JOURNAL OF COMPUTATIONAL PHYSICS **145,** 731–742 (1998)
ARTICLE NO. CP986048

Rapid Parallel Evaluation of Integrals in Potential Theory on General Three-Dimensional Regions

A. Greenbaum*,[1] and A. Mayo†

**Department of Mathematics, University of Washington, Box 354350, Seattle, Washington 98195, and †IBM T. J. Watson Research Center, Yorktown Heights, New York 10598*

Received June 10, 1997; revised June 23, 1998

We present a new, high order accurate method for the rapid, parallel evaluation of certain integrals in potential theory on general three-dimensional regions. These methods use fast methods for solving the differential equation which the kernel satisfies, and the number of operations needed to evaluate the integrals is essentially equal to the number of operations needed to solve the differential equation on a regular rectangular grid. In particular, one can evaluate integrals whose kernels are the Greens function for Poissons equation by using Fourier methods on a rectangular grid, or, a fast Poisson solver. Thus, these methods avoid the problems associated with using quadrature methods to evaluate an integral with a singular kernel. Numerical results are presented for experiments on a variety of geometries. © 1998 Academic Press

1. INTRODUCTION

New, fast numerical techniques for generating and solving the matrix equations that arise in integral equation methods [5, 6, 12, 15, 16] has led to their increased use in industrial applications. Despite the reduction in cost, however, the methods may still require expensive operations. This is because in order to completely solve a problem it is frequently necessary to evaluate a surface integral at many points of a region. Moreover, integral equation formulations for inhomogeneous differential equations usually require the evaluation of a volume integral.

The evaluation of both the volume and surface integrals is very expensive. In particular, we note that $O(n^6)$ operations are needed in order to evaluate a volume integral at every point of a three-dimensional $n \times n \times n$ grid, since the evaluation of each integral requires $O(n^3)$ operations.

[1] Corresponding author. E-mail: greenbau@math.washington.edu. *This work was supported by the Applied Mathematical Sciences Program of the U.S. Department of Energy under Contract DE-AC02-76ER03077.

0021-9991/98 $25.00
Copyright © 1998 by Academic Press
All rights of reproduction in any form reserved.

Aside from the cost of evaluating the integrals, an additional difficulty with integral equation methods is the problem of accurately evaluating the solution at points inside and near the boundary. In integral equation methods for solving elliptic differential equations, the solution is expressed as the integral over the boundary of the product of a density function and a singular kernel. The singularity of the kernel makes it difficult to compute the solution accurately by quadrature at points near the region of integration.

In this paper we develop fast, parallel three-dimensional methods for evaluating volume and surface integrals in potential theory which overcome these two problems. These methods are mesh-based and are generalizations of our previous work. (See [7–11].) This new work has required a number of new numerical techniques. In particular, it has required the derivation of several new formulas which are not obvious generalizations of the two-dimensional formulas, especially when the surface is curved. Specifically, we have had to use the Frenet formulas in order to obtain second-order accurate approximations to the integrals. We have also used nested triangulations in order to compute volume integrals and integrals of double layer density functions. The new numerical algorithms have been implemented on a distributed memory parallel computer, the IBM SP2.

The methods used to evaluate the integrals rely on fast finite difference methods. In these methods the irregular region on which it is desired to evaluate an integral U is embedded in a larger rectangular region on which there is a regular grid. If, for example, the kernel of U satisfies Laplace's equation, we compute an approximation to the integral by first computing an approximation to its discrete Laplacian at all points of the regular grid. Then we apply an operator that inverts the discrete Laplacian Δ_h to obtain an approximation to U. To compute an approximation to $\Delta_h U$ at mesh points which have all their neighboring mesh points on the same side of the boundary of the irregular region we use the fact that, up to truncation error, the discrete Laplacian is equal to the continuous Laplacian ΔU which is known. However, because the integrals we compute necessarily have discontinuities in some of their derivatives at the edge of the domain of integration, we must compute special correction terms at certain mesh points near the boundary. It turns out that these correction terms can be computed in terms of the discontinuities in the derivatives of the integral and how far the mesh points are from the boundary of the region of integration. We show how to compute these discontinuities and how to use them to compute approximations to $\Delta_h U$.

