SIAM J. SCI. COMPUT.
Vol. 23, No. 3, pp. 741–760

A NEW FAST-MULTIPOLE ACCELERATED POISSON SOLVER IN TWO DIMENSIONS*

FRANK ETHRIDGE† AND LESLIE GREENGARD‡

Abstract. We present an adaptive fast multipole method for solving the Poisson equation in two dimensions. The algorithm is direct, assumes that the source distribution is discretized using an adaptive quad-tree, and allows for Dirichlet, Neumann, periodic, and free-space conditions to be imposed on the boundary of a square. The amount of work *per grid point* is comparable to that of classical fast solvers, even for highly nonuniform grids.

Key words. fast multipole method, Poisson equation, adaptive refinement, fast Poisson solver

AMS subject classifications. 31A10, 35J05, 65R10, 78A30

PII. S1064827500369967

1. Introduction.

A variety of problems in scientific computing involve the solution of the Poisson equation

$$\Delta\psi = f, \tag{1}$$

subject to appropriate radiation or boundary conditions. In simple geometries (circular or rectangular domains) with regular grids, there are well-known fast direct solvers [6, 7] which typically rely on the fast Fourier transform (FFT) and are well suited to the task. When either restriction is relaxed, however, these methods no longer apply. Since practical problems tend to involve complex geometries, highly inhomogeneous source distributions f, or both, there has been a lot of effort directed at developing alternative approaches. Most currently available solvers rely on iterative techniques using multigrid, domain decomposition, or some other preconditioning strategy [5, 9, 21]. Unfortunately, while such multilevel strategies can achieve nearly optimal efficiency in theory, they require an appropriate hierarchy of coarse grids which is not provided in practice. Although there has been significant progress in this direction [1, 2, 10, 11, 20, 23], the available solvers compare unfavorably with the fast direct solvers in terms of work per grid point.

In this paper, we describe an integral equation method for solving the Poisson equation in two dimensions which is direct, high order accurate, insensitive to the degree of adaptive mesh refinement, and accelerated by the fast multipole method (FMM) [16, 17, 26]. It is competitive with standard fast solvers in terms of *work per grid point*. This is a rather stringent test, since we compare the time for a classical, FFT-based solver using N mesh points with our adaptive, FMM-based solver using the same number of points, ignoring the fact that the latter solver uses grids which are highly inhomogeneous. We allow for the imposition of various combinations of free-space, periodic, Dirichlet, and Neumann conditions on the boundary of a square.

Earlier work on FMM-based integral equation schemes in two dimensions includes [15, 22, 29]. The paper [22] describes a fast Poisson solver for complex geometries,

*Received by the editors April 3, 2000; accepted for publication (in revised form) December 8, 2000; published electronically August 15, 2001.
http://www.siam.org/journals/sisc/23-3/36996.html
†Department of Computer Science, Yale University, New Haven, CT 06520 (ethridge@cs.yale.edu).
‡Courant Institute of Mathematical Sciences, New York University, New York, NY 10012 (greengard@cims.nyu.edu). The work of this author was supported by the Applied Mathematical Sciences Program of the U.S. Department of Energy under contract DEFGO288ER25053.

where the boundary can be arbitrarily shaped and multiply-connected, but where the right-hand side is specified on a uniform underlying mesh. The other two papers discuss the inversion of (1) in free space by evaluation of the analytic solution

$$\psi(\mathbf{x}) = \frac{1}{2\pi} \int_D f(\mathbf{y}) \log(|\mathbf{x} - \mathbf{y}|)\, d\mathbf{y}. \tag{2}$$

The algorithms of both [15] and [29] are highly adaptive, with the former relying on a quad-tree and the latter relying on an unstructured triangulation. Neither, however, goes beyond the free-space problem.

The present approach is similar to that outlined in [15] but differs in several respects.

1. In the method of [15], one solves local Poisson problems with spectral methods on each leaf node of a quad-tree data structure and then patches the solutions together using the FMM in a domain decomposition approach. Here, we apply the FMM directly to the volume integral, using high order quadratures.
2. We incorporate a new version of the FMM described in [19], which is based on diagonal forms for translation operators (see section 3.8).
3. We incorporate the method of images to solve a variety of boundary value problems on a square (with adaptive refinement).
4. For fourth and sixth order accurate discretizations, we use locally uniform meshes, compatible with adaptive mesh refinement (AMR) data structures [3]. For eighth order accuracy, we follow [13, 15, 24] and rely on local spectral meshes.

The paper is organized as follows. In section 2, we outline the relevant potential theory, with particular emphasis on the method of images. In section 3, we describe the fast multipole algorithm itself, and in section 4, we present several numerical examples. Finally, it should be noted that our algorithm shares a number of features with the recently developed scheme of [12] for solving the pseudodifferential equation

$$(-\Delta)^{1/2}\psi = \omega \tag{3}$$

in the plane via the integral representation

$$\psi(\mathbf{x}) = \int_{\mathbf{R}^2} \frac{\omega(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|}\, d\mathbf{y}.$$

2. Potential theory. To complete a description of a well-posed problem, we must obviously add to the Poisson equation (1) a specification of boundary conditions on the unit square D. We allow the free-space conditions defined by (2), periodic boundary conditions, Dirichlet conditions, and Neumann conditions. We can also handle mixed conditions, but assume that the transition from one type to another (Dirichlet–Neumann, etc.) occurs only at corners. The solution to all these problems can be constructed analytically using the method of images.

