadratic triangular elements on the spherical surface is used with a total number of degrees of freedom of 2,050.

Table 5 compares the computational times in the formation of the linear systems using both algorithms and the corresponding parallel efficiencies. The experimental result clearly indicates that the second algorithm is much more efficient than the first. As shown in Table 5, for the second algorithm, the time used to assemble matrix in CFS disk is about 464 seconds. For the mesh of quadratic triangular elements, on average, each degree of freedom is shared by four elements, thus, each column of the matrix must be read or written eight times to complete assembling. The I/O speed attained by 3 I/O processors in the Intel iPSC/860 machine can be estimated by

$$\frac{2,050^2 \times 16 \times 8}{464} = 1.16 \text{ Mbytes/second.}$$

One thing which should be indicated is that the algorithm used to access disk in this study is relatively primitive, and a much better disk accessing speed may be obtained if this algorithm is fully optimized.

The parallel efficiencies in Table 5 are also computed by (40), where $T_1$ is the computational time of the formation of linear system for the test problem solved by 256 quadratic elements and 514 degrees of freedom in one processors, and from the experiment, we have that $T_1 = 3,001.3$ seconds.

Further tests were performed on the Touchstone Delta which possesses 512 processors, each of which has an i860 processor, like the iPSC/860. The available local memory of each single processor on the Touchstone Delta machine is about 16 Mbytes of which 8 or 9 Mbytes can be used to store the data of the linear system. Thus, the largest problem which can be stored using 512 processors is about

$$\sqrt{\frac{8 \times 10^6 \times 512}{16}} \approx 16,000$$

For $k = 20$ and $\partial p/\partial n = 0$ (rigid scattering), the test problem is solved for varying numbers of degrees of freedom and numbers of processors. In all cases, a uniform mesh of quadratic elements is used and the sizes of problem vary from 514 (256 elements) to 8,194 (4,196 elements).

Table 6 presents the computational times and estimated parallel efficiencies for the different sizes of problem by using different number of processors in Touchstone Delta to form the linear system. The parallel efficiency here is estimated by (40), i.e.,

$$E_{eff} = \frac{T_1}{PT_P}\frac{N_P^2}{N_1^2}.$$

where $N_P$ and $T_P$ represent the size of stiffness matrix and the computational times on $P$ processors, respectively. As shown in Table 6, a parallel efficiency of more than 50 percents is attained even when 512 processors are used.

Table 7 shows overall performance attained by the dense solver. The performance here is still expressed in the form of the computational speed per processor which is also computed by (43). Because of the larger local memory capacity and faster communication speed, the dense solver attains a better performance in the Touchstone Delta than in the iPSC/860 machine, and the computational speed per processor can exceed 23 Mflops in the best case. For the matrix operation, the peak performance of each individual processor here is still in the range of 35-40 Mflops, and so the dense solver is achieving better than 50% of peak for this part of computation. Considering the fact that the condition number estimation, and forward and backward substitution usually do not perform well on parallel computers, the performance obtained by the dense solver in this example is very good.

For the physical result, we show the pressure distribution for a plane acoustic wave impinging on a rigid sphere with $ka = 20$ in Fig. 5. The problem is solved by the mesh of 1,568 quadratic elements with total degrees of freedom of 3,138, and we here have that the $L^2$ relative error is 4.9 % and the corresponding estimated error is 5.2 %.

# 6 Summary and Future Work

A general parallel procedure for the boundary element methods is developed in this work. All of the parallel algorithms yielded good parallel efficiencies in a series of numerical experiments.

The parallel dense solver seems to be extremely efficient in the practical computation. Numerical experiments indicate that a double precision complex system composed of 8,194 linear equations can be solved in five minutes by using 256 processors of the Touchstone Delta. The experiment performed in [3] indicated that by using 512 processors, a similar linear system with 16,000 unknowns was solved in less than 20 minutes.

By using the Burton-Miller formulation, the integral equation becomes uniquely solvable on the exterior domain, and a satisfied numerical solution can be obtained even at range of very high frequencies. However, because the involvement of an extra hypersingular integral equation, the cost in the generation of linear system increases significantly. Thus, a major concern in future development is to reduce the cost in the generation of linear system. This can possibly be done by simplifying certain integral procedures involved in the computation of the local stiffness matrices.