We note that these three-dimensional volume integrals also arise in other contexts, in addition to the solution of inhomogeneous differential equations. In particular, they are needed when applying the Biot Savart law to evaluate the magnetic field induced by a conducting wire. This law says that the field B, induced by a conducting wire, is the curl of the volume integral of the fundamental solution of Poisson's equation multiplied by the current density J,

$$B(x, y, z) = \nabla \times \iiint_D J \cdot \frac{1}{4\pi r}\, dx'\, dy'\, dz', \tag{1.1}$$

where $r(x, y, z, x', y', z')$ is the distance between the source and the evaluation points. Therefore, each of the three components of the field is equal to the sum of integrals whose kernels are comprised of products of derivatives of the fundamental solution with components of the current density.

In addition to being fast in the serial mode, these methods are easily parallelized. We have, in fact, evaluated the volume and surface integrals on the IBM SP2 parallel computer.

The organization of this paper is as follows. In Section 2 we show how to compute a second-order accurate approximation to volume integrals whose kernels are a fundamental solution of Poisson's equation, and in Section 3 we discuss certain extensions of the method to the evaluation of other integrals. In particular, we show how to evaluate surface integrals of single and double layer density functions and the derivatives of these integrals. In Section 4 we provide results of numerical experiments, including some on the IBM SP2 distributed memory parallel computer.

2. EVALUATION OF VOLUME INTEGRALS

As an example of how the method works, we first show how to compute a second-order accurate approximation to a volume integral whose kernel satisfies Laplace's equation, that is, an integral of the form

$$U(x, y, z) = \frac{1}{4\pi} \iiint_D \frac{1}{r(x, y, z, x', y', z')} f(x', y', z')\, dx'\, dy'\, dz', \tag{2.1}$$

where $r(x, y, z, x', y', z') = \sqrt{(x - x')^2 + (y - y')^2 + (z - z')^2}$. We embed the region of integration D in a larger rectangular region R with mesh $\{(ih_x, jh_y, kh_z) : i = 0, \ldots, n_x, j = 0, \ldots, n_y, k = 0, \ldots, n_z\}$. The integral U is defined at all mesh points of R.

Let $(\Delta_h U)_{i,j,k}$ denote the usual seven-point approximation to the Laplacian:

$$\begin{aligned} &\frac{U(i+1, j, k) + U(i-1, j, k) - 2U(i, j, k)}{h_x^2} \\ &\quad + \frac{U(i, j+1, k) + U(i, j-1, k) - 2U(i, j, k)}{h_y^2} \\ &\quad + \frac{U(i, j, k+1) + U(i, j, k-1) - 2U(i, j, k)}{h_z^2}. \end{aligned}$$

To obtain our approximation U_h to the integral we compute the approximation to its discrete Laplacian at all points of R, and then apply a Poisson solver. This gives an approximation to $U(x, y, z)$ throughout R.

To compute the discrete Laplacian we use the fact that

$$\Delta U = f \quad \text{in } D, \quad \Delta U = 0 \quad \text{outside } D. \tag{2.2}$$

Therefore at mesh points of R inside D that have all their neighbors in D, we set $(\Delta_h U)_{ijk} = f_{ijk}$, and at points outside D whose neighbors are also outside D we set $(\Delta_h U)_{ijk} = 0$.

Since U is not a smooth function, we cannot use either formula at points near ∂D; that is, at mesh points inside (outside) D with one or more neighbors outside (inside) D. See Fig. 1 for a 2D illustration. (It is easy to see that U is not smooth at the boundary of D because the Laplacian of U is discontinuous there.) Therefore, at such points the values of the discrete Laplacian are not well approximated by the values of the exact Laplacian.