For periodic boundary conditions, one simply imagines the entire plane to be tiled with copies of the source distribution contained in the unit cell D. (How to compute the influence of each of these images efficiently is discussed in the next section.) For other boundary conditions, the construction is a bit more subtle. Let us consider, for

example, the boundary value problem

$$\begin{aligned} \Delta\psi &= f \quad \text{in } D, \\ \psi &= g_L \quad \text{on } \Gamma_L, \\ \psi &= g_R \quad \text{on } \Gamma_R, \\ \partial\psi/\partial n &= g_T \quad \text{on } \Gamma_T, \\ \partial\psi/\partial n &= g_B \quad \text{on } \Gamma_B, \end{aligned} \tag{4}$$

where Γ_L denotes the "left" boundary ($x = -0.5, -0.5 \le y \le 0.5$), Γ_R denotes the "right" boundary ($x = 0.5, -0.5 \le y \le 0.5$), Γ_T denotes the "top" boundary ($-0.5 \le x \le 0.5, y = 0.5$), and Γ_B denotes the "bottom" boundary ($-0.5 \le x \le 0.5, y = -0.5$).

This problem can be conveniently broken up into two parts. First, we can solve the Poisson equation:

$$\begin{aligned} \Delta\psi_1 &= f \quad \text{in } D, \\ \psi_1 &= 0 \quad \text{on } \Gamma_L, \\ \psi_1 &= 0 \quad \text{on } \Gamma_R, \\ \partial\psi_1/\partial n &= 0 \quad \text{on } \Gamma_T, \\ \partial\psi_1/\partial n &= 0 \quad \text{on } \Gamma_B. \end{aligned} \tag{5}$$

Then we can solve the Laplace equation with inhomogeneous boundary conditions:

$$\begin{aligned} \Delta\psi_2 &= 0 \quad \text{in } D, \\ \psi_2 &= g_L \quad \text{on } \Gamma_L, \\ \psi_2 &= g_R \quad \text{on } \Gamma_R, \\ \partial\psi_2/\partial n &= g_T \quad \text{on } \Gamma_T, \\ \partial\psi_2/\partial n &= g_B \quad \text{on } \Gamma_B. \end{aligned} \tag{6}$$

Clearly, $\psi = \psi_1 + \psi_2$.

To solve (5), suppose that we tile the plane with the pattern of images depicted in Figure 1. The shaded box is the computational domain containing the source distribution f. f_T denotes the even reflection of the function f across the top boundary Γ_T, $-f_R$ denotes the odd reflection of the function f across the right boundary Γ_R, and $-f_{RT}$ denotes the even reflection of the function $-f_R$ across the line $y = +\frac{1}{2}$. It is easy to verify that the vertical lines $x = \pm\frac{1}{2}$ are lines of odd symmetry and that the horizontal lines $y = \pm\frac{1}{2}$ are lines of even symmetry. Thus, the desired homogeneous boundary conditions are enforced if we account for the field due to all images. This task is simplified by the observation that the 2×2 supercell outlined with dashes in Figure 1 tiles the plane periodically.

To handle the inhomogeneous boundary conditions in (6), we recall the following classical results from potential theory [18, 30].

LEMMA 2.1. *Let $u(x, y)$ satisfy the Laplace equation $\Delta u = 0$ in the half-space $y > 0$ with Dirichlet boundary conditions $u(x, 0) = f(x)$. Then $u(x, y)$ is given by the double layer potential*

$$u(x,y) = 2\int_{-\infty}^{\infty} \frac{\partial G}{\partial y}(x-\xi, y)\, f(\xi)\, d\xi = \frac{1}{\pi}\int_{-\infty}^{\infty} \frac{y}{(x-\xi)^2 + y^2}\, f(\xi)\, d\xi.$$

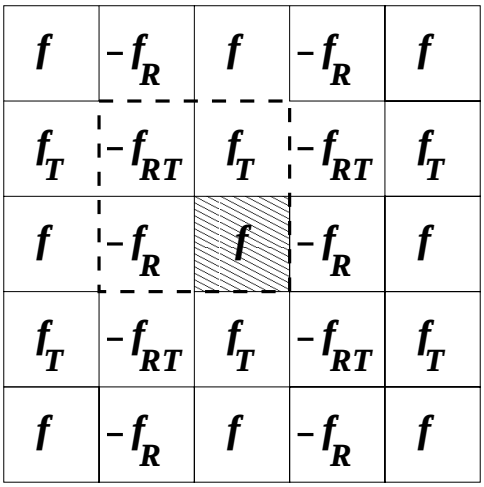

FIG. 1. *A source distribution tiling the plane which solves the Poisson equation with the homogeneous boundary conditions described by the system* (5). *The shaded box represents the computational domain itself.* f_T *denotes the even reflection of the function* f *across the top boundary* Γ_T, $-f_R$ *denotes the odd reflection of the function* f *across the right boundary* Γ_R, *and* $-f_{RT}$ *denotes the even reflection of the function* $-f_R$ *across the line* $y = +\frac{1}{2}$. *The* 2×2 *"supercell" with the dashed outline can be seen to tile the plane periodically.*

The solution satisfying Neumann boundary conditions $\frac{\partial u}{\partial n}(x,0) = g(x)$ *is given by the single layer potential*

$$u(x,y) = 2\int_{-\infty}^{\infty} G(x-\xi,y)\,g(\xi)\,d\xi = \frac{1}{\pi}\int_{-\infty}^{\infty} \ln\sqrt{(x-\xi)^2+y^2}\,g(\xi)\,d\xi.$$

Consider now the system of layer potentials depicted in Figure 2. We leave it to the reader to verify that, from the preceding lemma and symmetry considerations, the boundary conditions of (6) are satisfied. As with the tiling of source distributions, the task of accounting for the field due to all images is simplified by the observation that the layer potentials on boundary segments outlined with dots in Figure 2 tile the plane periodically. The evaluation of layer potentials is discussed in section 3.7.

3. Data structures and the FMM. We assume that the source distribution f in (2) is supported inside the unit square D, centered at the origin, on which is superimposed a hierarchy of refinements (a quad-tree). Grid level 0 is defined to be D itself, and grid level $l+1$ is obtained recursively by subdividing each square at level l into four equal parts. Using standard terminology, if d is a fixed square at level l, the four squares at level $l+1$ obtained by its subdivision will be referred to as its children. In order to allow for adaptivity, we do not use the same number of levels in all regions of D. We do, however, assume that the quad-tree satisfies one fairly standard restriction, namely, that two leaf nodes which share a boundary point must be no more than one refinement level apart (Figure 3).

