SIAM J. SCI. COMPUT.
Vol. 15, No. 3, pp. 713–735, May 1994

014

PRECONDITIONED, ADAPTIVE, MULTIPOLE-ACCELERATED ITERATIVE METHODS FOR THREE-DIMENSIONAL FIRST-KIND INTEGRAL EQUATIONS OF POTENTIAL THEORY*

K. NABORS†, F. T. KORSMEYER‡, F. T. LEIGHTON§, AND J. WHITE†

Abstract. This paper presents a preconditioned, Krylov-subspace iterative algorithm, where a modified multipole algorithm with a novel adaptation scheme is used to compute the iterates for solving dense matrix problems generated by Galerkin or collocation schemes applied to three-dimensional, first-kind, integral equations that arise in potential theory. A proof is given that this adaptive algorithm reduces both matrix-vector product computation time and storage to order N, and experimental evidence is given to demonstrate that the combined *preconditioned, adaptive, multipole-accelerated* (PAMA) method is nearly order N in practice. Examples from engineering applications are given to demonstrate that the accelerated method is substantially faster than standard algorithms on practical problems.

Key words. boundary-element methods, Laplace's equation, potential theory

AMS subject classifications. 65N38, 45L10

1. Introduction. Mixed first- and second-kind surface integral equations with $\frac{1}{r}$ and $\frac{\partial}{\partial n}\frac{1}{r}$ kernels are generated by a variety of three-dimensional engineering problems. For such problems, Nyström-type algorithms cannot be used directly, but an expansion for the unknown, rather than for the entire integrand, can be assumed and the product of the singular kernel and the unknown integrated analytically. Combining such an approach with a Galerkin or collocation scheme for computing the expansion coefficients is a general approach, but leads to dense matrix problems. In this paper, we focus on accelerating such techniques for purely first-kind integral equations of potential theory, and present an overlapping-block preconditioned Krylov-subspace iterative algorithm for solving the associated dense matrix problem, where a modified multipole algorithm with a novel adaptation scheme is used to compute the iterates. This approach follows along lines originally suggested in [1].

In the section that follows, we review Galerkin and collocation algorithms applied to the first-kind integral equation, as well as Krylov-subspace iterative algorithms for solving the generated dense matrix problem. Our approaches to applying a fast multipole algorithm to this problem are described in §3, and a simplified complexity analysis for our adaptive scheme is given. In §4, we present more general proofs of the adaptive multipole algorithm linear computational growth. The preconditioning strategy for accelerating the Krylov-subspace method convergence is described in §5, and experimental results gained from using the method to analyze a variety of structures derived from engineering problems are presented in §6. Finally, conclusions are given in §7.

2. Formulation. Consider the first-kind integral equation for a single-layer surface density, hereafter referred to as a charge density, generated by solution of the exterior Dirichlet problem in a multiply-connected domain. (For a second-kind formulation of this problem see

*Received by the editors June 22, 1992; accepted for publication (in revised form) February 12, 1993.This work was supported by the Defense Advanced Research Projects Agency contract N00014-91-J-1698, Office of Naval Research contract N00014-90-J-1085, the National Science Foundation contract MIP-8858764 A02, Federal Bureau of Investigation contract J-FBI-88-067, and grants from Digital Equipment Corporation and I.B.M.

†Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139 (ksn@rle-vlsi.mit.edu); (white@rle-vlsi.mit.edu).

‡Department of Ocean Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139 (xmeyer@chf.mit.edu).

§Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139 (ftl@math.mit.edu).

[2].) The charge density σ satisfies the integral equation

$$\psi(x) = \int_S \sigma(x') \frac{1}{\|x - x'\|} da', \qquad x \in S, \tag{1}$$

where S is a two-dimensional surface in $\mathbf{R}^3$, $\psi(x)$ is a given surface potential, da' is the incremental surface area, x, $x' \in \mathbf{R}^3$, and $\|x\|$ is the usual Euclidean length of x given by $\sqrt{x_1^2 + x_2^2 + x_3^2}$. More compactly, we denote (1) by

$$\psi = \mathcal{L}\sigma. \tag{2}$$

The charge density σ can be related to electrostatic capacitances and forces or fluid velocities for the case of potential flow. Electrostatic capacitances are useful figures of merit for designers of electronic packaging [3]; microelectromechanical system designers are interested in electrostatic forces [4]; and ocean vehicle designers are interested in potential flow [5]. For these engineering problems, 0.1%–1% accuracy is typically sufficient, and therefore low-order schemes are in common use.

To compute an approximation to σ, one can generally consider an expansion of the form

$$\sigma(x) \approx \sum_{i=1}^{N} q_i \theta_i(x), \tag{3}$$

where $\theta_1(x), \ldots, \theta_N(x) : \mathbf{R}^3 \to \mathbf{R}$ are a set of not necessarily orthogonal expansion functions, and $q_1, \ldots, q_N$ are the unknown expansion coefficients. Typically, $\theta_1(x), \ldots, \theta_N(x)$ represent an approximate discretization of the surface S, where each θ_i is nonnegative, of compact support, and satisfies a normalization condition

$$\int_S \theta_i(x') da' = 1. \tag{4}$$

The expansion coefficients are then determined by requiring that they satisfy a Galerkin or collocation condition of the form

$$Pq = \overline{p}, \tag{5}$$

where $P \in \mathbf{R}^{N \times N}$ and $\overline{p}, q \in \mathbf{R}^N$. In the case of a Galerkin condition,

$$P_{ij} = \langle \theta_j, \mathcal{L}\theta_i \rangle \tag{6}$$

and

$$\overline{p}_i = \langle \theta_i, \psi \rangle, \tag{7}$$

where $\langle f, g \rangle \equiv \int_S f(x')g(x')da'$. For the collocation condition,

$$P_{ij} = \langle \delta(x_j), \mathcal{L}\theta_i \rangle \tag{8}$$

and

$$\overline{p}_i = \langle \delta(x_i), \psi \rangle, \tag{9}$$

where $\langle \delta(x), u \rangle \equiv u(x)$, and $x_1, \ldots, x_N$ are the collocation points [6].

Remark. If the θ_i's are nonnegative, then for both the Galerkin and collocation methods, P is a positive matrix, and for the Galerkin method, P is symmetric and positive definite. The normalization condition implies that for pairs of θ_i's whose support is widely separated, the associated off-diagonals of P approach $\frac{1}{r}$, where r is the separation distance.

Example. The approach used in many engineering applications is to approximate the surface S with N planar quadrilateral and/or triangular panels over which σ is assumed to be uniform. The expansion functions are then

$$\theta_i(x) = \begin{cases} \frac{1}{a_i} & \text{if } x \text{ is on panel } i, \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

where a_i is the area of panel i. The Galerkin scheme for determining the expansion coefficients yields

$$P_{ij} = \frac{1}{a_i\, a_j} \int_{\text{panel}_i} \int_{\text{panel}_j} \frac{1}{\|x - x'\|}\, dada', \tag{11}$$

and for the collocation method

$$P_{ij} = \frac{1}{a_i} \int_{\text{panel}_j} \frac{1}{\|x_i - x'\|}\, da'. \tag{12}$$

There are closed-form expressions for the integral in (12) [5], but closed-form expressions for the integral in (11) are known only in special cases [3].

The dense linear system of (5) can be solved to compute expansion coefficients from a right-hand side derived from (7) or (9). If Gaussian elimination is used to solve (5), the number of operations is order N^3. Clearly, this approach becomes computationally intractable if the number of expansion functions exceeds several hundred. Instead, consider solving the linear system (5) using a Krylov-subspace method such as generalized minimal residual (GMRES)[7]. Such methods have the general form of Algorithm 2.1.

ALGORITHM 2.1. General Krylov-subspace algorithm for solving (5).
Set q^0 to some initial guess at the solution.
Compute the initial residual and Krylov vector, $r^0 = p^0 = \overline{p} - Pq^0$.
Determine the value of a cost function (e.g., $\|r^0\|$.)
Set $k = 0$.
repeat {
  if (cost < *tolerance*), return q^k as the solution.
  $p^{k+1} = Pp^k$.
  Choose α's and β in
    $q^{k+1} = \sum_{j=0}^{k} \alpha_j p^j + \beta p^{k+1}$
  to minimize the cost function.
  Set $k = k + 1$.
}

The dominant costs of Algorithm 2.1 are in calculating the N^2 entries of P using (6) or (8) before the iterations begin and in performing N^2 operations to compute Pp^k on each iteration. Described below are modified and adaptive multipole algorithms that avoid forming most of P and reduce the cost of forming Pp^k to order N operations. This does not necessarily imply that each iteration of a GMRES-style algorithm can be computed with order N operations. If the number of GMRES iterations required to achieve convergence approaches N, then to perform the minimization in each GMRES iteration will require order N^2 operations. This

problem is avoided through the use of a preconditioner, also described below, that in practice reduces the number of GMRES iterations required to achieve convergence to well below N for large problems.

Remark. As the Galerkin matrix P is symmetric and positive definite, it is tempting to replace a Krylov-subspace method suitable for nonsymmetric problems with the conjugate gradient algorithm. Unfortunately, this is not recommended because if a multipole algorithm is used to approximately compute Pp^k, this is equivalent to using an iterative method to solve $\tilde{P}q = \overline{p}$, where $\tilde{P}$ is the multipole algorithm's *not necessarily symmetric*, but sparse, approximation to P.