To compute an approximation to the discrete Laplacian of U at these irregular mesh points, we will determine the discontinuities in U and its derivatives and then use these as correction terms in a Taylor series expansion for the Laplacian.

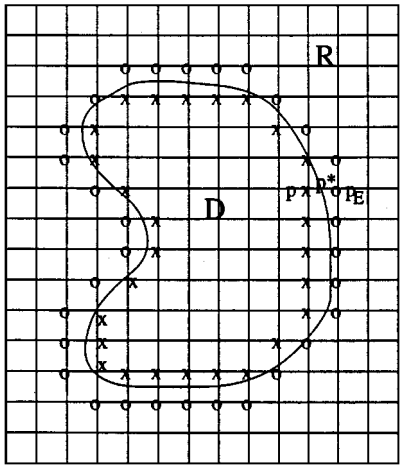

FIG. 1. Irregular mesh points: x's and o's.

For a given function g defined on R which is discontinuous on ∂D let $[g(p^*)]$ denote the discontinuity in g at a point p^* on ∂D. An integral of the form (2.1) and its normal derivative U_n are continuous across ∂D, i.e.

$$\begin{aligned} [U] &= 0, \\ [U_n] &= 0. \end{aligned} \tag{2.3}$$

Therefore, the components of the gradient are also continuous:

$$[U_x] = [U_y] = [U_z] = 0. \tag{2.4}$$

By (2.2) there must be discontinuities in the second derivatives of U.

To determine discontinuities in the six second derivatives of U we differentiate (2.1) in the normal direction and any two tangential directions to the surface, and we use the discontinuity in the Laplacian. That is, we use the six equations,

$$\begin{gathered} [U_{ss}] = 0, \quad [U_{st}] = 0, \quad [U_{tt}] = 0, \\ [U_{ns}] = 0, \quad [U_{nt}] = 0, \quad [U_{xx}] + [U_{yy}] + [U_{zz}] = f. \end{gathered} \tag{2.5}$$

where s and t are directions in the tangent plane.

To be more specific, suppose that the surface ∂D has been decomposed into curved triangular patches, with each patch being the image of a triangle T in the s–t plane. A patch corresponding to triangle T can be represented as the set of points $(x(s,t), y(s,t), z(s,t))$, where $(s,t) \in T$, and any point q inside the patch corresponds to a point $(s_q, t_q) \in T$.

For a fixed value t_q, the function

$$a(s) = (x(s,t_q), y(s,t_q), z(s,t_q)), \quad (s,t_q) \in T,$$

is a curve on the surface, and

$$a_1(s) \equiv a_1(s,t_q) = \left(\frac{\partial x}{\partial s}(s,t_q), \frac{\partial y}{\partial s}(s,t_q), \frac{\partial z}{\partial s}(s,t_q) \right)$$

is a tangent vector in the s direction. Similarly, for a fixed value s_q, the curve $(x(s_q, t), y(s_q, t), z(s_q, t))$, $(s_q, t) \in T$, has tangent vector in the t direction given by

$$a_2(t) \equiv a_2(s_q, t) = \left(\frac{\partial x}{\partial t}(s_q, t), \frac{\partial y}{\partial t}(s_q, t), \frac{\partial z}{\partial t}(s_q, t) \right).$$

Let

$$a_1(s_q) = (a_{11}, a_{12}, a_{13})$$

and

$$a_2(t_q) = (a_{21}, a_{22}, a_{23})$$

be the tangent vectors at the point q. Then

$$U_s(q) = a_1 \cdot \nabla U = a_{11} U_x(q) + a_{12} U_y(q) + a_{13} U_z(q),$$

and so

$$U_{ss} = a_{11}^2 U_{xx} + 2a_{11}a_{12}U_{xy} + a_{12}^2 U_{yy} + 2a_{11}a_{13}U_{xz} + 2a_{12}a_{13}U_{yz} + a_{13}^2 U_{zz} + \frac{da_1}{ds} \cdot \nabla U.$$