The parallel dense solver used in this research is an in-core algorithm for which the entire linear system is stored in the inner memory of different processors; thus, the size of the problem solved is limited by the memory capacity of the machine. Because of the large disk storage available on parallel computer systems, a natural extension of the current work is to develop an out-of-core dense solver. For the out-of-core solver, the linear system is stored on disks and the local memory of each processor keeps only the matrix block needed in the current computation; thus, there is virtually no limitation on the problem size. At present, an out-of-core dense solver is under development and should be available soon.

**Acknowledgement**: The support of the Office of Naval Research of this work under Contract N00014-89-J-3109 is gratefully acknowledged. This research was performed in part using the Intel Touchstone Delta System operated by the California Institute of Technology on behalf of the Concurrent Supercomputing Consortium. Access to this facility was provided by Intel Supercomputer Systems Division and the California Institute of Technology. The parallel dense linear solver was developed under contract to the Jet Propulsion Laboratory, and resides in the public domain. To obtain this code, send a message to `rvdg@cs.utexas.edu`.

# References


[1] E. Anderson, Z. Bai, J. Demmel, J. Dongarra, J. DuCroz, A. Greenhaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. Sorensen 1992 **LAPACK users' guide**. Philadelphia: *SIAM*.

[2] A. J. Burton and G. F. Miller 1971 *Proc. Roy. Soc.* London Ser. A323, 201-210. The application of integral equation methods to the numerical solution of some exterior boundary-value problems.

[3] T. Cwik, R. A. van de Geijn and J. Patterson 1994 *J. Opt. Soc. Am. A* **11**, 1538-1545. Application of Massively Parallel Computation to Integral Equation Models of Electromagnetic Scattering.

[4] L. Demkowicz, J.T. Oden, M. Ainsworth and P. Geng 1991 *Journal of Computational and Applied Mathematics* **36**, 29-63. Solution of elastic scattering problems in linear acoustics using h-p boundary element methods.

[5] L. Demkowicz, A. Karafiat and J.T. Oden 1992 *Comp. Meths. Appl. Mech. Engrg.* **101**, 251-282. Solution of elastic scattering problems in linear acoustics using *h-p* boundary element method.

[6] J. Dongarra, R. A. van de Geijn, and D. Walker 1994 *Journal of Parallel and Distributed Computing* **22**, 523-537. Scalability Issues Affecting the Design of a Dense Linear Algebra Library.

[7] J. Dongarra, J. Bunch, C. Moler and G. W. Stewart 1970 **LINPACK user's guide**. Philadelphia: *SIAM*.

[8] D. S. Jones 1989 *Q. J. Mech. Appl. Math.* **44**, 129-142 Integral equations for exterior acoustic problem.

[9] M. C. Junger and D. Feit 1972 **Sound, structures, and their interaction**. Cambridge, Massachusetts: The MIT Press.

[10] A. Karafiat, J.T. Oden and P. Geng 1993 *International Journal of Engineering Sciences* **31**, 649-672. Variational formulations and *hp*-boundary element approximation of hypersingular integral equations for Helmholtz-exterior boundary-value problems in two dimensions.

[11] H. R. Kutt 1975 *Report WISK 179.* On the numerical evaluation of finite part integrals involving an algebraic singularity. Pretoria: The National Research Institute for Mathematical Sciences.

[12] A. W. Maue 1949 *Z. Phys.* **126**, 601-618. Zur Formulierung Eines Allegm einen Beugungsproblems Durch Eine Integralgleichung.

[13] K. M. Mitzner 1952 *J. Math. Phys.* **7**, 2053-2060. Acoustic Scattering from an Interface between Media of Greatly Different Density.

[14] I. O. Panich 1965 *Russian Math. Surv.* **20**, 221-226. On the question of solvability of the exterior boundary value problems for the wave equation and Maxwell's equations.

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery 1992 **Numerical Recipes in Fortran**. New York: Cambridge University Press, second edition.

[16] A. H. Schenck 1967 *J. Acoust. Soc. Am.* **44**, 41-58. Improved integral formulation for acoustic radiation problems.