3. Multipole algorithms. In the case of collocation, computing the dense matrix-vector product Pq is equivalent to evaluating the potential at N collocation points, $\{x_1, \ldots, x_N\}$, due to a charge density described by $\sum_{i=1}^{N} q_i \theta_i(x)$. It is possible to avoid forming P, and to substantially reduce the cost of computing Pq, using the fast multipole algorithm [8], [9]. The fast multipole algorithm uses a hierarchical partitioning of the problem domain and careful application of multipole and local expansions to accurately compute potentials at N points due to N charged particles in order N operations.

In particular, in the fast multipole algorithm, potentials due to clusters of charges are represented by truncated multipole expansions. These expansions have the general form

$$\psi(r, \theta, \phi) \approx \sum_{n=0}^{l} \sum_{m=-n}^{n} \frac{M_n^m}{r^{n+1}} Y_n^m(\theta, \phi), \tag{13}$$

where l is the expansion order, r, θ, and ϕ are the spherical coordinates with respect to the multipole expansion's origin (usually the center of the charge cluster), $Y_n^m(\theta, \phi)$'s are the surface spherical harmonics, and the M_n^m's are the multipole coefficients [10].

Multipole expansions can be used to efficiently evaluate the potential due to a cluster of charges at any point where the distance between the evaluation point and the cluster center is significantly larger than the radius of the cluster. A dual optimization is possible using local expansions. That is, for a cluster of evaluation points, the potential due to charges whose distances from the cluster center are significantly larger than the radius of the cluster can be combined into a local expansion at the cluster center. Then, this local expansion can be used to efficiently compute potentials at evaluation points in the cluster. A local expansion has the form

$$\psi(r, \theta, \phi) \approx \sum_{n=0}^{l} \sum_{m=-n}^{n} L_n^m Y_n^m(\theta, \phi) r^n, \tag{14}$$

where l is the order of the expansion, r, θ, and ϕ are the spherical coordinates of the evaluation location with respect to the expansion center, and the L_n^m's are the local expansion coefficients.

For a more complete introduction to the three-dimensional fast multipole algorithm, refer to the references. In the remainder of this section, we focus instead on the several aspects of the fast multipole algorithm that must be modified to create an efficient algorithm for computing the matrix-vector products associated with using an iterative scheme to solve the discretized integral equations of potential theory. For such problems, the charge is a surface density given by $\sum_{i=1}^{N} q_i \theta_i(x)$, rather than a set of point charges, and this mildly complicates the procedure for computing multipole expansion coefficients. In addition, using an iterative algorithm to solve the discretized integral equation implies that the multipole algorithm is used many times to compute potentials, but in iteration computation, only the coefficients of the charge density expansion functions change. This implies that efficiency can be improved if quantities with only geometric dependencies are computed once and stored. Finally, the spatial nonuniformity

associated with surface discretizations can be efficiently handled using an adaptive algorithm different from that given in [11].

3.1. Computing multipole coefficients. In general, the coefficients of a multipole expansion for a charge density σ in a volume V are given by

$$M_n^m = \int_V \sigma(\rho, \alpha, \beta)\rho^n Y_n^{-m}(\alpha, \beta)dV, \tag{15}$$

where ρ, α, and β represent position in spherical coordinates. In the case where the charge density is given by $\sum_{i=1}^{N} q_i\theta_i(x)$, substitution in (15) leads to

$$M_n^m = \sum_{i=1}^{N} q_i \int_V \theta_i(\rho, \alpha, \beta)\rho^n Y_n^{-m}(\alpha, \beta)dV. \tag{16}$$

Note that the relationship between the multipole coefficients and the density expansion coefficients is linear, assuming that the geometric quantities are given.

In general, the multipole coefficients can be computed easily using quadrature formulas; the integrand in (16) contains no singularities. It is also possible to derive a closed form expression for the integral in (16) for the case of the piecewise-constant expansion function in (10).

THEOREM 3.1. *Let Q be any triangular or quadrilateral region of a plane in $\mathbf{R}^3$. There exists a closed form expression for*

$$\int_Q \rho^n Y_n^{-m}(\alpha, \beta)da, \tag{17}$$

where ρ, α, β are the spherical coordinates of points on the panel surface.

Proof. Formulas for shifting and rotating spherical harmonics are already well known, having been derived for problems in quantum angular momentum [12]. It is therefore sufficient to demonstrate a closed form expression for (17) in the case where the coordinate system origin is coincident with the panel centroid, and the coordinate system x_1-x_2 plane is coplanar with the panel.

To begin, note that by definition,

$$Y_n^m(\phi, \theta) = \sqrt{\frac{(n - |m|)!}{(n + |m|)!}} P_n^{|m|}(\cos\theta)e^{im\phi}. \tag{18}$$

Substituting into (17) and noting that the coordinate system assumption implies that $\cos\beta = 0$ for any point on the panel surface,

$$\int_Q \rho^n Y_n^{-m}(\alpha, \beta)da = \sqrt{\frac{(n - |m|)!}{(n + |m|)!}} P_n^{|m|}(0) \int_Q \rho^n e^{im\phi}da. \tag{19}$$

For $k = n - |m|$ odd, $P_n^{|m|}(0) = 0$ and therefore (19) evaluates to zero. For $k = n - |m|$ even, $\rho^n e^{im\phi}$ can be expanded in terms of x_1 and x_2 as

$$\rho^n e^{im\phi} = \sum_{j=0}^{\frac{n-|m|}{2}} \binom{\frac{n-|m|}{2}}{\frac{n-|m|}{2} - j} \sum_{k=0}^{|m|} (-\mathrm{sgn}(m)i)^{|m|-k} \binom{|m|}{|m| - k} x_1^{2j-k} x_2^{n-(2j+k)}. \tag{20}$$

The terms

$$\mathcal{I}_{j,k} \equiv \int_Q x_1^j x_2^k da \tag{21}$$

are the moments of the panel for which analytic formulas can be found in [5]. Using the moments and combining (20) and (19) leads to

$$M_n^m = K_n^m \sum_{j=0}^{\frac{n-|m|}{2}} \binom{\frac{n-|m|}{2}}{\frac{n-|m|}{2}-j} \sum_{k=0}^{|m|} (-\mathrm{sgn}(m)i)^{|m|-k} \binom{|m|}{|m|-k} \mathcal{I}_{2j-k,n-(2j+k)}, \tag{22}$$

where

$$K_n^m = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}} P_n^{|m|}(0). \qquad \square \tag{23}$$

3.2. Matrix representation. As mentioned above, the iterative algorithm described here uses the fast multipole algorithm to compute matrix-vector products. This implies that the algorithm is used many times without a change in geometry, and this fact can be exploited to improve algorithm efficiency. As is made clearer below, the fast multipole algorithm involves constructing multipole expansions from charge density expansions, shifting and combining multipole expansions, converting multipole expansions and charge density expansions into local expansions, shifting local expansions, and evaluating the potential due to charge density expansions, multipole expansions, or local expansions. However, each of the construction, translation, and evaluation operations is a linear function of the expansion coefficients, and therefore can be represented as a matrix whose elements are functions of geometry alone [13].

From the above observation we have the following theorem.

THEOREM 3.2. *The fast multipole algorithm computes an approximation to the potential due to a charge density that is a linear function of the density expansion coefficients. In addition, it is possible to represent this linear transformation directly in terms of translation matrices whose entries are functions only of geometry.*

The fast multipole algorithm produces an approximation to Pq, denoted $\tilde{P}q$, which is a linear function of q. That is, the fast multipole algorithm is directly analogous to a sparse representation of the dense matrix P. This has several important consequences. When the multipole algorithm is used to compute matrix-vector products in an iterative method, the iterative method's convergence is controlled by the properties of $\tilde{P}$, not by how well $\tilde{P}$ approximates P. Also, the translation matrices need be computed only once, then used repeatedly until the iteration converges.

3.3. The modified multipole algorithm. Mostly to establish notation, we give a modified multipole algorithm that closely follows the development in [8]. The algorithm allows for the charge density to be described in terms of expansion functions, and also exploits Theorem 3.2 by exclusively using translation matrices.

To begin, we consider the hierarchical domain partitioning. Let the root cube be the smallest cube containing the problem domain. More precisely, the root cube is the smallest cube that contains all the collocation points and for which x outside the cube implies $\theta_i(x) = 0$ for all i. The hierarchy is then just a recursive eight-way sectioning of this root cube.

DEFINITION 3.1. *Cube hierarchy. The cube containing the problem domain is referred to as the level* 0, *or root, cube. Then, the volume of the cube is subdivided into eight equally sized child cubes, referred to as level-*1 *cubes, and each has the level-*0 *cube as its parent. The collocation points are distributed among the child cubes by associating a collocation point*

*with a cube if the point is contained in the cube. Each of the level-*1 *cubes is then subdivided into eight level-*2 *child cubes and the collocation points are again distributed. The result is a collection of* 64 *level-*2 *cubes and a* 64*-way partition of the collocation points. This process is repeated to produce D levels of cubes and D distributions of collocation points starting with an* 8*-way partition and ending with an* 8^D*-way partition. The depth D is chosen so that the maximum number of* θ_i*'s whose support intersects any finest level cube is less than a selected constant* (*see Definition* 3.11).