Similarly,

$$\begin{aligned} U_{ts} = {} & a_{11}a_{21}U_{xx} + (a_{11}a_{22} + a_{12}a_{21})U_{xy} + a_{12}a_{22}U_{yy} + (a_{11}a_{23} + a_{13}a_{21})U_{xz} \\ & + (a_{12}a_{23} + a_{13}a_{22})U_{yz} + a_{13}a_{23}U_{zz} + \frac{da_2}{ds} \cdot \nabla U. \end{aligned}$$

(By da_2/ds we mean the derivative of the tangent vector in the t direction, as we move along the surface in the s direction.) It follows that in order to use (2.5) to determine discontinuities in derivatives of U at a point q on the surface we need to determine quantities such as da_1/ds and da_2/ds.

These can be determined using the Frenet formulas [14]:

$$\frac{dT}{ds} = \kappa N \tag{2.6a}$$

$$\frac{dN}{ds} = -\kappa T + \tau B \tag{2.6b}$$

$$\frac{dB}{ds} = -\tau N, \tag{2.6c}$$

where T is the tangent vector along a curve, N is the normal vector, B is the binormal, κ is the cuvature, and τ is the torsion.

Using (2.6a) on the curve $a(s)$ with tangent $a_1(s)$ we see that

$$\frac{da_1}{ds} = \kappa N.$$

To determine da_2/ds we note that, since for fixed (s_q, t_q) the vectors a_1 and a_2 are tangent vectors at point q and $B(s)$ also lies in the tangent plane (N is orthogonal to it), we have

$$a_2 = (a_2, a_1)a_1 + (a_2, B)B.$$

This equation and (2.6c) imply that

$$\frac{da_2}{ds} = (a_2, a_1)\kappa N - (a_2, B)\tau N.$$

Thus, the quantities on the right-hand side of (2.5) can be determined, and the system is solved to find the discontinuities in the six second derivatives of U at a point q.

By differentiating the above equations and using higher derivatives of the boundary surface it is possible to derive formulas for discontinuities in higher order derivatives.

We now show how to use these discontinuities to determine the discrete Laplacian of U at the irregular mesh points. The integral U is defined at points both inside and outside D. We again let $[U(p^*)]$ denote the jump in U at a point p^* on ∂D.

To see how to compute the approximation to $\Delta_h U$ at irregular points, suppose, for example, that a point p is in D, but its neighbor to the right, p_E, is not. Let p^* be the point on the line between p and p_E which intersects ∂D, let h_1 be the distance between p and p^*, and let $h_2 = h - h_1$.

By manipulating the Taylor series at p and p_E both evaluated at p^*, we can derive the following expression for $U(p_E) - U(p)$ (for details see [9]):

$$U(p_E) - U(p) = [U(p^*)] + h_2[U_x(p^*)] + \frac{1}{2}h_2^2[U_{xx}(p^*)] + \frac{1}{6}h_2^3[U_{xxx}(p^*)] + hU_x(p) + \frac{1}{2}h^2U_{xx}(p) + \frac{1}{6}h^3U_{xxx}(p) + O(h^4). \tag{2.7}$$

Note that the first four terms depend on the discontinuities in U and in its x derivatives at the boundary. The other terms are the usual Taylor series terms. Therefore, the right-hand side of (2.7) is a sum of terms that can be evaluated in terms of the discontinuities in U and its derivatives and terms that would be present even if there were no boundary between p and p_E.

Now let p_W, p_N, p_S, p_F, and p_B be the mesh points to the left of, above, below, in front of, and behind p. We get the same type of expressions for the differences between the value of U at p and at its other neighbors, that is $U(p_W) - U(p)$, $U(p_N) - U(p)$, $U(p_S) - U(p)$, $U(p_F) - U(p)$, and $U(p_B) - U(p)$, except that there will not be any boundary terms unless ∂D passes between p and that neighbor. Therefore, we can compute an approximation to the seven-point discrete Laplacian of U, which is just the sum of the above six differences. This is done at all the irregular points.