[17] F. Ursell 1973 *Math. Proc. Camb. Phil. Soc.* **74**, 117-125. On the exterior problems of acoustics.

[18] F. Ursell 1978 *Math. Proc. Camb. Phil. Soc.* **84**, 545-548. On the exterior problems of acoustics II.

[19] R. A. van de Geijn 1992 *Dept. Comp. Sci., U of Texas* **TR-91-28**. Massively parallel LINPACK benchmark on the Intel Touchstone DELTA and IPSC/860 systems.

Table 1: The number of processors, elements and degrees of freedom for 2-D BEM

| No. of Processors | No. of Elements | dof |
|---|---|---|
| P = 1 | 256 | 512 |
| P = 2 | 360 | 720 |
| P = 4 | 512 | 1,024 |
| P = 8 | 720 | 1,440 |
| P = 16 | 1,024 | 2,048 |
| P = 32 | 1,448 | 2,896 |

Table 2: Computation time on the iPSC/860 for each step of computation and the estimated parallel efficiencies with the logical mesh of processors as $1 \times P$ for 2-D BEM

| | Formation of linear Sys. | | Matrix Factorization | | For. and back Substitution | | Error estimation | |
|---|---|---|---|---|---|---|---|---|
| | time | $E_{eff}$ | time | $E_{eff}$ | time | $E_{eff}$ | time | $E_{eff}$ |
| P= 1 | 512 s | | 16.5 s | | 0.25 s | | 453 s | |
| P= 2 | 509 s | 100.5% | 24.7 s | 92.9% | 0.44 s | 56.7% | 447 s | 101.3% |
| P= 4 | 514 s | 99.6% | 55.3 s | 61.3% | 0.68 s | 36.7% | 451 s | 100.4% |
| P= 8 | 510 s | 100.4% | 102.3 s | 46.1% | 0.94 s | 26.6% | 445 s | 101.8% |
| P=16 | 519 s | 98.6% | 119.7 s | 56.9% | 1.63 s | 15.3% | 451 s | 100.4% |
| P=32 | 541 s | 94.5% | 224.4 s | 42.7% | 2.80 s | 9.0% | 466 s | 97.2% |

Table 3: Computation time on the iPSC/860 for each step of computation and the estimated parallel efficiencies with the logical mesh of processors as given in (5.10) for 2-D BEM.

| | Formation of linear Sys. | | Matrix Factorization | | For. and back Substitution | | Error estimation | |
|---|---|---|---|---|---|---|---|---|
| | time | $E_{eff}$ | time | $E_{eff}$ | time | $E_{eff}$ | time | $E_{eff}$ |
| P= 1 | 512 s | | 16.5 s | | 0.25 s | | 453 s | |
| P= 2 | 509 s | 100.5% | 24.7 s | 92.9% | 0.44 s | 56.7% | 447 s | 101.3% |
| P= 4 | 604 s | 84.8% | 35.6 s | 92.5% | 0.48 s | 52.4% | 451 s | 100.4% |
| P= 8 | 598 s | 85.6% | 57.8 s | 79.2% | 0.89 s | 28.2% | 445 s | 101.8% |
| P=16 | 653 s | 78.4% | 76.3 s | 86.2% | 1.12 s | 22.4% | 451 s | 100.4% |
| P=32 | 665 s | 77.0% | 132.5 s | 72.0% | 1.88 s | 13.3% | 466 s | 97.2% |

Table 4: Performance attained on the iPSC/860 by the dense solver for 2-D BEM. The matrix size, as well as the number of million bytes used on each processor to store the matrix are also listed.

| no. of processor | logical mesh | Mflops per processor | Mbytes per processor | dimension of matrix |
|---|---|---|---|---|
| P= 1 | $1 \times 1$ | 21.2 | 4.19 | 512 |
| P= 2 | $1 \times 2$ | 19.4 | 4.15 | 720 |
| P= 4 | $2 \times 2$ | 19.53 | 4.19 | 1,024 |
| P= 8 | $2 \times 4$ | 16.7 | 4.15 | 1,440 |
| P=16 | $4 \times 4$ | 18.2 | 4.19 | 2,048 |
| P=32 | $4 \times 8$ | 15.27 | 4.19 | 2,896 |

Table 5: Comparison of two parallel algorithms for the generation of the linear system for 3-D BEM.