3.1. The volume integral. We restrict our attention, for the moment, to the free-space problem. Extended volume integrals such as the ones depicted in Figure 1 will be discussed in section 3.6.

The leaf nodes on which the source distribution is given will be denoted by D_i.

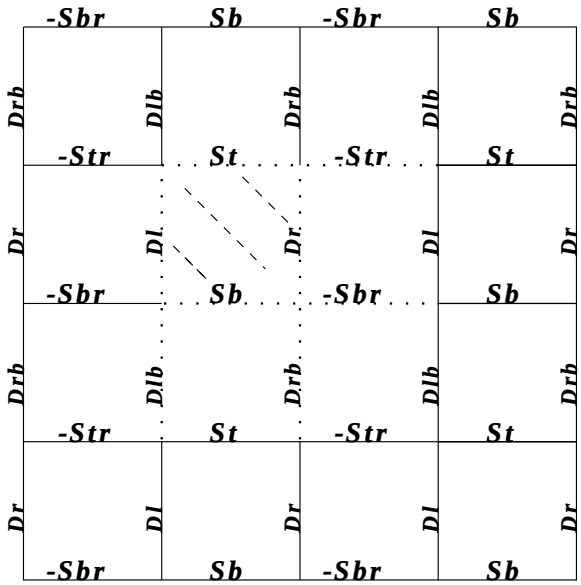

FIG. 2. *A tiling of the plane with layer potentials to solve* (6). *The computational domain is indicated with diagonal dashes. Dl denotes a double layer potential with density* g_L, *and Dr denotes a double layer potential with density* g_R. *Their even reflections across the bottom boundary* Γ_B *are Dlb and Drb, respectively. St denotes a single layer potential with density* g_T, *and Sb denotes a single layer potential with density* g_B. *Their odd reflections across the right boundary* Γ_R *are* $-Str$ *and* $-Sbr$, *respectively. Symmetry considerations show that all four boundary conditions are satisfied. Note that the layer potentials on boundary segments outlined with dots tile the plane periodically.*

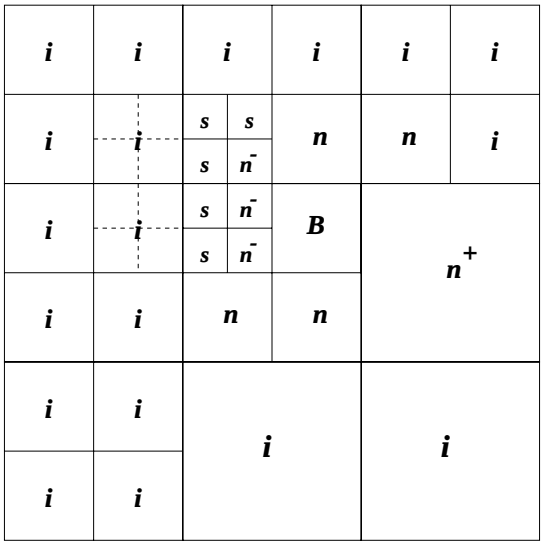

FIG. 3. *For the childless node* B, *colleagues are labeled* n, *coarse neighbors are labeled* n^+, *and fine neighbors are labeled* n^-. *The interaction list for* B *consists of the boxes marked* i. *The boxes marked by* s *are children of* B*'s colleagues which are separated from* B, *so they are not fine neighbors. They constitute the* s-list *for* B *(see Definition* 3.1*).*

Thus, $D = \cup_{i=1}^{M} D_i$ and we rewrite (2) in the form

$$\psi(\mathbf{x}) = \sum_{i=1}^{M} \frac{1}{2\pi} \int_{D_i} f(\mathbf{y}) \log(|\mathbf{x} - \mathbf{y}|)\, d\mathbf{y}. \tag{7}$$

DEFINITION 3.1. *The* colleagues *of a square B are squares at the same refinement level which share a boundary point with B. (B is considered to be a colleague of itself.) The* coarse neighbors *of B are leaf nodes at the level of B's parent which share a boundary point with B. The* fine neighbors *of B are leaf nodes one level finer than B which share a boundary point with B. Together, the union of the colleagues, coarse neighbors, and fine neighbors of B will be referred to as B's* neighbors. *The* s-list *of a box B consists of those children of B's colleagues which are not fine neighbors of B.*

The interaction region *for B consists of the area covered by the neighbors of B's parent, excluding the area covered by B's colleagues and coarse neighbors. The* interaction list *for B consists of those squares in the interaction region which are at the same refinement level, as well as leaf nodes in the interaction region which are at coarser levels. When the distinction is important, the squares at the same refinement level will be referred to as the* standard interaction list*, while the squares at coarser levels will be referred to as the* coarse interaction list.

In our FMM, following [8, 16, 17], terms in the convolution integral (7) from neighbor leaf nodes are computed directly. More distant interactions are accounted for on coarser levels, through the use of a hierarchy of far-field and local multipole expansions. We consider the local interactions first.

3.2. Local interactions. For fourth and sixth order accuracy, we assume that we are given f on a cell-centered $k \times k$ grid for each leaf node B, with $k = 4$ or 6, respectively. We can, therefore, take these k^2 data points and construct a kth order polynomial approximation to f of the form

$$f_B(x, y) \approx \sum_{j=1}^{N_k} c_B(j)\, b_j(x - x_B, y - y_B),$$

where $N_k = \frac{k(k+1)}{2}$ is the number of basis functions needed for kth order accuracy and where (x_B, y_B) denotes the center of B. The basis functions $b_1(x, y), \ldots, b_{N_k}(x, y)$ are given by

$$\{x^i y^j |\ i, j \geq 0, i + j \leq k - 1\}.$$

If we let $\vec{f_B} \in \mathbf{R}^{k^2}$ denote the given function values (in standard ordering), then the calculation of the coefficient vector $\vec{c}_B$ is clearly overdetermined. We obtain it through a least squares fit based on the singular value decomposition. The pseudoinverse matrix $\mathcal{P} \in \mathbf{R}^{N_k \times k^2}$, such that

$$\vec{c}_B = \mathcal{P}\, \vec{f_B},$$

can be precomputed and stored.