If B denotes the set of irregular mesh points, then for mesh points $(x_i, y_j, z_k) \in B$ define the mesh function m_{ijk} to be the value of the extra terms in the discrete Laplacian obtained by this procedure using f and its derivatives. We take U_h to be the solution of the equations:

$$\Delta_h U_h = \begin{cases} f_{ijk}, & (i, j, k) \in D \backslash B, \\ f_{ijk} + m_{ijk}, & (i, j, k) \in B \cap D, \\ m_{ijk}, & (i, j, k) \in B \cap (R \backslash D), \\ 0, & (i, j, k) \in (R \backslash D) \backslash B. \end{cases}$$

If the values of $m_{i,j,k}$ are third-order accurate, then by applying a second-order accurate Poisson solver we obtain a second-order accurate approximation U_h to U (see [10, 11]).

We can clearly also compute higher order accurate solutions by using higher order accurate approximations to the Laplacian.

Before applying a Poisson solver to obtain a solution we must supply appropriate boundary conditions on the boundary of the embedding cube. In particular, if we want to evaluate the integral (2.1) corresponding to "free space" boundary conditions we need to use a Poisson solver that provides the proper $1/r$ decay at infinity. It turns out in this case that we can use a method originally developed by Hockney [3] and later improved by James [4], where one calculates the boundary potential by finding a set of correction charges on the boundary of the embedding region, and then convolves them with a suitable Green's function. This method, however, requires the application of two Poisson solvers.

In some applications only a particular solution of Poisson's equation is needed. In that case we can use any Poisson solver. The integral that the method gives an approximation to is the one associated with the same boundary conditions as the fast Poisson solver we use. For example, if we use a triply periodic Poisson solver, then we obtain an approximation to the integral whose kernel is the periodic Green's function for the Laplacian, instead of the free space Green's function $1/r$. If we need an integral with a specific kernel, say one that is periodic in only one direction, then we use the corresponding Poisson solver which is periodic in that direction. (We note that the discontinuities in the derivatives of the integral, and therefore, the discrete Laplacian, will be the same, independent of which fundamental solution of Laplace's equation is used as the kernel.)

3. EXTENSIONS

In this section we present extensions of the method of the previous section to the evaluation of other integrals. Specifically, we show how to evaluate certain surface integrals in potential theory, and we also show how to evaluate derivatives of the surface and volume integrals.

In order to evaluate the surface integrals we use the same basic method that is used to evaluate the volume integrals. In particular, this method can be used to evaluate integrals of single layer density functions:

$$W = 1/4\pi \int\!\!\int_{\partial D} \rho(s,t)(1/r)\, dS.$$

As for volume integrals, the problem of evaluating this type of integral reduces to evaluating its Laplacian in the two regions bounded by the boundary surface and evaluating the discontinuities in its derivatives at the boundary. We first note that the Laplacian vanishes in the two regions, i.e.

$$\Delta W = \begin{cases} 0, & \text{inside } D, \\ 0, & \text{outside } D. \end{cases}$$

It is also known [13] that such an integral is continuous in the tangential direction and has a discontinuity equal to the density in the normal direction:

$$W_n^i - W_n^e = \rho(s,t)$$
$$W^i - W^e = 0.$$

The above two sets of equations and their derivatives determine the discontinuities in the derivatives of W in the coordinate directions. Once determined, these discontinuities are used to determine the discrete Laplacian of W and the function W itself.

Similarly, we can evaluate integrals of double layer density functions:

$$W = 1/4\pi \iint_{\partial D} \mu(s,t) \frac{\partial}{\partial n_s}(1/r)\, dS.$$

Just as in the case of an integral of a single layer density function, this type of integral is harmonic in the region D and in the region outside. The only difference is that it is continuous in the normal direction and has a discontinuity equal to the density in the tangential direction:

$$W_n^i - W_n^e = 0,$$
$$W^i - W^e = \mu(s,t).$$

Once again these equations determine the discontinuities in the derivatives of these integrals in the coordinate directions, and these are used to determine the discrete Laplacian at the irregular mesh points.