| | Time elapsed | Parallel efficiency |
|---|---|---|
| Algorithm 1 | 6,125 (s) | 49 % |
| Algorithm 2 | 3,887 (s) | 78 % |
| I/O assembling | 464 (s) | |

Table 6: Computation time and estimated parallel efficiency for the generation of linear system for different numbers of processors and sizes of problem in the Touchstone Delta machine.

| no. of processor | size of problem | computational time (second) | estimated parallel efficiency |
|---|---|---|---|
| P= 1 | 514 | 3869.5 | |
| P= 32 | 2,050 | 1938.3 | 97% |
| P= 64 | 4,098 | 4216.5 | 91% |
| P= 128 | 2,050 | 594.1 | 81% |
| P= 128 | 4,098 | 2345.2 | 82% |
| P= 128 | 6,146 | 5224.2 | 82% |
| P= 512 | 8,194 | 3580.0 | 53% |

Table 7: Computation speed attained by the dense solver on the Touchstone Delta machine. The matrix size, the number of million bytes used on each processor to store the matrix, and the computation time are also listed.

| no. of processor | logical mesh | computational time (second) | Mflops per processor | Mbytes per processor | dimension of matrix |
|---|---|---|---|---|---|
| P=1 | $1 \times 1$ | 17 | 21.46 | 4.23 | 514 |
| P=16 | $4 \times 4$ | 68 | 21.01 | 4.20 | 2,050 |
| P=32 | $4 \times 8$ | 297 | 19.30 | 8.40 | 4,098 |
| P=64 | $8 \times 8$ | 407 | 17.43 | 4.20 | 4,098 |
| P=64 | $8 \times 8$ | 164 | 23.77 | 9.44 | 6,146 |
| P=128 | $8 \times 16$ | 130 | 11.04 | 2.10 | 4,098 |
| P=256 | $16 \times 16$ | 297 | 19.31 | 4.20 | 8,194 |

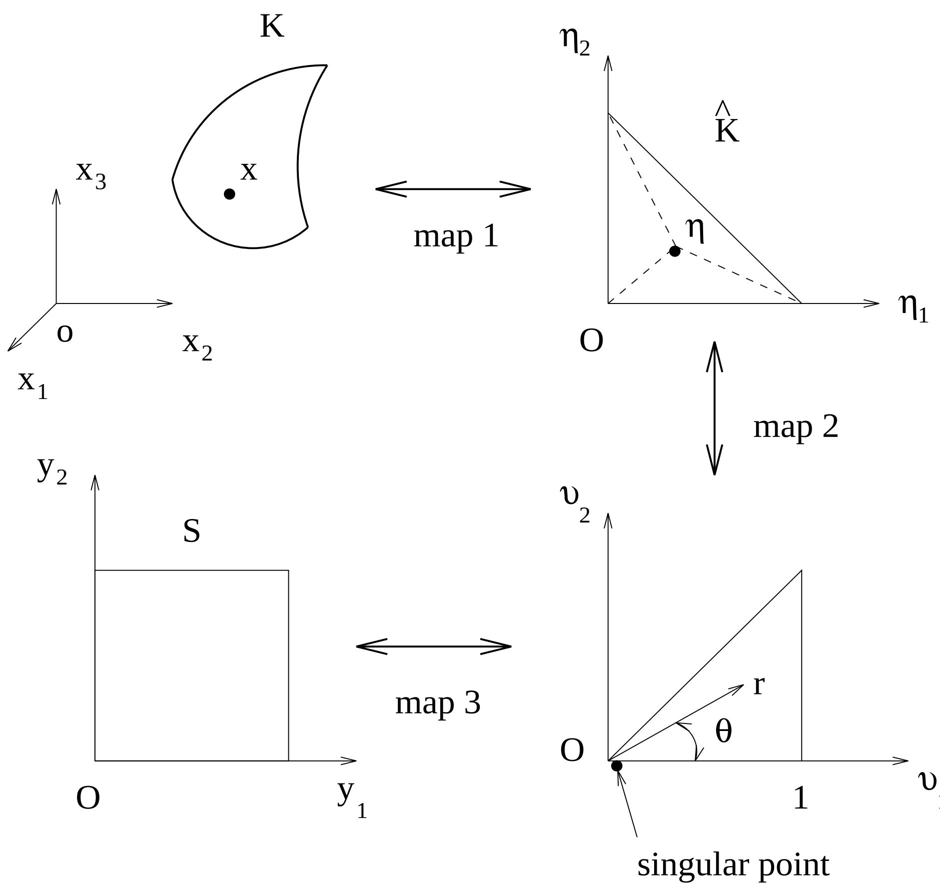


Figure 1: Evaluation of singular integrals: Duffy triangular coordinate method.