Remark 3.1. For eighth order accuracy, we assume that f is given on a scaled 8×8 classical tensor product Chebyshev grid [7] and use as basis functions

$$\{T_i(x) T_j(y) |\ i, j \geq 0, i + j \leq k - 1\},$$

where $T_i(x)$ denotes the Chebyshev polynomial of degree i. The coefficients of the Chebyshev expansion can be computed efficiently using the fast cosine transform.

Consider now a target point Q, which lies in a neighbor of B. The field induced at Q by f_B is approximated by

$$\psi_B(Q) = \sum_{n=1}^{N_k} c_B(n) f(Q, n), \tag{8}$$

where

$$f(Q, n) = \frac{1}{2\pi} \int_B b_n(x - x_B, y - y_B) \log |Q - (x, y)| \, dxdy. \tag{9}$$

Since the target points Q are regularly spaced in each neighboring square, we can precompute the weights (9) for each of the k^2 possible locations at each of 9 possible colleagues, 12 possible fine neighbors, and 12 possible coarse neighbors. To be more precise, we can precompute the weights assuming that B is the unit square $[-0.5, 0.5]^2$ because of the following straightforward lemma.

LEMMA 3.2. *Let B be a leaf node at level l and let Q denote a target point in one of B's neighbors. Let Q^* denote the scaled target point for the unit cell centered at the origin*

$$Q^* = 2^{l-1} \cdot (Q - (x_B, y_B)),$$

let

$$f^*(Q^*, n) = \frac{1}{2\pi} \int_{-1/2}^{1/2} \int_{-1/2}^{1/2} b_n(x, y) \log |Q^* - (x, y)| \, dxdy, \tag{10}$$

and let

$$\bar{f}(n, l) = \frac{1}{2\pi} \left(\frac{1}{2^{l-1}} \right)^{d+2} \int_{-1/2}^{1/2} \int_{-1/2}^{1/2} b_n(x, y) \log \left(\frac{1}{2^{l-1}} \right) \, dxdy. \tag{11}$$

Then the integral $f(Q, n)$ defined in (9) is given by

$$f(Q, n) = \left(\frac{1}{2^{l-1}} \right)^{d+2} f^*(Q^*, n) + \bar{f}(n, l),$$

where d is the degree of the polynomial basis function b_n.

Thus, we need only obtain weights for a box of unit area. Elementary counting arguments show that the storage required for this precomputation is

$$\begin{aligned} k \times k \cdot N_k \cdot 9 \text{ real numbers} \quad & \text{for colleagues,} \\ k \times k \cdot N_k \cdot 12 \text{ real numbers} \quad & \text{for fine neighbors,} \\ k \times k \cdot N_k \cdot 12 \text{ real numbers} \quad & \text{for coarse neighbors,} \end{aligned}$$

for a total of approximately $17 \times k^4$ real numbers.

3.3. Far-field interactions. We turn now to the calculation of far-field interactions, which are computed by means of multipole expansions. We refer the reader to [14, 16] for more detailed discussions of potential theory. Our starting point is the usual multipole expansion for a charge distribution, which we state formally as a theorem.

THEOREM 3.3 (multipole expansion). *Let $\rho(\mathbf{y})$ be a charge distribution contained within a square D_i with center C and let $\Phi(\mathbf{x})$ denote the induced field at a point $\mathbf{x}$ in the interaction list of D_i:*

$$\Phi(\mathbf{x}) = \frac{1}{2\pi} \int_{D_i} \log|\mathbf{x}-\mathbf{y}|\rho(\mathbf{y})\, d\mathbf{y}.$$

Then $\Phi(\mathbf{x})$ can be described by the multipole expansion

$$\Phi(\mathbf{x}) = \alpha_0 \log|\mathbf{x}-C| + \Re\left(\sum_{k=1}^{\infty} \frac{\alpha_k}{(x_1+ix_2-C)^k}\right), \tag{12}$$

where C is viewed as a point in the complex plane, $\mathbf{x} = (x_1, x_2)$, and $\Re(w)$ denotes the real part of the complex quantity w. The coefficients α_k are given by

$$\begin{aligned} \alpha_0 &= \frac{1}{2\pi} \int_{D_i} \rho(y_1, y_2) dy_1\, dy_2, \\ \alpha_k &= -\frac{1}{2\pi} \int_{D_i} \frac{(y_1+iy_2-C)^k \rho(y_1,y_2)}{k} dy_1\, dy_2. \end{aligned} \tag{13}$$

In the hierarchical framework of the FMM, an upper bound for the error in truncating the expansion after n terms is given by

$$\left(\frac{1}{2}\right)^n \frac{1}{2\pi} \int_{D_i} |\rho(y_1,y_2)|\, dy_1\, dy_2. \tag{14}$$

THEOREM 3.4 (local expansion). *Let $\rho(\mathbf{y})$ be a charge distribution contained outside the neighbors of a square D_i with center C and let $\Psi(\mathbf{x})$ denote the induced field at $\mathbf{x} \in D_i$. Then $\Psi(\mathbf{x})$ can be described by a local expansion*

$$\Psi(\mathbf{x}) = \Re\left(\sum_{l=0}^{\infty} \beta_l (x_1+ix_2-C)^l\right), \tag{15}$$

where C is viewed as a point in the complex plane and $\mathbf{x} = (x_1, x_2)$.