We now show how to evaluate derivatives, or a linear combination of the derivatives of a surface or volume integral.

To evaluate discontinuities in derivatives of integrals with differentiated kernels we use the fact that those integrals are derivatives of integrals with undifferentiated kernels. For example, suppose

$$V(p) = \iiint_D f(p') G_x(p, p')\, dV',$$

where $G(p, p')$ is the fundamental solution $1/r$ or its normal derivative. Then $V(p) = U_x(p)$, where

$$U(p) = \iiint_D f(p') G(p, p')\, dV'.$$

Since we know how to compute the discontinuities in the derivatives of U, we can, of course, compute the discontinuities in the derivatives of $V = U_x$. As before, once the discontinuity terms are known they can be used to approximate the discrete Laplacian of V.

4. NUMERICAL EXPERIMENTS

We first tested this method on problems for which one can evaluate the integrals analytically, both inside and outside the region of integration. The test regions D we used were the unit cube, a simplified recording head geometry, and the unit sphere. The surfaces ∂D of all the regions were triangulated, and the computations were performed in double precision (16 digits) on an IBM 3090 computer. The running time was essentially equal to the time needed to apply a three-dimensional Poisson solver. (We used POIS3D from FISHPAK. [17]) Although we have implemented fourth-order accurate versions of this method in two dimensions [7, 10], so far we have implemented only a second-order accurate method in three dimensions.

TABLE I
Embedded Cube, Maximum Relative Errors

n	Surface nodes	Max error for $(x-3)^2-y^2$	Max error for $y((x+4)^2-z^2)+3$
16	26	0.009	0.02
32	386	0.005	0.007
64	1538	0.0009	0.0005

The methods were tested by using Greens theorem to evaluate harmonic functions. That is, inside a region D we represented a harmonic function U as the sum of a single and a double layer density function:

$$U = 1/4\pi \int\int_{\partial D} \left(U \frac{\partial}{\partial n_s} \frac{1}{r} - U_n \frac{1}{r} \right) dS.$$

We prescribed the exact values of the function U and its normal derivative at the surface nodes (i.e. at the vertices of the triangles) and used linear interpolation to obtain the values of these functions and their derivatives at other points inside the triangles.

In the first test examples we let $U = x + y + z$ on all three regions. As expected, we obtained essentially machine accuracy (14 digits) in computing this function since it is linear. Although the Poisson solver is second-order accurate, one does not obtain machine accuracy in computing polynomials of degree 2, since linear interpolation of the second-degree polynomial values from the vertices to other points in the triangles is not exact. (When we provided the exact values of U at all points inside the triangle we again obtained essentially machine accuracy.)

In Tables I and II we present results of other test problems. In these examples the embedding region was the cube $[-0.23, 1.3]^3$. The numbers in column 1 are the number of mesh points in each direction in the embedding cube, and the numbers in the second column are the number of surface nodes.

The numbers in column 3 of Tables I and II are the maximum relative errors in computing the function $(x-3)^2 - y^2$ on the cube, and the numbers in column 4 are the maximum relative errors obtained in computing the function $y((x+4)^2 - z^2) + 3$ on the cube. As the surface mesh width is halved (the number of surface triangles grows by a factor of 4) and the mesh width in each direction of the embedding cube is halved (n grows by a factor of 2), we expect about a factor of 4 reduction in the error. A somewhat better reduction is seen in going from $n = 32$ to $n = 64$, perhaps because we have not yet reached the asymptotic regime.