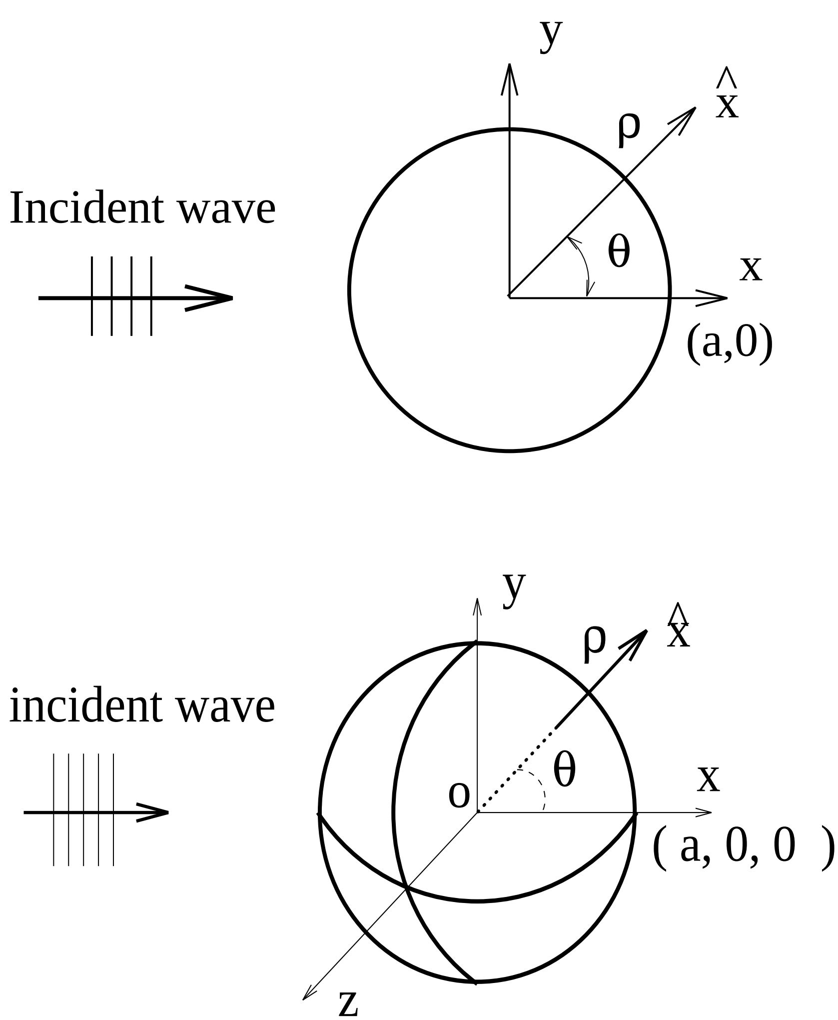


Figure 2: The test problems in two- and three-dimensions.