The FMM relies on the ability to manipulate multipole and local expansions for every box in the tree hierarchy. We omit the technical details and refer the reader to the original papers [8, 14, 16, 17].

DEFINITION 3.5. *We denote by $S_{l,k}$ the kth square at refinement level l.*

We denote by $\Phi_{l,k}$ the multipole expansion describing the far field due to the source distribution supported inside $S_{l,k}$.

We denote by $\Psi_{l,k}$ the local expansion describing the field due to the source distribution outside the neighbors of $S_{l,k}$.

We denote by $\tilde{\Psi}_{l,k}$ the local expansion describing the field due to the source distribution outside the neighbors of the parent *of $S_{l,k}$.*

Remark 3.2. Let $S_{l,k}$ be a square in the quad-tree hierarchy and let $S_{l',k'}$ be a square in its interaction list. Then there is a linear operator $\mathcal{T}_{MM}$ for which

$$\Phi_{l,k} = \mathcal{T}_{MM}[\Phi(C_1), \Phi(C_2), \Phi(C_3), \Phi(C_4)], \tag{16}$$

where $\Phi(C_j)$ denotes the multipole expansion for the jth child of $S_{l,k}$. In other words, we can merge the expansions for four children into a single expansion for the parent. Similarly, there is a linear operator $\mathcal{T}_{LL}$ for which

$$[\tilde{\Psi}(C_1), \tilde{\Psi}(C_2), \tilde{\Psi}(C_3), \tilde{\Psi}(C_4)] = \mathcal{T}_{LL}\, \Psi_{l,k}, \tag{17}$$

where C_j denotes the jth child of $S_{l,k}$. In other words, we can shift the local expansion Ψ for a box to the corresponding expansion $\tilde{\Psi}$ for each of its children. Finally, there is a linear operator $\mathcal{T}_{ML}$ for which the field in $S_{l,k}$ due to the source distribution in $S_{l',k'}$ is described by $\Psi = T_{ML}\Phi_{l',k'}$. It is easy to verify that

$$\Psi_{l,k} = \tilde{\Psi}_{l,k} + \sum_{i \in \mathcal{IL}} T_{ML}\Phi_i, \tag{18}$$

where $\mathcal{IL}$ denotes the interaction list for square $S_{l,k}$.

Remark 3.3. One slight complication in the adaptive algorithm concerns the interaction between boxes of different sizes. Referring to Figure 3, we need to account for the influence of a childless square B on each box marked s and vice versa. (This interaction doesn't arise if B undergoes further refinement.) For the box marked s, its multipole expansion is rapidly convergent at each of the k^2 target points in B. Thus, its influence can be computed by direct evaluation of the truncated series. For the reverse, however, note that B's multipole expansion is not so rapidly convergent. In this case, we can map directly from the polynomial coefficients $\vec{c}_B$ of B to the local expansion in s. A more precise statement than (18) is

$$\Psi_{l,k} = \tilde{\Psi}_{l,k} + \sum_{i \in SIL} T_{ML}\Phi_i, + \sum_{i \in CIL} L_{direct}(\vec{c}_i), \tag{19}$$

where SIL denotes the standard interaction list and CIL denotes the coarse interaction list. The operator L_{direct}, which maps the coefficients of the polynomial approximation of the density in the coarse box onto the p coefficients of the local expansion can be precomputed and stored.

The bulk of the work in the FMM consists of applying the operators $\mathcal{T}_{MM}, \mathcal{T}_{LL}$, and especially $\mathcal{T}_{ML}$ in a systematic fashion. Unfortunately, these operators are dense. Using multipole and local expansions truncated after p terms, the naive cost of application is proportional to p^2. Recent improvements in the FMM have reduced this cost in both two and three space dimensions [17, 19]. A brief discussion of the technical ideas is presented in section 3.8.

3.4. The FMM algorithm.

Initialization

Comment [We assume we are given a square domain $D = S_{0,0}$, on which is superimposed an adaptive hierarchical quad-tree structure. We let M be the number of leaf nodes and denote them by $D_i,\ i = 1, \ldots, M$. The number of grid points is, therefore, $N = 16M$. We let p denote the order of the multipole expansion ($p \approx \log_2 \epsilon$, where ϵ is the desired accuracy). We let l_{max} denote the maximum refinement level.]

Step I: Multipole sweep

Upward pass

for $l = l_{max}, \ldots, 0$
  for all boxes j on level l
    if j is childless **then**
      form the multipole expansion $\Phi_{l,j}$ from (12)
    else
      form the multipole expansion $\Phi_{l,j}$ by merging the expansions of its children using the operator $\mathcal{T}_{MM}$ (see (16))
  end
end

Downward pass

Initialize the local expansion $\Psi_{0,0} = 0$.
for $l = 1, \ldots, l_{max}$
  for all squares j on level l
    Compute $\tilde{\Psi}_{l,j}$ by shifting its parent's Ψ expansion using the operator $\mathcal{T}_{LL}$
    Compute $\Psi_{l,j}$ by adding in the contributions from all squares in j's interaction list according to (19).
    if j is childless **then**
      for all boxes k in the *s-list* of j:
        evaluate the multipole expansion Φ_k at each target in square j.
      end
      Evaluate the local expansion $\Psi_{l,j}$ at each target in square j.
    endif
  end
end

Cost [The upward pass requires approximately Mp^2 work, where M is the number of leaf nodes. The downward pass requires approximately $3Mp^2$ work using plane-wave expansions (see section 3.8 below).]

Step II: Local interactions

Comment [At this point, for each leaf node D_i, we have computed the influence of the source distribution f over all leaf nodes D_j outside the neighbors of D_i.]

do $i = 1, \ldots, M$
  For each target point in D_i, evaluate the influence of each neighbor according to (8) using the precomputed tables of coefficients (10).
end

Cost [The maximum number of neighbors a square can have is thirteen (twelve fine neighbors and itself). Thus the local work is bounded by is $13 \cdot \frac{k(k+1)}{2} \cdot N$ operations.]