TABLE II
Recording Head, Maximum Relative Errors

n	Surface nodes	Max error for $(x-3)^2-y^2$	Max error for $y((x+4)^2-z^2)+3$
16	96	0.09	0.08
32	361	0.03	0.008
64	1438	0.002	0.001

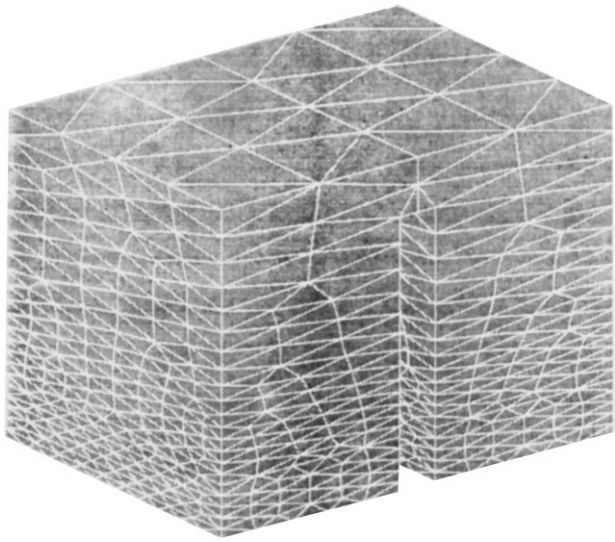

FIG. 2. U-shaped recording head.

The numbers in Table II are for computations performed on the recording head geometry shown in Fig. 2. This region had equal pole tips of width 0.1, gap 0.1, height 0.8, and uniform depth 0.8.

In the next set of experiments the region of integration D was the unit sphere, and the embedding region was the cube $[-2.3, 2.3]$. We computed the x component of the magnetic field of a unit sphere uniformly magnetized in the x direction. We assumed unit magnetization $M = 1$. This function, which can be represented as the x derivative of a volume integral with density $f(x, y, z) = 1$, is equal to 1 inside the sphere, and equal to $-3x^2/r^5 + 1/r^3$ outside the sphere. The maximum error we obtained in computing this function when we placed 8190 triangles on the surface was 0.006, and when there were 2046 surface triangles the maximum error was 0.04.

The final set of experiments involved parallelization of the code. For large 3D problems parallelization is a necessity because of storage requirements. A $1000 \times 1000 \times 1000$ embedding cube contains one billion mesh points and these cannot be stored on a single processor. Although the fast Poisson solver requires only $O(n^3 \log n)$ operations for an $n \times n \times n$ grid, time can also be an issue for large n. Our goal in parallelizing is to be able to solve a problem on a $2n \times 2n \times 2n$ grid using $8p$ processors in about the same amount of time that it takes to solve an $n \times n \times n$ problem using p processors. For this purpose we used the IBM SP2, a distributed memory parallel machine with up to 512 processors.

The part of the code that requires the most time is the Poisson solver. The first step in generating the right-hand side for the Poisson solver is to identify irregular mesh points—those that are inside the region D but have one or more neighbors outside, or vice versa. This is accomplished by looping through the triangles that make up the boundary of the region and determining which lattice lines intersect each triangle. Each processor can handle its share of the boundary triangles, but a merge of the data is required at the end to check for duplicates, since if a lattice line intersects the boundary at a vertex or edge of a triangle, it might be recorded more than once. Additionally, if a lattice point lies near the surface of the region then it might be reported as outside the region and paired with its neighbor

TABLE III
SP-2 Timings (Fast Poisson Solve Only)

Mesh points	Processors	Time (s)
128^3	1	9.4
256^3	8	13.0
512^3	64	16.7

inside, or it might be reported as inside the region and paired with its neighbor outside. If the intersection point is the same, the lattice pairs should be considered the same. It makes little difference whether the point is considered to be just inside or just outside the domain, but it should be recorded only once. Currently, each processor retains the entire list of irregular mesh points.

After determining the irregular mesh points, each processor is assigned certain z-planes in the embedding cube. The processor initializes the right-hand side elements corresponding to its planes to 0 and then loops through the irregular mesh points to determine which ones affect its elements of the right-hand side vector. For the appropriate irregular mesh points, each processor uses information about the boundary intersection point and discontinuties in the integral to determine the right-hand side element corresponding to that point.