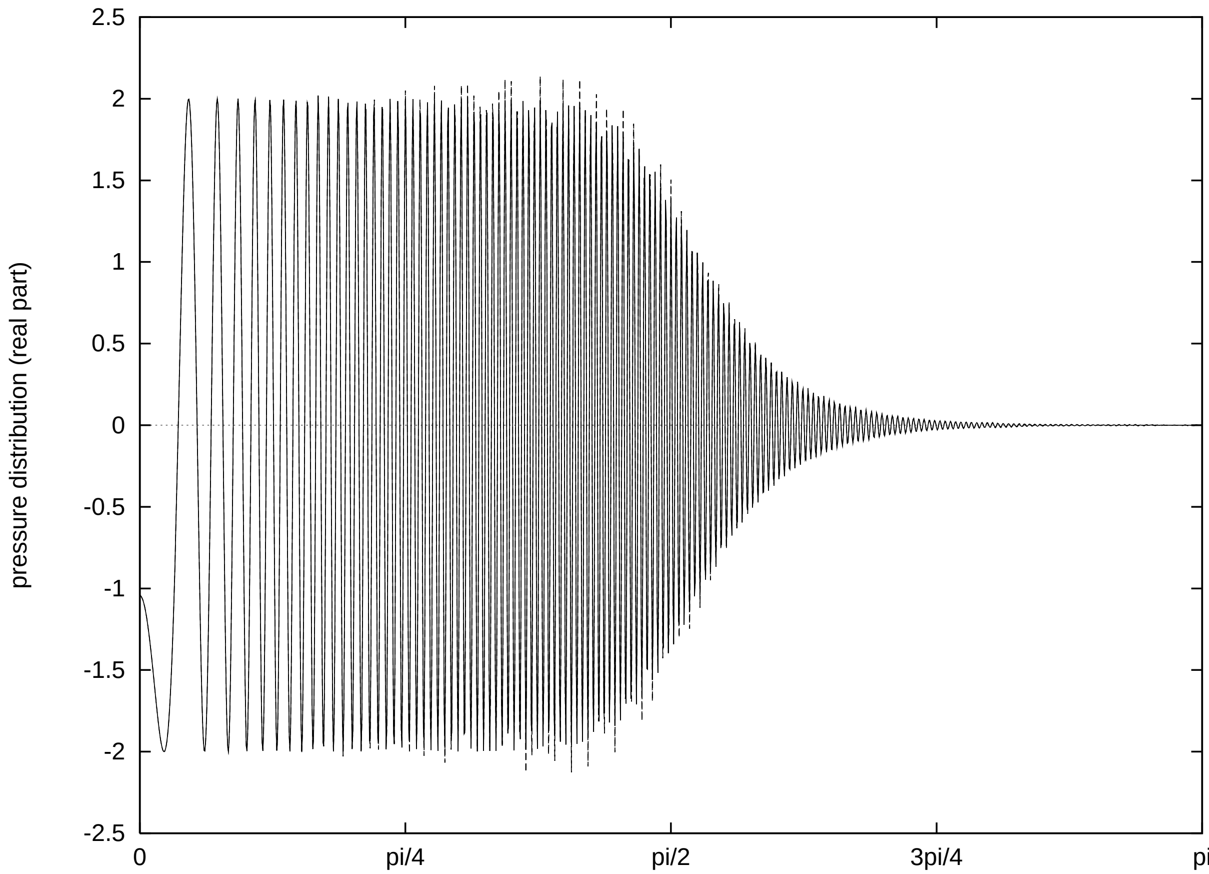


Figure 3: The comparison between numerical and exact solutions around the cylindrical boundary with $ka = 400$ and $\partial p/\partial n = p$. Because of symmetry, only the solution from $0^0$ to $180^0$ around the cylindrical boundary is plotted. The relative $L^2$ error and the corresponding estimated error are 2.28 % and 2.35 %, respectively. The problem is solved with a uniform mesh of 1,024 quadratic elements and 2,048 degrees of freedom.
——— analytic solution, - - - - numerical solution.