The terms below are used to concisely describe the modified multipole algorithm.

DEFINITION 3.2. *Evaluation points of a cube. The collocation points within the cube.*

DEFINITION 3.3. *Nearest neighbors of a cube. Those cubes that have a corner in common with the given cube.*

DEFINITION 3.4. *Second-nearest neighbors of a cube. Those cubes that are not nearest neighbors but that have a corner in common with a nearest neighbor of the given cube.*

Note that there are at most 124 nearest and second-nearest neighbors of a cube, excluding the cube itself.

DEFINITION 3.5. *Interaction set of a cube. The set of cubes that are either the second nearest neighbors of the given cube's parent, or are children of the given cube's parent's nearest neighbors, excluding nearest or second-nearest neighbors of the given cube.*

There is a maximum of 189 cubes in an interaction set. Roughly half of the cubes are from a level one coarser than the level of the given cube; the rest are on the same level.

For the jth cube on level d: its parent's index on level $d-1$ is denoted $F(d, j)$; its set of $d+1$ level children is denoted $C(d, j)$; its set of of interaction cubes is denoted $I(d, j)$; the set of cube j and cube j's nearest and second-nearest neighbors is denoted $N(d, j)$; the vector of multipole expansion coefficients representing the charge density in the cube is denoted $M_{d,j}$; the vector of local expansion coefficients for the cube is denoted $L_{d,j}$; the vectors of the cube's charge density expansion coefficients and collocation point potentials are denoted $q_{d,j}$ and $p_{d,j}$, respectively. The matrix that represents the conversion of a multipole expansion from the level $\tilde{d}$ cube $\tilde{j}$ center to a local expansion at the level d cube j center is denoted $M2L(d, j, \tilde{d}, \tilde{j})$. The $L2L$, $M2M$, and $Q2P$ translation matrices are similarly specified. Finally, the matrix that maps $q_{\tilde{d},\tilde{j}}$ to $M_{d,j}$ is denoted $Q2M(d, j, \tilde{d}, \tilde{j})$, and the $L2P$ matrix is similarly specified.

ALGORITHM 3.1. The Modified Multipole Algorithm.

/* THE CONSTRUCTION PHASE: Computes multipole expansions at the finest level. */

For each level D cube $j = 1$ to 8^D

$M_{D,j} = Q2M(D, j, D, j)q_{D,j}$

/* THE UPWARD PASS: Computes multipole expansions. */

For each level $d = D-1$ to2

For each level d cube $j = 1$ to 8^d

$M_{d,j} = \sum_{\tilde{j}\in C(d,j)} M2M(d, j, d+1, \tilde{j})M_{d+1,\tilde{j}}$

/* THE INTERACTION PHASE: Converts multipole expansions to local expansions. */

For each level $d = 2$ to D

For each level d cube $j = 1$ to 8^d

$L_{d,j} = \sum_{\tilde{d},\tilde{j}\in I(d,j)} M2L(d, j, \tilde{d}, \tilde{j})M_{\tilde{d},\tilde{j}}$

/* THE DOWNWARD PASS: Transfers and accumulates local expansions. */

For each level $d = 3$ to D

For each level d cube $j = 1$ to 8^d

$L_{d,j} = L_{d,j} + L2L(d, j, d-1, F(d, j))L_{d-1,F(d,j)}$

/* THE EVALUATION PHASE: Evaluates the potential. */
For each level D cube $j = 1$ to 8^D
$p_{D,j} = L2P(D, j)L_{D,j} + \sum_{\tilde{j} \in N(D,j)} Q2P(D, j, D, \tilde{j})q_{D,j}$

3.4. The adaptive multipole algorithm. Almost certainly, the surfaces of a given three-dimensional problem will not fill the volume of the root cube, and this will necessarily lead to a nonuniform distribution of the support of the θ_i's. It is possible to derive an adaptive multipole algorithm from Algorithm 3.1 by breaking up the problem domain nonuniformly. In this case, when the cube hierarchy is created, cubes are not subdivided if they intersect the support of fewer than some limiting number of θ_i's [11]. However, for the kinds of charge density expansion functions in common use, such an approach can sometimes require more computation than a nonadaptive algorithm [14]. A more effective approach in this setting, one which is guaranteed to use fewer operations than a nonadaptive algorithm, is to avoid translation from charge density expansion coefficients to multipole expansions, and to avoid forming local expansions, whenever such representations are inefficient. For example, consider computing $M_{D,j}$ from $Q2M(D, j, D, j)q_{D,j}$. If the number of entries in vector $q_{D,j}$ is smaller than the number of multipole coefficients in $M_{D,j}$, which for an lth-order multipole expansion is $(l+1)^2$, then $q_{D,j}$ is a more efficient representation of the charge distribution, and that representation can be propagated through the algorithm instead of $M_{D,j}$. Note that using such an approach requires some rather straight-forward bookkeeping and a few easily derived translation operators [13]. In particular, multipole coefficient to evaluation point and charge to local expansion coefficient translation operators, denoted $M2P$ and $Q2L$, respectively, are required.

A second optimization for the nonuniform case, one that is guaranteed to reduce the operation count during the upward pass and also improve accuracy slightly, is to exploit the fact that there is no need to construct a multipole expansion for a cube with only one nonempty child. This is made clear in the following remark.

Remark. Suppose the charge density in a given cube is entirely contained in one of the cube's descendants. Then the potential due to charge in the given cube is at least as accurately represented using the descendant's multipole expansion about the descendant's center, as by a shifted version of the descendant's multipole expansion about the given cube's center.

A similar optimization can be used to improve the efficiency of the downward pass. That is, there is no advantage to creating a local expansion for a cube with only one nonempty child. Instead, the multipole expansions associated with the members of a cube's interaction set can be translated directly to the cube's only nonempty child.

To describe an adaptive algorithm that exploits the above optimizations, the following additional definitions are used.

DEFINITION 3.6. *Adaptive cube. Any nonempty finest level cube or a coarser level cube that has more than one nonempty child.*

DEFINITION 3.7. *Adaptive child. An adaptive child of a given cube is any adaptive cube descendent which has no ancestors that are adaptive cube descendents of the given cube. The set of adaptive children of level d cube with index j is denoted* $C^A(d, j)$.

Note that a cube need *not* be adaptive to have adaptive children. However, if a nonempty cube is not an adaptive cube, $|C^A(d, j)|$, which denotes the number of elements in $C^A(d, j)$, is precisely one.

DEFINITION 3.8. *Adaptive parent. The adaptive parent of a given adaptive cube is the unique adaptive cube for which the given adaptive cube is an adaptive child. The adaptive parent of a level d adaptive cube j is denoted by* $F^A(d, j)$.

Note that an adaptive parent can be separated from an adaptive child by an arbitrary number of levels.

DEFINITION 3.9. *Adaptive interaction set. The adaptive interaction set of a cube is the set of all members of the cube interaction set, except any nonadaptive cube member is replaced with its adaptive child. The adaptive interaction set is denoted* $I^A(d, j)$.

ALGORITHM 3.2. Adaptive Multipole Algorithm.

/* THE CONSTRUCTION PHASE: Computes multipole expansions at the finest level. */

For each level D nonempty cube j

if $(size(q_{D,j}) > (l+1)^2)$ $M_{D,j} = Q2M(D, j, D, j)q_{D,j}$.

/* THE UPWARD PASS: Computes multipole expansions. */

For each level $d = D-1$ to 2

For each level d *adaptive* cube j

if $(size(q_{d,j}) > (l+1)^2)$ {

For each level $\tilde{d}$ cube $\tilde{j} \in C^A(d, j)$

if $(size(q_{\tilde{d},\tilde{j}}) > (l+1)^2)$ $M_{d,j} = M_{d,j} + M2M(d, j, \tilde{d}, \tilde{j})M_{\tilde{d},\tilde{j}}$

else $M_{d,j} = M_{d,j} + Q2M(d, j, \tilde{d}, \tilde{j})q_{\tilde{d},\tilde{j}}$

}

/* THE INTERACTION PHASE: Converts multipole expansions to local expansions. */

For each level $d = 2$ to D

For each level d cube j for which $|I^A(d, j)| > 0$

if $(|C^A(d, j)| > 1)$ $\hat{d} = d$ and $\hat{j} = j$.

else $\hat{d}, \hat{j}$ is the only member of $C^A(d, j)$.

if $(size(p_{d,j}) > (l+1)^2)$ {

For each $\tilde{d}, \tilde{j} \in I^A(d, j)$

if $(size(q_{\tilde{d},\tilde{j}}) > (l+1)^2)$ $L_{\hat{d},\hat{j}} = L_{\hat{d},\hat{j}} + M2L(\hat{d}, \hat{j}, \tilde{d}, \tilde{j})M_{\tilde{d},\tilde{j}}$.

else $L_{\hat{d},\hat{j}} = L_{\hat{d},\hat{j}} + Q2L(\hat{d}, \hat{j}, \tilde{d}, \tilde{j})q_{\tilde{d},\tilde{j}}$.