3.5. Periodic boundary conditions. The inversion formula (2) and the fast algorithm described above assume that the right-hand side f is supported within a unit square. When imposing periodic boundary conditions, as mentioned in section 2, one can simply assume that the entire plane is tiled with copies of f centered at the lattice points $\{(i,j)|i,j \in \mathbf{Z}\}$. In order to account for the influence of these images, we follow the approach introduced in [16], the essence of which can already be found in

Lord Rayleigh's classic paper [25]. The main thing to notice is that, after the upward pass in the FMM, we have a net multipole expansion describing the far field due to the entire source distribution f contained in the unit cell centered at the origin:

$$\phi(\mathbf{x}) = \Re\left(\sum_{n=1}^{p} \frac{\alpha_n}{z^n}\right). \tag{20}$$

(There is no logarithmic term since we assume that the source distribution has no net charge.) This is then the expansion for each of the periodic images of the box with respect to its own center. All of these images, except for the nearest neighbors centered at {(-1,-1), (-1,0), (-1,1), (0,-1), (0,1), (1,-1), (1,0), (1,1)}, are well separated from the computational domain itself. Thus, the fields they induce inside the computational domain are accurately representable by a p-term local expansion where, as before, p is the number of terms needed to achieve a relative precision ϵ. This local representation can be written as

$$\Psi_{0,0}(w) = \Re\left(\sum_{n=0}^{p} \beta_n w^n\right). \tag{21}$$

It remains only to obtain the operator mapping the coefficients $\{\alpha_n\}$ to the coefficients $\{\beta_n\}$. We refer the reader to [4, 16, 25] for a discussion of this operator, which is based on the precomputation of certain lattice sums. The reason we denote the local expansion in (21) by $\Psi_{0,0}$ is for consistency of notation with the FMM described above; the *downward pass* is modified in the initialization step. In the remainder of the downward pass and in Step II, only two changes are required; the interaction list and the local computations must be adjusted for boxes near the boundary to account for periodic images. This involves no significant increase in the amount of work.

3.6. Other homogeneous boundary conditions. As noted in section 2, problems with homogeneous Dirichlet and Neumann conditions can be solved using the method of images. Since there is a 2×2 "supercell" which tiles the plane periodically, it is straightforward to embed such problems in a periodic version of the FMM. Done naively, this would entail a fourfold increase in CPU time and storage. Careful implementation considerations allow one to recover this overhead, but the details are tedious and will be omitted.

3.7. Inhomogeneous boundary conditions. In order to impose inhomogeneous boundary conditions using potential theory, we need to consider arrangements of single and double layer potentials such as the one depicted in Figure 2. These can be viewed as singular charge distributions and can be handled by the same FMM as above, with three modifications. First, the far field due to a box B with a single layer density σ and a double layer density μ along its boundary Γ is given by

$$\phi(\mathbf{x}) = \Re\left(\alpha_0 \log|x_1 + ix_2 - C| + \sum_{k=1}^{\infty} \frac{\alpha_k}{(x_1 + ix_2 - C)^k}\right), \tag{22}$$

where

$$\alpha_0 = -\frac{1}{2\pi}\int_\Gamma \sigma(s)ds \tag{23}$$

and

$$\alpha_k = -\frac{1}{2\pi}\int_\Gamma \frac{(y_1(s) + iy_2(s) - C)^k \sigma(s)}{k} + (y_1(s) + iy_2(s) - C)^{k-1}\mu(s)\, ds. \tag{24}$$

Here, $(y_1(s), y_2(s))$ is an arclength parametrization of Γ. Second, contributions to the local field from a leaf node containing layer potentials are precomputed as in section 3.2. Finally, the interaction list and the local computations must be adjusted for boxes near the boundary.

3.8. Fast translation operators. Consider a box B centered at X_B, containing sources $\{z_1, \ldots, z_M\}$ with source strengths $\{q_1, \ldots, q_M\}$ and a target box D centered at X_D in its interaction list. We assume for the moment that $\Re(X_D) > \Re(X_B)$. In the original FMM, the field outside B is represented as

$$\phi(\mathbf{x}) = \Re\left(\alpha_0 \log(x_1 + ix_2 - X_B) + \sum_{k=1}^{p} \frac{\alpha_k}{(x_1 + ix_2 - X_B)^k}\right). \tag{25}$$

The field inside D is represented as

$$\phi(\mathbf{x}) = \Re\left(\sum_{l=0}^{p} \beta_l \, (x_1 + ix_2 - X_D)^l\right) \tag{26}$$

with

$$\beta_0 = \alpha_0 \log(X_D - X_B) \; + \; \sum_{k=1}^{\infty} \frac{\alpha_k}{(X_D - X_B)^k}(-1)^k,$$

$$\beta_l = -\frac{\alpha_0}{l \cdot (X_D - X_B)^l} \; + \; \frac{1}{(X_D - X_B)^l} \sum_{k=1}^{\infty} \frac{\alpha_k}{(X_D - X_B)^k} \binom{l+k-1}{k-1} (-1)^k \quad \text{for } l \geq 1.$$

This describes the translation operator denoted by $\mathcal{T}_{ML}$ in section 3.3 and requires $O(p^2)$ work to apply. In [19], Hrycak and Rokhlin suggest an alternative representation of ϕ, based on the formula

$$\frac{1}{z - w} = \int_0^{\infty} e^{-\lambda(z-w)} \, d\lambda. \tag{27}$$

This integral can be discretized using generalized Gaussian quadratures [31] which take into account the nature of the integrand as well as the precise geometry of the interaction list. The number of quadrature nodes needed to achieve a precision ϵ is less than or equal to the number of multipole coefficients. Tables of weights and nodes for various values of ϵ are provided in [31]. For numerical purposes, we begin with an approximation of the form

$$\frac{1}{z_i - w} \approx \sum_{k=1}^{p} w_k e^{-\lambda_k (z_i - w)}.$$

Integrating both sides, we have

$$\log(z_i - w) \approx \sum_{k=1}^{p} \frac{-w_k}{\lambda_k} e^{-\lambda_k (z_i - w)} + C,$$

where C is a constant of integration. Choosing

$$C = \sum_{k=1}^{p} \frac{w_k}{\lambda_k} e^{-\lambda_k}$$

enforces the condition that $\log(1) = 0$.