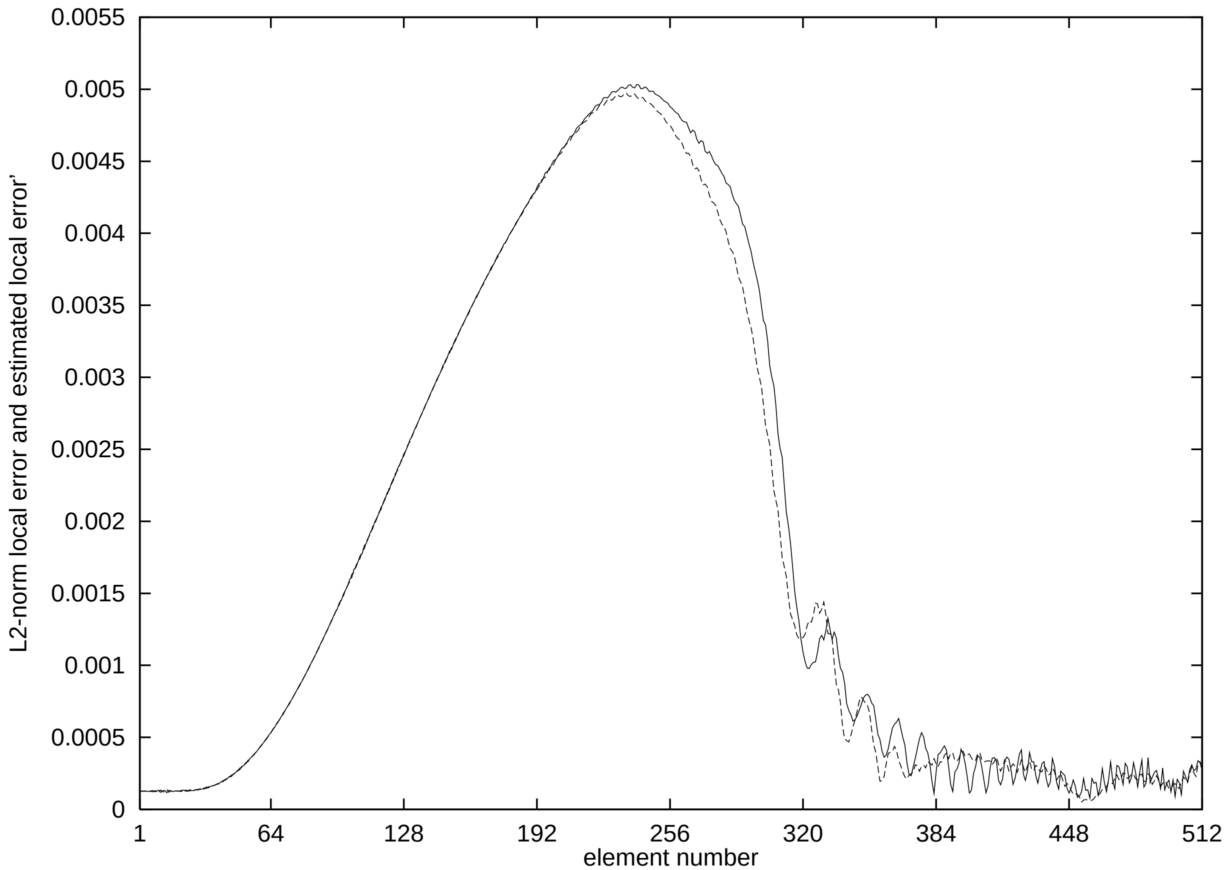


Figure 4: The comparison between the local error and the estimated local error. Because of symmetry, only results on the elements from 1 to 512 are presented.
—— estimated local error, - - - - true local error.