}

else {

For each $\tilde{d}, \tilde{j} \in I^A(d, j)$

if $(size(q_{\tilde{d},\tilde{j}}) > (l+1)^2)$ $p_{d,j} = p_{d,j} + M2P(d, j, \tilde{d}, \tilde{j})M_{\tilde{d},\tilde{j}}$.

else $p_{d,j} = p_{d,j} + Q2P(d, j, \tilde{d}, \tilde{j})q_{\tilde{d},\tilde{j}}$.

}

/* THE DOWNWARD PASS: Transfers and accumulates local expansions. */

For each level $d = 3$ to D

For each level d *adaptive* cube j

$\tilde{d}, \tilde{j} = F^A(d, j)$

if $(size(p_{d,j}) > (l+1)^2)$

if $(\tilde{d} > 1)$ $L_{d,j} = L2L(d, j, \tilde{d}, \tilde{j})L_{\tilde{d},\tilde{j}}$

else if $(size(p_{\tilde{d},\tilde{j}}) > (l+1)^2)$

if $(\tilde{d} > 1)$ $p_{d,j} = L2P(d, j, \tilde{d}, \tilde{j})L_{\tilde{d},\tilde{j}}$

/* THE EVALUATION PHASE: Evaluates the potential. */

For each level D nonempty cube j

if $(size(p_{D,j}) > (l+1)^2)$ $p_{D,j} = L2P(D, j)L_{D,j}$

$p_{D,j} = p_{D,j} + \sum_{\tilde{j} \in N(D,j)} Q2P(D, j, D, \tilde{j})q_{D,\tilde{j}}$

Since the adaptive algorithm is derived by reducing or avoiding operations in a nonadaptive algorithm, the theorem below follows directly.

THEOREM 3.3. *Algorithm* 3.2 *always uses fewer operations than Algorithm* 3.1.

3.5. Complexity analysis. It is clear from the description of both the nonadaptive and adaptive multipole algorithms that they can require more than order N operations if expansion functions with arbitrary support are used to represent the charge density. Although there are techniques that use sets of expansion functions whose support is the entire surface, we will insist on the following two conditions.

DEFINITION 3.10. *Compactness condition. The problem domain and $\theta_1, \ldots, \theta_N$ are such that there exists a cube hierarchy of depth D for which the number of cubes intersecting the support of each θ_i is bounded by K_{ct}, independent of N.*

DEFINITION 3.11. *Bounded overlap condition. The problem domain and $\theta_1, \ldots, \theta_N$ are such that there exists a cube hierarchy of depth D for which the number of θ_i's whose support intersects each finest-level cube i is bounded by K_{tc}, independent of N.*

Previous approaches to bounding the computation of adaptive multipole algorithms assumed an a priori bound on on the maximum number of levels in a cube hierarchy, denoted $\tilde{D}$ [11]. Such an assumption is loosely justified by the notion that given a machine precision, there is a smallest representable finest-level cube. The precision assumption has the suspicious consequence that in any cube hierarchy, each cube has a bounded number, in fact $\tilde{D}$, ancestors. Given this, simple examples can lead to seeming contradictions. Consider the surface S to be a flat square plate of unit area, and suppose the surface charge density on the plate is represented with expansion functions corresponding to small square panels over which the charge density is assumed constant (see (10)). If a $\sqrt{N} \times \sqrt{N}$ array of such square panels is used to discretize the plate, then the finest-level cube size required to satisfy the compactness and bounded overlap conditions must be order $\sqrt{N}$ in diameter. But this requires a partitioning depth of order $\log N$, violating the a priori bound on the depth implied by the precision assumption.

Nevertheless, the multipole algorithm completes in order N operations for the square plate example. For that example, and for typical problems associated with the discretization of two-dimensional surfaces in three-dimensional domains, a perhaps more relevant assumption is the hierarchical contraction condition given below.

DEFINITION 3.12. *Hierarchical contraction condition. The problem domain and $\theta_1, \ldots, \theta_N$ are such that the number of nonempty cubes at level $d-1$ is less than γ times the number of nonempty cubes at level d, where γ is strictly less than one and independent of N.*

The importance of the hierarchical contraction condition is made clear in the following lemma.

LEMMA 3.4. *If the hierarchical contraction condition is satisfied, then the number of nonempty cubes, summed over all the levels, is bounded by*

$$N * K_{ct} * \beta, \tag{24}$$

where $\beta \equiv \frac{1}{1-\gamma}$ is independent of N.

Proof. If the hierarchical contraction condition is satisfied, then, by definition, the total number of nonempty cubes summed over all the levels is

$$\sum_{i=0}^{D} N * K_{ct} * \gamma^i < N * K_{ct} * \sum_{i=0}^{\infty} \gamma^i = N * K_{ct} * \beta. \tag{25}$$

□

Given that a problem satisfies the hierarchical contraction condition, it is easily shown that Algorithm 3.2 completes in order N operations. It is also possible to prove such a result without this assumption, and we return to the general case in §4.

THEOREM 3.5. *Given the hierarchical contraction condition, if the problem domain and $\theta_1, \ldots, \theta_N$ are such that there exists a cube hierarchy of depth D for which the compactness and bounded overlap conditions above are satisfied, then the nonadaptive modified multipole*

algorithm (if empty cubes are ignored), and the adaptive multipole algorithm, complete in order N operations.

Proof. To prove the theorem, operations are counted for each of the five steps of Algorithm 3.1, assuming that empty cubes are ignored. Then by Theorem 3.3, the result is a also a bound on the number of operations for Algorithm 3.2.

Construction phase. Each N θ_i contributes to $(l+1)^2$ terms in the multipole expansion of at most K_{ct} cubes.

$$\text{Cost} \leq N * K_{ct} * (l+1)^2.$$

The upward pass. By Lemma 3.4, the total number of nonempty cubes summed over all the levels is bounded by $N * K_{ct} * \beta$. In the upward pass, each nonempty cube at each level is associated with one $M2M$ translation to its nonempty parent, so there is a total of no more than $N * K_{ct} * \beta$ $M2M$ operations, each of which costs $(l+1)^4$ operations.

$$\text{Cost} \leq N * K_{ct} * \beta * (l+1)^4.$$

The interaction phase. Again by Lemma 3.4, the total number of nonempty cubes summed over all the levels is bounded by $N * K_{ct} * \beta$. Each nonempty member of a nonempty cube interaction set is associated with one $M2L$ translation, and there are no more than 189 nonempty members in a cube interaction set. Therefore, in the interaction phase, there is a total of no more than $N * K_{ct} * \beta * 189$ $M2L$ operations, each of which costs $(l+1)^4$ operations.

$$\text{Cost} \leq N * K_{ct} * \beta * (l+1)^4 * 189.$$

The downward pass. Again by Lemma 3.4, the total number of nonempty cubes is bounded by $N * K_{ct} * \beta$, and as each nonempty cube is associated with one $L2L$ translation from its nonempty parent, there is a total of no more than $N * K_{ct} * \beta$ $L2L$ operations, each of which costs $(l+1)^4$ operations, in the downward pass.

$$\text{Cost} \leq N * K_{ct} * \beta * (l+1)^4.$$

The evaluation phase. For each collocation point in a finest-level cube j, the contribution of the local expansion to the potential, which costs $(l+1)^2$ operations to evaluate, must be added to the potential due to the charge density associated with the fewer than $125 * K_{tc}$ θ_i's whose support intersects cube j.

$$\text{Cost} \leq N * (125 * K_{tc} + (l+1)^2).$$

Adding the costs leads to an order N bound on the total number of operations,

$$\text{Total Cost} \leq N * \left[\left((K_{ct}+1) * (l+1)^2\right) + \left(K_{ct} * \beta * 191 * (l+1)^4\right) + (125 * K_{tc})\right]. \quad \square \tag{26}$$

Note that for the flat square plate with a square panel discretization example mentioned above, $\gamma = 0.25$, and therefore $\beta = \frac{4}{3}$.

One last aspect of the multipole algorithm in this context must be considered. For efficiency, it is assumed that all the translation matrices will be computed once and stored, so they can be rapidly reapplied during an iterative matrix solution algorithm. This suggests that the question of the required memory be addressed. Since each application of the multipole algorithm uses every translation matrix element, the theorem below follows easily.

THEOREM 3.6. *If the multipole algorithm completes in order N operations, then storing the translation matrices requires order N storage.*

4. General theorem for Algorithm 3.2. In §3, it is claimed that the number of operations in Algorithm 3.2 is order N. However, to simplify the proof of the bounds on the upward pass, the downward pass and the interaction phase of Algorithm 3.2, a hierarchical contraction condition was assumed. In this section, we give more general proofs of these bounds.

To simplify notation, results in this section will be given in terms of the number of nonempty finest-level cubes, denoted M. Using the notation of §3, it is clear that $M \leq (N * K_{ct})$.

To bound the number of operations in the upward and downward passes of Algorithm 3.2, we make use of a graph that can represent either pass. The graph tree structure leads naturally to the bounds using results from elementary graph theory (see, for example, [15]).