Instead of the classical multipole expansion (25), we instead work with the *exponential representation*

$$\phi(\mathbf{x}) = \Re\left(\sum_{k=1}^{p} a_k e^{-\lambda_k\,(x_1+ix_2-X_B)} + C'\right), \tag{28}$$

where the coefficients a_k are exponential moments of the charge distribution:

$$a_k = \sum_{j=1}^{M} q_j e^{+\lambda_k\,(z_j-X_B)}$$

and

$$C' = \left(\sum_{j=1}^{M} q_j\right) \cdot C.$$

The advantage of this approach is that translation has been diagonalized. Transmitting the expansion from box B to D is carried out by computing

$$\psi(\mathbf{x}) = \Re\left(\sum_{k=1}^{p} b_k e^{-\lambda_k\,(x_1+ix_2-X_D)} + C'\right), \tag{29}$$

where the new coefficients b_k are obtained from the a_k through the translation formula

$$b_k = a_k\, e^{-\lambda_k(X_D-X_B)}.$$

The details of how to incorporate such expansions into an adaptive two-dimensional FMM code can be found in [19]. For the three-dimensional analogue, see [17].

4. Numerical results. Fast Poisson solvers using the algorithms described above have been implemented in Fortran 77. Here, we demonstrate their performance on four problems involving varying degrees of adaptivity. All of the timings listed below correspond to calculations performed on a 440MHz SUN Ultra-10 with 256 MB RAM using the compiler option (-fast).

There are few efficient adaptive solvers which are widely available. Therefore, we have chosen a simple and stringent standard for comparison: the time taken by a classical FFT-based code for the same number of degrees of freedom (grid points). Using the second order accurate FORTRAN code HWSCRT by Swartztrauber [27] and Swartztrauber and Sweet [28] (available from www.netlib.org), with the same machine and compiler option as above, we obtain the data shown in Table 1.

We have, as yet, said little about our adaptive refinement strategy. It is straightforward. Let B be a leaf node with $k \times k$ grid points, as discussed in section 3.2 and let $f_B(\mathbf{x})$ denote the kth order polynomial used to approximate the right-hand side on B. We then evaluate $f_B(x, y)$ on a $2k \times 2k$ grid covering B and compute the discrete L_2 error $E_2 = \|f(x, y) - f_B(x, y)\|_2$ over these target points. If $E_2 > tol$, the leaf node B is subdivided. Of course, the tree obtained by this procedure may not satisfy the level restriction that neighboring leaf nodes be at most one level apart. It is a straightforward matter to "fix" the tree in a subsequent sweep. We omit the details.

TABLE 1

Timing results for the FFT-based second order accurate code HWSCRT. N denotes the number of grid points, T_{hwscrt} denotes the required solution time in seconds, and rate denote the number of grid points "processed" per second (N/T_{hwscrt}).

N	T_{hwscrt}	Rate
256×256	0.17	$3.8\ 10^5$
512×512	0.78	$3.4\ 10^5$
1024×1024	4.0	$2.6\ 10^5$
2048×2048	19.4	$2.2\ 10^5$

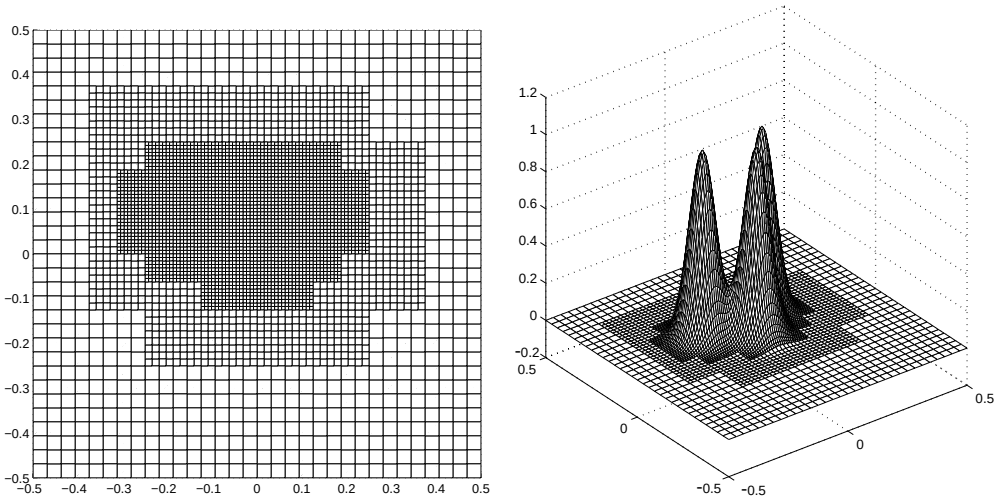

FIG. 4. *The left-hand figure shows an adaptive mesh resolving the source distribution in* (30). *The right-hand figure shows a surface plot of the solution.*

Example 4.1. In our first experiment, we consider the equation

$$\Delta\psi(\mathbf{x}) = \sum_{i=1}^{3} (4\alpha^2 \ \|\mathbf{x} - \mathbf{x_i}\|^2 - 4\alpha) e^{-\alpha\|\mathbf{x}-\mathbf{x_i}\|^2} \tag{30}$$

in free space, for which the exact solution (Figure 4) is the sum of three Gaussians

$$\psi(\mathbf{x}) = \sum_{i=1}^{3} e^{-\alpha\|\mathbf{x}-\mathbf{x_i}\|^2}. \tag{31}$$

We consider the case where $\alpha = 250$, $\mathbf{x_1} = (.1, .1)$, $\mathbf{x_2} = (0, 0)$, and $\mathbf{x_3} = (-.15, .1)$. The right-hand side in (30) is supported, with an exponentially small error, in the box $[-0.5, 0.5]^2$, which we use as the computational domain. Our adaptive mesh is depicted in Figure 4. Note that fine grids are created only near the centers of the Gaussians. The performance of the fourth, sixth, and eighth order codes are summarized in Table 2.