Now each processor contains its piece of the right-hand side vector and a parallel 3D fast Poisson solver can be applied. The fast Poisson solver uses sine transforms in the x, y, and z directions. These are carried out by each processor using ESSL routine SINFT [2]. First the transforms in the x and y directions are computed, and then the data is transposed, so that each processor contains only certain x-planes. Then each processor can compute the sine transforms in the z direction. After this, results are multiplied by appropriate factors, and the inverse sine transforms in the z and y directions are computed. The data is transposed again, so that each processor again contains data for certain z-planes, and inverse transforms in the x direction are computed. This requires two global communication steps. Fortunately, for the problems we have run, this global communication has been a small part of the overall computation time for large problems. Table III shows timings for the fast Poisson solver only, when the amount of data per processor remains fixed. As can be seen, the goal of solving larger problems using more processors, without greatly increasing computation time has been fairly well achieved. In the future we plan to run a problem with a 1024^3 mesh on 512 processors.

The operation count for the fast Poisson solver is approximately $15n^3 \log_2 n + 92n^3$ for an $n \times n \times n$ grid. The 128^3 problem required approximately 4.13×10^8 double precision floating point operations, giving a single processor execution rate of about 44 mflops. For the 256^3 problem, run on eight processors, each processor executed at about 34 mflops; for the 512^3 problem on 64 processors, this execution rate was about 29 mflops per processor. Aside from the ESSL library references, the code is written in standard Fortran and has not been optimized. These execution rates could probably be improved with additional coding effort.

REFERENCES

1. O. Buneman, *A Compact Noniterative Poisson Solver*, Rep. SUIPR-294 (Inst. Plasma Research, Stanford University, Stanford CA, 1969).
2. *Engineering and Scientific Subroutine Library, Version 2, Release 2: Guide and Reference*, IBM SC23-0526.

3. R. Hockney, *Methods in Computational Physics, Vol. 9* (Academic Press, New York, 1970), p. 135.
4. R. James, The solution of Poisson's equation for Isolated source distributions, *J. Comput. Phys.* **25**, 71 (1977).
5. R. Fletcher, Conjugate gradient methods for indefinite systems, in *Numerical Analysis, Dundee 1975*, edited by G. A. Watson (Springer-Verlag, New York/Berlin, 1976), p. 73.
6. R. Freund and N. Nachtigal, QMR: A quasi-minimal residual method for non-Hermitian linear systems, *Numer. Math.* **60**, 315 (1991).
7. A. Mayo, A decomposition finite difference method for the fourth-order accurate solution of Poisson's equation on general regions, *Int. J. High Speed Comput.* **3**(2), 89 (1991).
8. A. Mayo and A. Greenbaum, Fast parallel iterative solution of Poisson's and the biharmonic equations on irregular regions, *SIAM J. Sci. Stat. Comput.* **13**(1), (1992).
9. A. Mayo, The fast solution of Poisson's and the biharmonic equations on general regions, *SIAM J. Numer. Anal.* **21**(2), 285 (1984).
10. A. Mayo, Rapid, high order accurate evaluation of volume integrals in potential theory, *J. Comput. Phys.* **100**(2), 236 (1992).
11. A. McKenney, L. Greengard, and A. Mayo, A fast Poisson solver for complex geometries, *J. Comput. Phys.* **118**, 348 (1995).
12. M. Meijerink and H. van der Vorst, An iterative solution method for linear systems, *Math. Comput.* **31**, 148 (1977).
13. S. Mikhlin, *Integral Equations and their Applications* (Pergamon, New York, 1957).
14. B. O'Neill, *Elementary Differential Geometry* (Academic Press, London, 1966).
15. V. Rokhlin, Rapid solution of integral equations of classical potential theory, *J. Comput. Phys.* **60**(2), 187 (1985).
16. Y. Saad and M. Schultz, GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.* **7**, 856 (1986).
17. P. Swarztrauber, Fast Poisson solvers, in *MAA Studies in Numerical Analysis, Vol. 24*, edited by G. H. Golub (Math. Assoc. Am., Washington, DC, 1984), p. 319.