DEFINITION 4.1. *Adaptive upward-pass graph. Let each adaptive cube in each level of the cube hierarchy correspond to a node in a graph, and insert an edge between pairs of nodes if one of the associated adaptive cubes is an adaptive child of the other. The resulting graph is the adaptive upward-pass graph.*

Since an edge connects every adaptive child cube to its parent, there is a one-to-one correspondence between the edges in the adaptive upward-pass graph and the $M2M$ or $Q2M$ translation operations in the upward pass of Algorithm 3.2. Similarly, there is also a one-to-one correspondence between the edges in the adaptive upward-pass graph and the $L2L$ or $L2P$ translation operations in the downward pass of Algorithm 3.2.

LEMMA 4.1. *The adaptive upward-pass graph is a tree with M leaves, or childless nodes, and every nonleaf node in the tree except the root has at least three edges connected to it. That is, each nonleaf node except the root has degree greater than two.*

Proof. A graph that is connected and has e edges and n nodes with $e = n - 1$ is a tree. The adaptive upward-pass graph is connected because any two nodes in the graph correspond to two cubes that are parts of at least one larger cube. A path from any particular node to another node can always be found passing through the node corresponding to the larger cube or directly between the two nodes of given cubes if one of the two given cubes is contained in the other.

Furthermore, every node represents an adaptive child cube and therefore has a single edge connecting it to the node corresponding to its adaptive parent, except the node corresponding to the level 0 cube. Since this level 0 node has no such edge, $e = n - 1$. Thus the adaptive upward-pass graph is a tree, and the level 0 node may be taken as its root.

The definition of an adaptive cube implies that each nonroot and nonfinest-level adaptive cube must have an adaptive parent and at least two adaptive children. Therefore, every associated nonroot and nonleaf node in the adaptive upward-pass graph has degree greater than two. The leaves of the upward-pass graph obviously correspond to the M nonempty finest-level cubes since these adaptive cubes have no children. □

The bounds are a direct consequence of the upward-pass graph tree structure as summarized by Lemma 4.1.

THEOREM 4.2. *The total number of $M2M$ or $Q2M$ operations in the upward pass, as well as the total number of $L2L$ or $L2P$ operations in the downward pass, is bounded by $2 * M$, where M is the number of nonempty finest-level cubes.*

Proof. Since there is a one-to-one correspondence between each edge in the adaptive upward-pass graph and either upward-pass or downward-pass translation operations, the theorem can be proved by demonstrating that the upward-pass graph has no more than $2 * M$ edges. Since the sum of the degrees of any graph's nodes is equal to twice the number of edges in the graph (each edge connects exactly two nodes), Lemma 4.1 implies

$$M + 3(n - M - 1) + 2 \leq 2e, \tag{27}$$

in the case of an upward-pass graph with n nodes and e edges. The first term is the sum of the leaf-node degrees, the second corresponds to the sum of minimum nonleaf node degrees, and the last is the minimum degree of the root. Since $e = n - 1$,

$$e \leq 2(M - 1). \qquad \square \tag{28}$$

The following theorem addresses the computational complexity of the interaction phase in Algorithm 3.2. The interaction phase uses $M2L$, $Q2L$, $M2P$, and $Q2P$ operations to translate information between cubes in each interaction set. We now prove that the total number of these interaction operations is of the order of the number of nonempty finest-level cubes. To avoid somewhat less interesting complications that arise when an interaction set contains cubes on two different levels, the theorem is proved using the definition of an interaction set given in the original description of the fast multipole algorithm [8]. To avoid confusion, we refer to this set as the regular interaction set.

DEFINITION 4.2. *Regular interaction set of a cube. Those cubes that are children of the given cube's parent's nearest and second-nearest neighbors, excluding nearest or second-nearest neighbors of the given cube.*

Unlike the interaction set given in Definition 3.5, the regular interaction set of a cube has many more members (875 rather than 189) but all the members are cubes from the same level as the given cube. Clearly, using the regular interaction set in Algorithm 3.2 only increases the required computation, so the upper bounds on computation derived below are also upper bounds on the interaction phase of Algorithm 3.2.

A second useful definition describes one approach to merging regular interaction sets of a cube and its descendents.

DEFINITION 4.3. *Generalized interaction set of a cube. Those cubes at the same level as the given cube that are either members of the given cube's regular interaction set or that contain descendents who are members of the regular interaction set of one of the given cube's descendents. For cubes with no descendents, that is, those cubes on the finest level of a hierarchy, the generalized interaction set is defined by assuming that the hierarchy extends to an additional finer level.*

The generalized interaction set of a cube can also be described as the union of the cube's regular interaction set with the cube's nearest and second-nearest neighbors. Or, equivalently, the generalized interaction set of a cube contains the children of the cube's parent, excluding the given cube, and the children of the parent's nearest and second-nearest neighbors. Therefore, the generalized interaction set for a cube contains no more than 999 members, all at the level of the given cube, and a cube is in no more than 999 generalized interaction sets (cubes near the sides or corners of the problem domain may have fewer members in their interaction sets).

THEOREM 4.3. *Regardless of the distribution or number of levels in the cube hierarchy, the number of interaction operations in the interaction phase of Algorithm* 3.2 *is bounded by*

$$1998 * M, \tag{29}$$

where M is the number of nonempty finest-level cubes.

Proof. Given a nonempty cube at the Dth, or finest, level in the cube hierarchy, there is at most one interaction operation associated with each of up to 999 members of the given cube's generalized interaction set. And as each nonempty cube is a member of at most 999 generalized interaction sets, each cube is associated with at most 999 interaction operations. This may seem to be an unnecessarily generous count, as on this finest level there are only interaction operations between a nonempty cube and the nonempty members in the cube's *regular* interaction set. However, for the purposes of establishing an inductive argument, we

consider the possibility of an interaction operation associated with any member of a cube's *generalized* interaction set.

Now consider cubes at level $D-1$. Suppose that for each of the nonempty $D-1$ level cubes, all but one of the cube's nonempty Dth level children are emptied. We denote the number of emptied cubes as p_D, where p_D is also equal to the difference between the number of nonempty cubes at level D and the number of nonempty cubes at level $D-1$. And as the p_D Dth level cubes are emptied in a way that does not completely empty any originally nonempty $D-1$ level cubes, no $D-1$ level or coarser interaction operations will be eliminated or added. In addition, each of the emptied cubes has no more than 999 Dth level cubes in its generalized interaction set, and each emptied cube is contained in at most 999 Dth level generalized interaction sets. Therefore, by emptying the p_D Dth level cubes, fewer than $1998 * p_D$ Dth level interaction translation operations will be eliminated.

Once the p_D cubes have been emptied, each nonempty $D-1$ level cube contains exactly one nonempty child. As a result, given a nonempty $D-1$ level cube, there is at most one interaction operation associated with each member of the given cube's generalized interaction set. This follows from the fact that each nonempty member of a given $D-1$ level cube's generalized interaction set *either* is in the given cube's regular interaction set, *or* contains a single nonempty child that is in the generalized interaction set of the given cube's only nonempty child, *but not both*. If the former is true, the associated interaction operation is between $D-1$ level cubes, and if the latter is true, the associated interaction operation is between Dth level cubes.

The above argument establishes that emptying p_D Dth level cubes, and eliminating the associated interaction operations, results in a set of $D-1$ level cubes which have the property that given a nonempty $D-1$ level cube, there is at most one interaction operation associated with each member of the given cube's generalized interaction set. This was the only property about the Dth level cubes used in the above argument, so the argument can be reapplied to levels $D-1$, $D-2$, through to the coarsest level. In inductively applying the above argument at level $d-1$, p_d nonempty d level cubes will be emptied, and, in doing so, no more than $1998 * p_d$ interaction operations will be eliminated. In addition, since only nonempty cubes will be emptied, each emptied cube must correspond to at least one of M nonempty finest-level cubes, and therefore $\sum_{d=1}^{D} p_d \leq M$. Finally, as there are no interaction operations on the coarsest level, all the interaction operations are eliminated by continuing the induction to the coarsest level. Therefore, the total number of interaction operations in the interaction phase of Algorithm 3.2 is bounded by

$$\sum_{d=1}^{D} 1998 * p_d \leq 1998 * M. \qquad \square \tag{30}$$

5. Preconditioning the iterative method. In general, the convergence of a Krylov-subspace method can be significantly accelerated by preconditioning if there is an easily computed approximation to the problem's inverse. We denote the approximation to P^{-1} in (5) by $\tilde{C}$, in which case preconditioning is equivalent to solving

$$P\tilde{C}x = \overline{p} \tag{31}$$

for the unknown vector x, from which the vector of expansion coefficients is determined by $q = \tilde{C}x$. Clearly, if $\tilde{C}$ is precisely P^{-1}, then (31) is trivial to solve, but $\tilde{C}$ will be very expensive to compute.

In [16] and [17], it was suggested that a good approximation to P^{-1} could be derived by solving overlapping subproblems. Such an approach fits naturally with the hierarchical

multipole algorithm because the preconditioner can be constructed and applied in a cube-by-cube fashion. First, $\tilde{C}$ is computed by inverting a sequence of reduced P matrices, one associated with each finest-level cube, as in Algorithm 5.1 below.