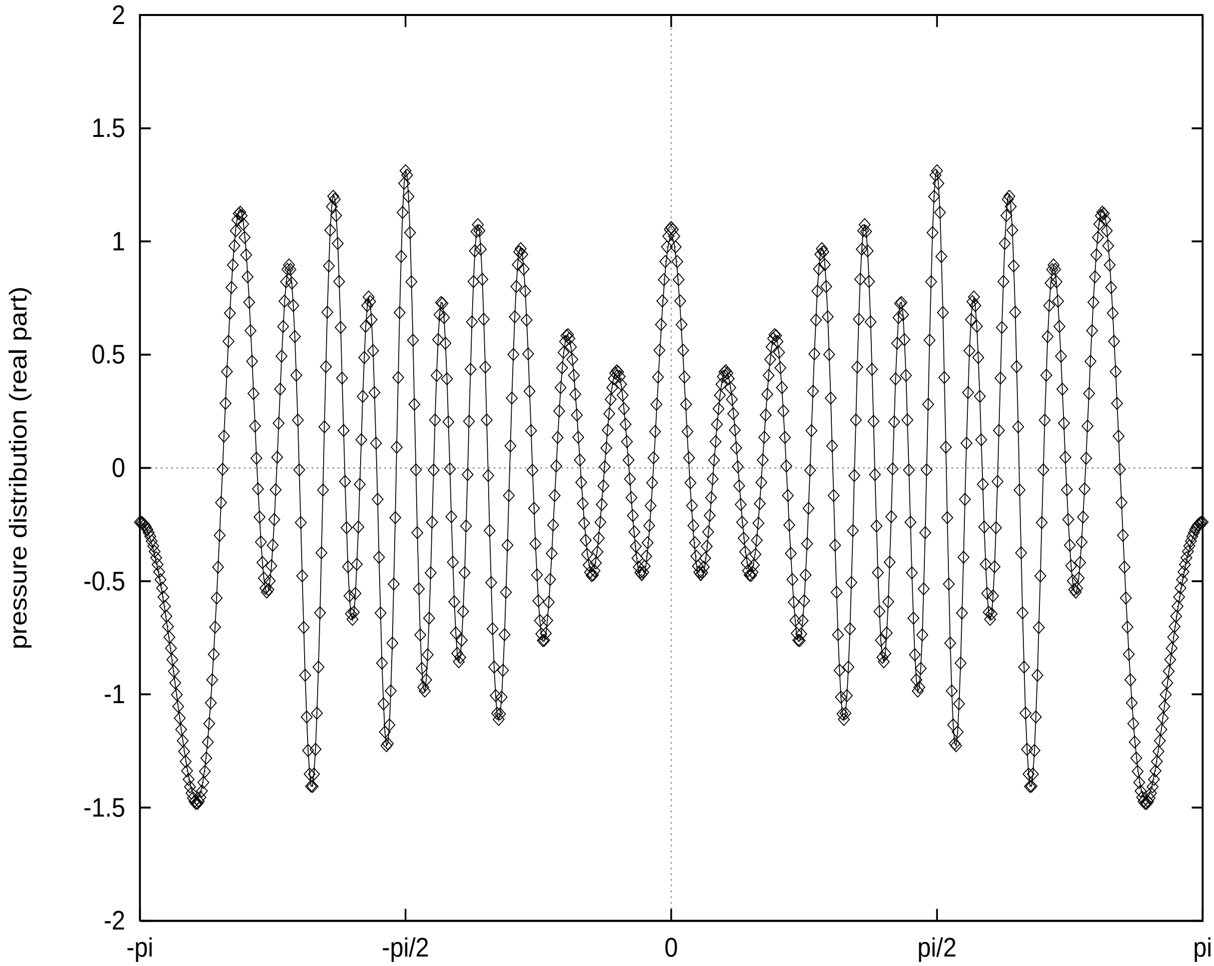


Figure 5: The comparison between numerical and analytic solutions around the curve of $\sqrt{x^2+y^2}=1.5$, $z=0$ for the test problem in three dimensions with $ka=20$ and $\partial p/\partial n=0$. The relative $L^2$ error and the corresponding estimated error are 4.9 % and 5.2 %, respectively. The problem is solved with the mesh of 1,568 quadratic elements and 3,138 degrees of freedom. ——— analytic solution, $\diamond$ numerical solution.

# Caption: Figures 1 to 5

Figure 1. Evaluation of singular integrals: Duffy triangular coordinate method.

Figure 2. The test problems in two- and three-dimensions.

Figure 3. The comparison between numerical and exact solutions around the cylindrical boundary with $ka = 400$ and $\partial p/\partial n = p$. Because of symmetry, only the solution from $0^0$ to $180^0$ around the cylindrical boundary is plotted. The relative $L^2$ error and the corresponding estimated error are 2.28 % and 2.35 %, respectively. The problem is solved with a uniform mesh of 1,024 quadratic elements and 2,048 degrees of freedom.
——— analytic solution, - - - - numerical solution.

Figure 4. The comparison between the local error and the estimated local error. Because of symmetry, only results on the elements from 1 to 512 are presented.
——— estimated local error, - - - - true local error.

Figure 5. The comparison between numerical and analytic solutions around the curve of $\sqrt{x^2 + y^2} = 1.5$, $z = 0$ for the test problem in three dimensions with $ka = 20$ and $\partial p/\partial n = 0$. The relative $L^2$ error and the corresponding estimated error are 4.9 % and 5.2 %, respectively. The problem is solved with the mesh of 1,568 quadratic elements and 3,138 degrees of freedom.
——— analytic solution, $\diamond$ numerical solution.