ALGORITHM 5.1. Forming $\tilde{C}$.
For each finest-level nonempty cube j
/* Form P^j with the Q2P matrices of cube j's nearest neighbors. */
For each $\tilde{j} \in N(D, j)$
For each $\tilde{k} \in N(D, j)$
$P^j_{\tilde{j},\tilde{k}} = Q2P(D, \tilde{j}, D, \tilde{k})$
Compute $\tilde{C}^j = (P^j)^{-1}$.
For each θ_i whose support intersects cubes in $N(D, j)$
if θ_i's support does *not* intersect cube j **delete** the associated row from $\tilde{C}^j$.

The matrix P^j is a block matrix, and each entry $P^j_{\tilde{j},\tilde{k}}$ is a matrix, not a scalar. In general, since the number of collocation points contained in the cubes in $N(D, j)$ will not necessarily be equal to the number of expansion functions whose support intersects the cubes in $N(D, j)$, P^j is not square. Therefore, forming $(P^j)^{-1}$ should be interpreted to imply a generalized inverse.

By comparing Algorithm 5.1 with Algorithm 3.2, it is clear that P^j uses only those elements of the full P matrix that are already required in Algorithm 3.2, and therefore the computational cost in computing the preconditioner is only in inverting small P^j matrices. Then computing the product $P\tilde{C}x^k$, which would be used in a Krylov-subspace method applied to solving (31), is accomplished in two steps. First, the preconditioner is applied to form $q^k = \tilde{C}x^k$ using Algorithm 5.2 below. Then, Pq^k is computed using Algorithm 3.2.

ALGORITHM 5.2. Forming $q = \tilde{C}x$.
For each finest-level nonempty cube j
For each θ_i whose support intersects cube j
For each θ_k whose support intersects cubes in $N(D, j)$
Add $\tilde{C}^j_{i,k}x_k$ to q_i.

The cost of the preconditioner is linked to the cost of the multipole algorithm, provided the collocation points are distributed among the expansion functions in a reasonable manner. The statement is made precise below.

THEOREM 5.1. *If the collocation points are distributed so that associated with each expansion function there is a unique collocation point contained in the expansion function's support, and the expansion functions satisfy the bounded overlap and compact support conditions, then computing the preconditioner requires order N operations.*

Proof. If the collocation points satisfy the distribution condition in Theorem 5.1, and the expansion functions satisfy the bounded overlap condition, then each of the dimensions of each P^j is bounded by $K_{tc} * 125$. From the compact support condition, there are at most $K_{ct} * N$ nonempty cubes. Therefore, computing the preconditioner costs $N * K_{ct} * (K_{tc} * 125)^3$ operations. □

6. Experimental results. In this section, results from computational experiments in solving (1) are presented. The experiments are conducted using FASTCAP [13], a three-dimensional electrostatic analysis program that uses an implementation of the preconditioned GMRES algorithm with adaptive multipole acceleration (PAMA). The program uses the piecewise-constant panel expansion functions given in (10) and a panel centroid collocation scheme. First, idealized examples of potential flow problems are examined to allow a

controlled investigation of aspects of the algorithm, and then results from realistic engineering applications are shown to exhibit the practicality of the method.

6.1. Spheres in potential flow. The potential due to a unit sphere in an infinite fluid translating at unit velocity along the x_3-axis is given by

$$\psi(x) = -\frac{x_3}{2\|x\|^3}. \tag{32}$$

The charge density σ satisfying (1) for the potential in (32) can be determined analytically. To derive the formula, Green's theorem is written twice, once for the known ψ on the sphere's outside, and once for a ψ' (some as yet unknown potential) on the sphere's inside. Summing the two expressions gives

(33)
$$2\pi(\psi(x) + \psi'(x)) + \iint_S [(\psi(x') - \psi'(x'))\nabla G(x, x') - G(x, x')\nabla(\psi(x') - \psi'(x'))] \cdot \hat{n} da' = 0, \quad x \in S,$$

where $\hat{n}$ is the inward directed surface normal and

$$G(x, x') = \frac{1}{\|x - x'\|}. \tag{34}$$

If we let σ satisfy

$$\sigma(x) = \frac{1}{4\pi}\nabla(\psi(x) - \psi'(x)) \cdot \hat{n}, \tag{35}$$

then (1) may be obtained from (33), if ψ' is such that $\psi - \psi'$ vanishes on S; namely,

$$\psi'(x) = -\frac{1}{2}x_3. \tag{36}$$

By substitution of (32) and (36) in (35),

$$\sigma(x) = \frac{-3}{8\pi}x_3, \qquad x \in S. \tag{37}$$

Figure 1 is a plot of a shaded sphere, where the shading corresponds to the charge density given in (37).

Having the solution for the translating sphere problem in closed form allows us to demonstrate the convergence of the complete algorithm. Figure 2 shows that the convergence rate will approach $\frac{1}{N}$ if the tolerance on the convergence of the iterative method is small enough, and if the order to which terms are retained in the spherical harmonic expansions is high enough. The definition of the integrated error is the following summation over the N panels

$$\mathcal{E} = \sum_{i=1}^{N} |q_i - a_i\sigma(x_i)|, \tag{38}$$

where a_i is the area of panel i and $|\frac{q_i}{a_i} - \sigma_i(x)|$ is the error between the computed and exact solution at panel i's collocation point. The results in the figure using lower-order expansions and a larger tolerance for convergence of the iterative method show how these parameters limit the accuracy of the computed solution given a particular spatial discretization.

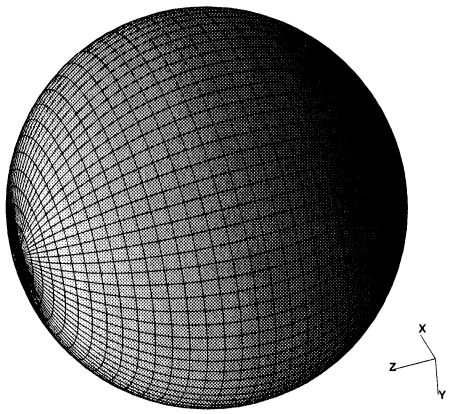

FIG. 1. *The single sphere discretized by* 2592 *panels translating in an infinite fluid. The shading corresponds to the density strength* $\sigma(x)$. *The dark shading at the pole of the sphere is a plotting artifact.*

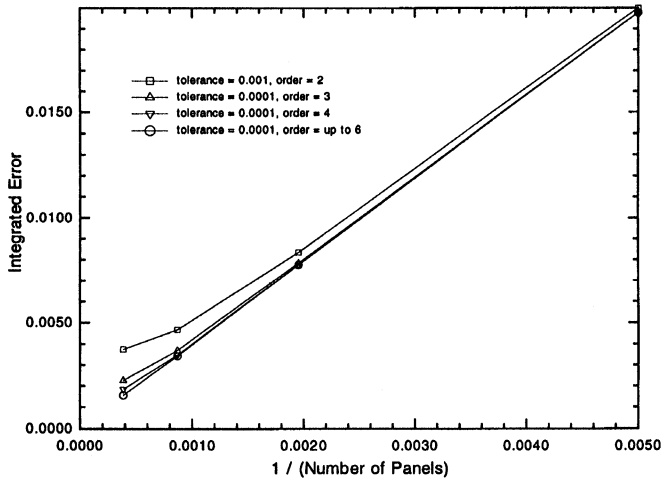

FIG. 2. *Convergence study for the single sphere translating in an infinite fluid. Tolerance refers to the convergence criterion for the iterative method, and order refers to the highest term retained in the spherical harmonic expansions.*

To investigate the advantages of using the adaptive multipole-accelerated (AMA) algorithm and the preconditioner, a more complicated case of two spheres, shown in Fig. 3, is considered. In this case, we do not have a closed form solution, so a Dirichlet problem is contrived by applying the known potential of the single sphere case to each of the two spheres. The difference in the cost of computing Pq directly and with the adaptive multipole algorithm is shown in Fig. 4. Here the operation count is the number of multiply-add operations required to compute Pq once, assuming that the entries in P and the multipole translation matrices have been precomputed. As the graph in Fig. 4 clearly demonstrates, the cost of computing Pq with the adaptive multipole algorithm increases linearly with the number of panels, and for the case where second-order expansions are used, is faster than direct computation with as few as 500 panels.

The effect of using the preconditioner to solve the two-sphere problem is shown in Fig. 5. As is clear from the figure, the number of iterations required is proportional to $\sqrt{N}$ without preconditioning, but curiously decreases slightly with N with preconditioning. Note

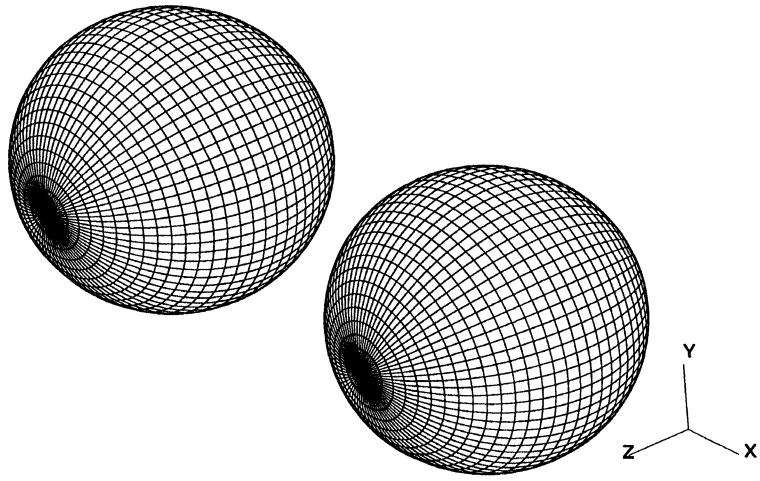

FIG. 3. *The two-sphere case, each discretized by* 2592 *panels. Here, a fictitious potential is applied. The sphere centers are three radii apart.*

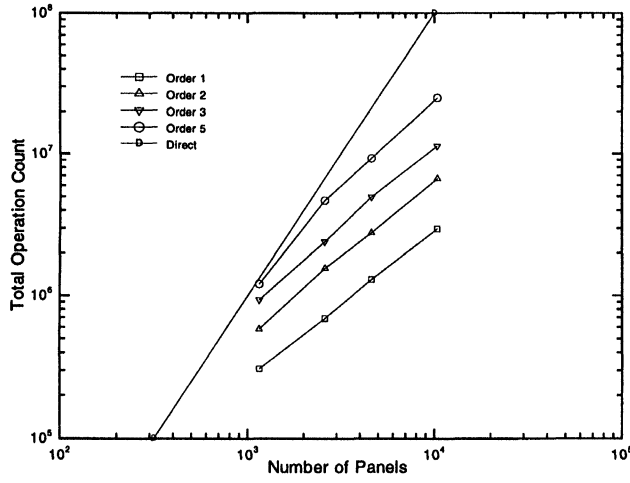

FIG. 4. *Operation counts for computing the iterates for the fictitious Dirichlet problem of two spheres, where the discretization is refined.* Direct *refers to the standard (order* (N^2)*) application of the matrix to a vector,* Order 1 *refers to the adaptive multipole algorithm with expansions to order* 1.

that the number of levels in the cube hierarchy directly affects what is used as a preconditioner. As Fig. 5 shows, the smaller the number of levels used, the larger the spatial extent of the preconditioner; the result is a reduction in the number of iterations required to achieve convergence.

6.2. Capacitance extraction. In this section we present results using the complete algorithm to solve for the capacitance matrix associated with a collection of m ideal conductors embedded in a uniform lossless dielectric medium. The capacitance matrix is defined as the $m \times m$ matrix that relates conductor potentials to the integral of conductor charge density. The jth column of the capacitance matrix can be computed by solving for the surface charge density on each of the conductors when the jth conductor is at unit potential, and all the other

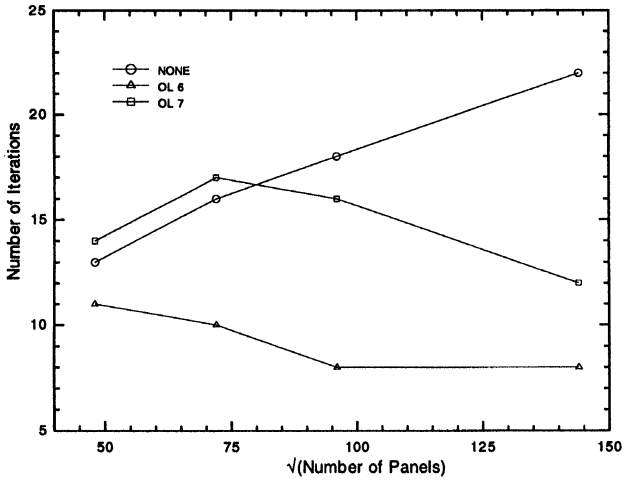

FIG. 5. *Iterations required for convergence to a tolerance of* 0.001 *in the solution of the fictitious Dirichlet problem of two spheres, as the discretization is refined.* None *refers to no preconditioning,* OL *refers to use of the overlapping-block preconditioner, and the associated number,* 6 *or* 7, *indicates the number of levels in the cube hierarchy.*

conductors are at zero potential. Note that all of the capacitance calculations are the result of using second-order multipole expansions and a GMRES relative convergence tolerance of 0.01.

The complete algorithm is nearly as accurate as the direct factorization method on complex problems, such as the 2×2 woven bus structure in Fig. 6. In Table 1, the capacitances computed using the two methods are compared using coarse, medium, and fine discretizations of the woven bus structure, also shown in Fig. 6. Note that even the coupling capacitance C_{12} between conductors one and two, which is forty times smaller than the self-capacitance C_{11}, is computed nearly as accurately with the complete algorithm as with direct factorization.

TABLE 1

Capacitance values (in pF) illustrating the accuracy of the PAMA *algorithm for the complicated geometry of Fig.* 6.

Method	Problem					
	Woven1 1584 panels		Woven2 2816 panels		Woven3 4400 panels	
	C_{11}	C_{12}	C_{11}	C_{12}	C_{11}	C_{12}
Direct	251.6	−6.353	253.2	−6.446	253.7	−6.467
PAMA	251.8	−6.246	253.3	−6.334	253.9	−6.377

On complex capacitance extraction problems, the computational cost of using the complete algorithm is roughly proportional to the product of the number of conductors, m, and the number of panels n. This is experimentally verified by computing the capacitances of the2×2 woven bus structure in Fig. 6, with progressively finer discretizations. In Fig. 7, the execution times required to compute these capacitances are plotted as a function of mn, and as the graph demonstrates, the execution time does grow nearly linearly.

To demonstrate the effectiveness of various aspects of the PAMA iterative algorithm on a range of problems, the execution times required to compute the capacitances of four different examples using four different methods are given in Table 2. The 2×2 woven bus example is described above, and the 5×5 woven bus example is the obvious extension. The via

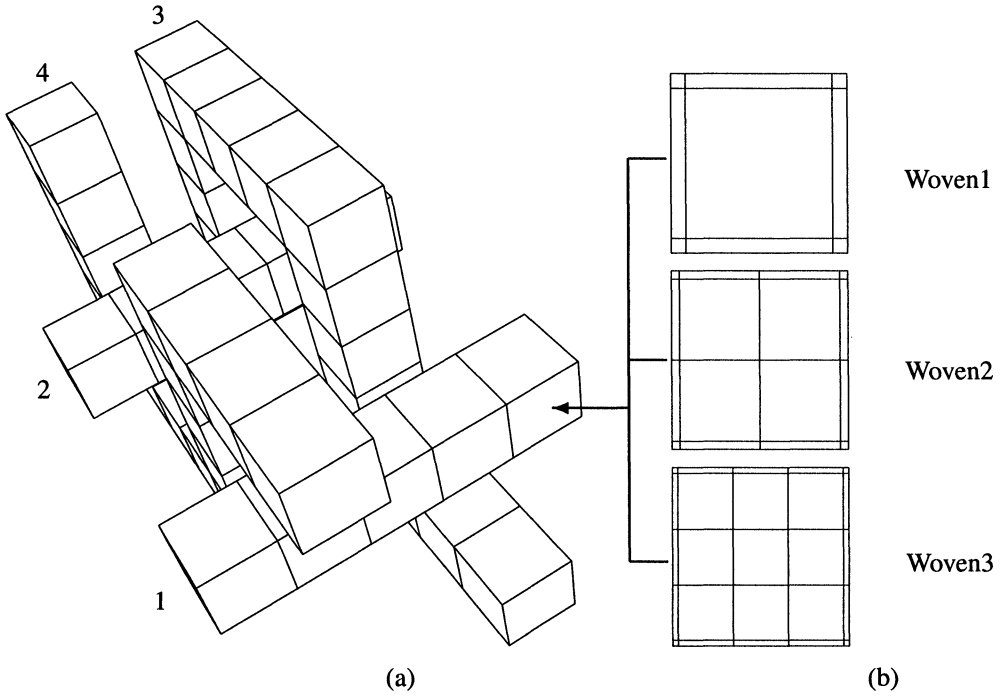

FIG. 6. *The* 2 × 2 *woven bus problem: bars have* 1*m* × 1*m cross sections. The three discretizations are obtained by replacing each square region in* (a) *with the corresponding set of panels in* (b).

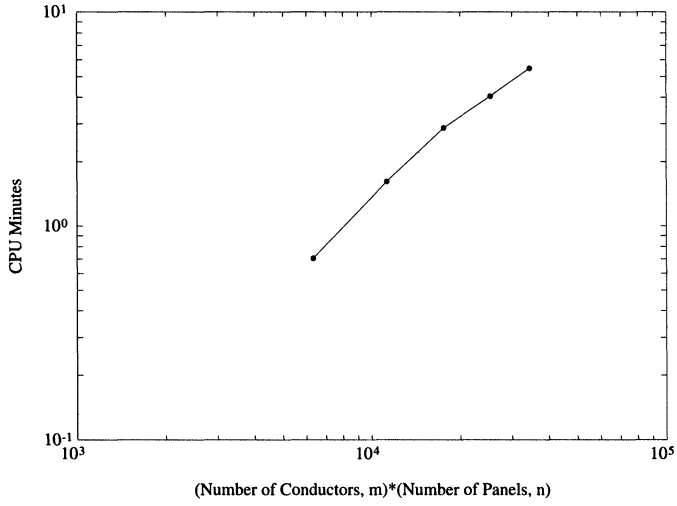

FIG. 7. *Execution time as a function of mn for the* PAMA *algorithm applied to solving progressively finer discretizations of the* 2 × 2 *woven bus problem.*

example, shown in Fig. 8, models a pair of connections between integrated circuit pins and a chip-carrier, and the diaphragm example, shown in Fig. 9, is a model for a microsensor [18].

From Table 2, it can be seen that using the AMA algorithm can improve execution time by a factor of two over using the multipole-accelerated (MA) algorithm alone, and combining the preconditioner with the AMA algorithm can reduce the execution time by as much as a factor

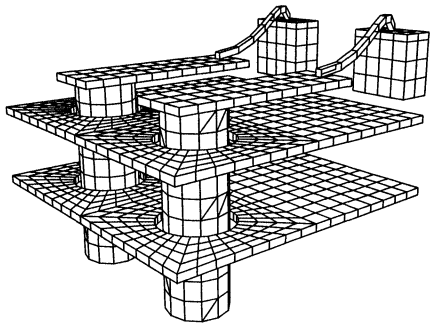

FIG. 8. *Two signal line vias passing through conducting planes.*

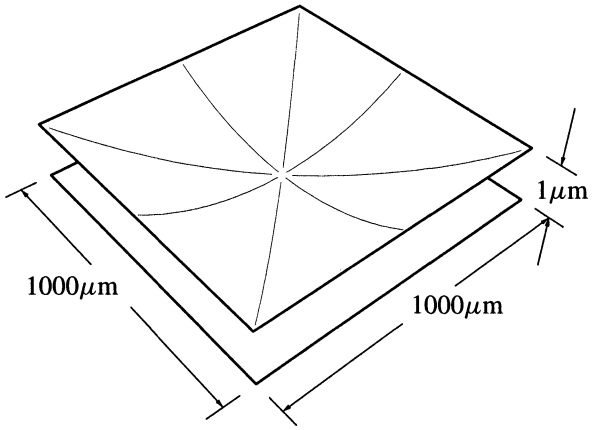

FIG. 9. *A schematic illustration of the diaphragm problem. The gap between the two plates is* 0.02 *μm at the center.*

TABLE 2

CPU times in minutes on an IBM RS6000/540 *for the nonadaptive* MA, AMA, *and* PAMA *algorithms. Times in parentheses are extrapolated.*

Method	2 × 2 woven bus 4400 panels	Via 6185 panels	Diaphragm 7488 panels	5 × 5 woven bus 9630 panels
Direct	185	(490)	(890)	(1920)
MA	6.0	11	8.7	42
AMA	3.3	4.7	5.9	23
PAMA	2.3	3.2	1.3	11

of seven. The improvement due to the adaptive algorithm is small because it is being compared to an MA algorithm that is already somewhat adaptive; empty cubes being ignored. Exploiting empty cubes is easy to implement, and makes a significant difference. For the largest problem, the 5 × 5 woven bus, more than 252,000 out of 262,000 cubes used to partition the problem domain are empty. A truly nonadaptive MA algorithm would therefore be *twenty-five* times slower than the MA algorithm used here for comparison.

The reduction in execution time afforded by the AMA algorithm over the nonadaptive MA stems from more efficiently computing the Pq product on each iteration of the GMRES algorithm, and using the preconditioner reduces execution time by reducing the number of

iterations required to achieve convergence. Because of various program overheads, comparing total execution times can hide the sometimes dramatic effect the preconditioner can have on GMRES convergence. To show the impact of the preconditioner more directly, the norm of the residual, $\|\overline{p} - Pq^k\|$, is plotted in Fig. 10 as a function of iteration for the various algorithms applied to solving the diaphragm problem. As is evident in the figure, the nonadaptive MA and AMA algorithms converge nearly identically, as expected, but the residuals computed with the PAMA algorithm decrease considerably faster. It is this rapid convergence that easily offsets the disadvantage that preconditioned iterates are slightly more expensive to compute than unpreconditioned iterates.

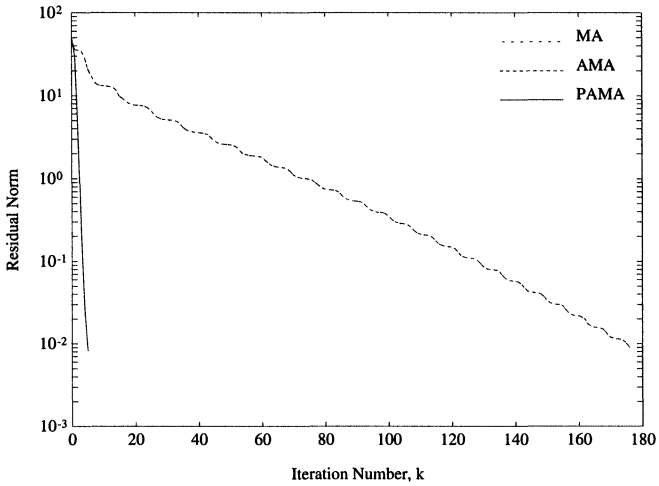

FIG. 10. *The* GMRES *residual norms for the linear system solution corresponding to the diaphragm problem (Fig.* 9*) with the top conductor at unit potential. As is evident here, the* PAMA *algorithm converges significantly more rapidly than the unpreconditioned* AMA *or* MA *algorithms.*

7. Conclusions. In this paper, a PAMA approach to solving first-kind surface integral equations with a $\frac{1}{r}$ kernel is described, and the method is shown to be effective for several engineering problems. A novel adaptive fast multipole algorithm is given and is proved to require order N computation and order N storage. Also, experimental evidence is given to demonstrate that in practice the combined algorithm is nearly order N. Note that the derivation and results are for a collocation scheme, the extension to a Galerkin scheme is straightforward though somewhat more cumbersome to implement.

Acknowlegments. The authors would like to thank Stephen Vavasis for his suggestions about approaches to preconditioning, Don Baltus for his suggestions about using graphs to represent the multipole algorithm, and Songmin Kim for his help in linking the program with the solid modeler PATRAN. The authors would also like to thank David Ling and Albert Ruehli of the I.B.M. T. J. Watson Research Center for their helpful suggestions about capacitance calculations and Dick Yue for his help in understanding potential flow problems.

REFERENCES

[1] V. ROHKLIN, *Rapid solution of integral equation of classical potential theory*, J. Comput. Phys., 60 (1985), pp. 187–207.

[2] A. GREENBAUM, L. GREENGARD, AND G. B. MCFADDEN, *Laplace's Equation and the Dirichlet–Neumann Map in Multiply Connected Domains*, Tech. rep., Courant Institute, New York University, NY, March 1991.

[3] A. E. RUEHLI AND P. A. BRENNAN, *Efficient capacitance calculations for three-dimensional multiconductor systems*, IEEE Trans. Microwave Theory and Techniques, 21 (1973), pp. 76–82.

[4] S. D. SENTURIA, R. M. HARRIS, B. P. JOHNSON, S. KIM, K. NABORS, M. A. SHULMAN, and J. K. White, *A computer-aided design system for microelectromechanical systems* (memcad), IEEE J. Microelectromechanical Systems, 1 (1992), pp. 3–13.

[5] J. N. NEWMAN, *Distributions of sources and normal dipoles over a quadrilateral panel*, J. Engrg. Math., 20 (1986), pp. 113–126.

[6] R. KRESS, *Linear Integral Equations*, Springer-Verlag, Berlin, 1989.

[7] Y. SAAD AND M. H. SCHULTZ, GMRES: *A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[8] L. GREENGARD AND V. ROKHLIN, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[9] L. GREENGARD, *The Rapid Evaluation of Potential Fields in Particle Systems*, M.I.T. Press, Cambridge, MA, 1988.

[10] E. W. HOBSON, *The Theory of Spherical and Ellipsoidal Harmonics*, Chelsea, New York, 1955.

[11] J. CARRIER, L. GREENGARD, AND V. ROKHLIN, *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 669–686.

[12] J. J. SAKURAI, *Modern Quantum Mechanics*, Addison-Wesley, Reading, MA, 1985.

[13] K. NABORS AND J. WHITE, *Fastcap: A multipole accelerated* 3-D *capacitance extraction program*, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, 10 (1991), pp. 1447–1459.

[14] K. NABORS, S. KIM, AND J. WHITE, *Fast capacitance extraction of general three-dimensional structures*, IEEE Trans. Microwave Theory and Techniques, 40 (1992), pp. 1496–1506.

[15] N. DEO, *Graph Theory with Applications to Engineering and Computer Science*. Prentice-Hall, Englewood Cliffs, NJ, 1974.

[16] S. A. VAVASIS, *Preconditioning for boundary integral equations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 905–925.

[17] K. NABORS, S. KIM, J. WHITE, AND S. D. SENTURIA, *An adaptive multipole algorithm for* 3-D *capacitance calculation*, in Proc. Internat. Conf. Computer Design, Cambridge, MA, October 1991.

[18] B. JOHNSON, S. KIM, S. D. SENTURIA, AND J. WHITE, MEMCAD *capacitance calculations for mechanically deformed square diaphragm and beam microstructures*, in Proc. Transducers 91, San Francisco, CA, June 1991.