TABLE 2

Timing results for the fourth, sixth, and eighth order accurate codes in Example 4.1. ϵ_{FMM} *denotes the requested precision from far-field interactions within the FMM,* ϵ_{RHS} *denotes the requested precision in discretizing the right-hand side, and* N_{lev} *denotes the number of levels used in the FMM hierarchy.* E_2 *and* E_∞ *denote the relative* L_2 *and* L_∞ *errors of the computed solution,* N *denotes the number of grid points used,* T_{FMM} *denotes the required solution time in seconds, and rate denote the number of grid points "processed" per second* (N/T_{FMM}).

ϵ_{FMM}	ϵ_{RHS}	N_{lev}	E_2	E_∞	N	T_{FMM}	Rate
Fourth order							
10^{-3}	10^{-3}	7	$3.7\ 10^{-4}$	$1.2\ 10^{-4}$	11488	0.08	$1.4\ 10^{5}$
10^{-3}	10^{-6}	9	$7.0\ 10^{-5}$	$1.4\ 10^{-4}$	96592	0.64	$1.5\ 10^{5}$
10^{-6}	10^{-6}	9	$4.9\ 10^{-6}$	$1.3\ 10^{-6}$	96592	1.08	$8.9\ 10^{4}$
10^{-6}	10^{-9}	10	$8.4\ 10^{-8}$	$4.9\ 10^{-7}$	821824	8.38	$9.8\ 10^{4}$
10^{-9}	10^{-9}	10	$1.4\ 10^{-8}$	$3.4\ 10^{-9}$	821824	12.17	$6.8\ 10^{4}$
Sixth order							
10^{-3}	10^{-3}	6	$8.2\ 10^{-5}$	$1.2\ 10^{-4}$	10296	0.08	$1.3\ 10^{5}$
10^{-3}	10^{-6}	7	$7.0\ 10^{-5}$	$1.6\ 10^{-4}$	43236	0.29	$1.5\ 10^{5}$
10^{-6}	10^{-6}	7	$1.5\ 10^{-7}$	$4.2\ 10^{-7}$	43236	0.39	$1.1\ 10^{5}$
10^{-6}	10^{-9}	9	$8.6\ 10^{-8}$	$5.6\ 10^{-7}$	279432	2.45	$1.1\ 10^{5}$
10^{-9}	10^{-9}	9	$2.4\ 10^{-9}$	$2.2\ 10^{-9}$	279432	3.48	$8.0\ 10^{4}$
10^{-9}	10^{-12}	10	$2.3\ 10^{-10}$	$2.4\ 10^{-9}$	1725984	17.19	$1.0\ 10^{5}$
10^{-12}	10^{-12}	10	$2.0\ 10^{-12}$	$8.4\ 10^{-13}$	1725984	27.28	$6.3\ 10^{4}$
Eighth order							
10^{-3}	10^{-3}	6	$1.0\ 10^{-4}$	$2.0\ 10^{-4}$	13888	0.16	$8.7\ 10^{4}$
10^{-3}	10^{-6}	7	$9.0\ 10^{-5}$	$2.0\ 10^{-4}$	63616	0.68	$9.4\ 10^{4}$
10^{-6}	10^{-6}	7	$1.7\ 10^{-7}$	$6.8\ 10^{-7}$	63616	0.80	$8.0\ 10^{4}$
10^{-6}	10^{-9}	8	$1.4\ 10^{-7}$	$6.8\ 10^{-7}$	273280	3.11	$8.8\ 10^{4}$
10^{-9}	10^{-9}	8	$4.4\ 10^{-10}$	$2.7\ 10^{-9}$	273280	3.62	$7.5\ 10^{4}$
10^{-9}	10^{-12}	9	$4.2\ 10^{-10}$	$2.8\ 10^{-9}$	1281472	16.02	$8.4\ 10^{4}$
10^{-12}	10^{-12}	9	$9.2\ 10^{-13}$	$6.1\ 10^{-13}$	1281472	21.68	$5.9\ 10^{4}$

Example 4.2. For our second experiment, we consider the singular equation

$$\begin{aligned} \Delta\psi &= 0 \quad \text{in } D, \\ \psi &= 0 \quad \text{on } \Gamma_L, \\ \psi &= 0 \quad \text{on } \Gamma_R, \\ \psi &= 1 \quad \text{on } \Gamma_T, \\ \psi &= 0 \quad \text{on } \Gamma_B, \end{aligned} \tag{32}$$

In Figure 5, we plot the solution obtained with our solver on an adaptive grid, the solution obtained by HWSCRT on a uniform 64×64 mesh, and the error in the HWSCRT solution. Note that we resolve the corner singularities adaptively and that our solution is exact (up to the requested FMM tolerance), since the data is piecewise polynomial (here, constant).

Remark 4.1. There is an enormous difference in the meaning of "order of accuracy" in our solver and in standard finite difference or finite element codes. Our solver is *exact* for a certain order of approximation of the data. A kth order accurate PDE-based solver on an $N \times N$ mesh has a global error which decays like $(1/N)^k$ with a constant of proportionality which depends on the kth derivative of the solution. In the present example, our solver is exact. The finite difference code is only first order

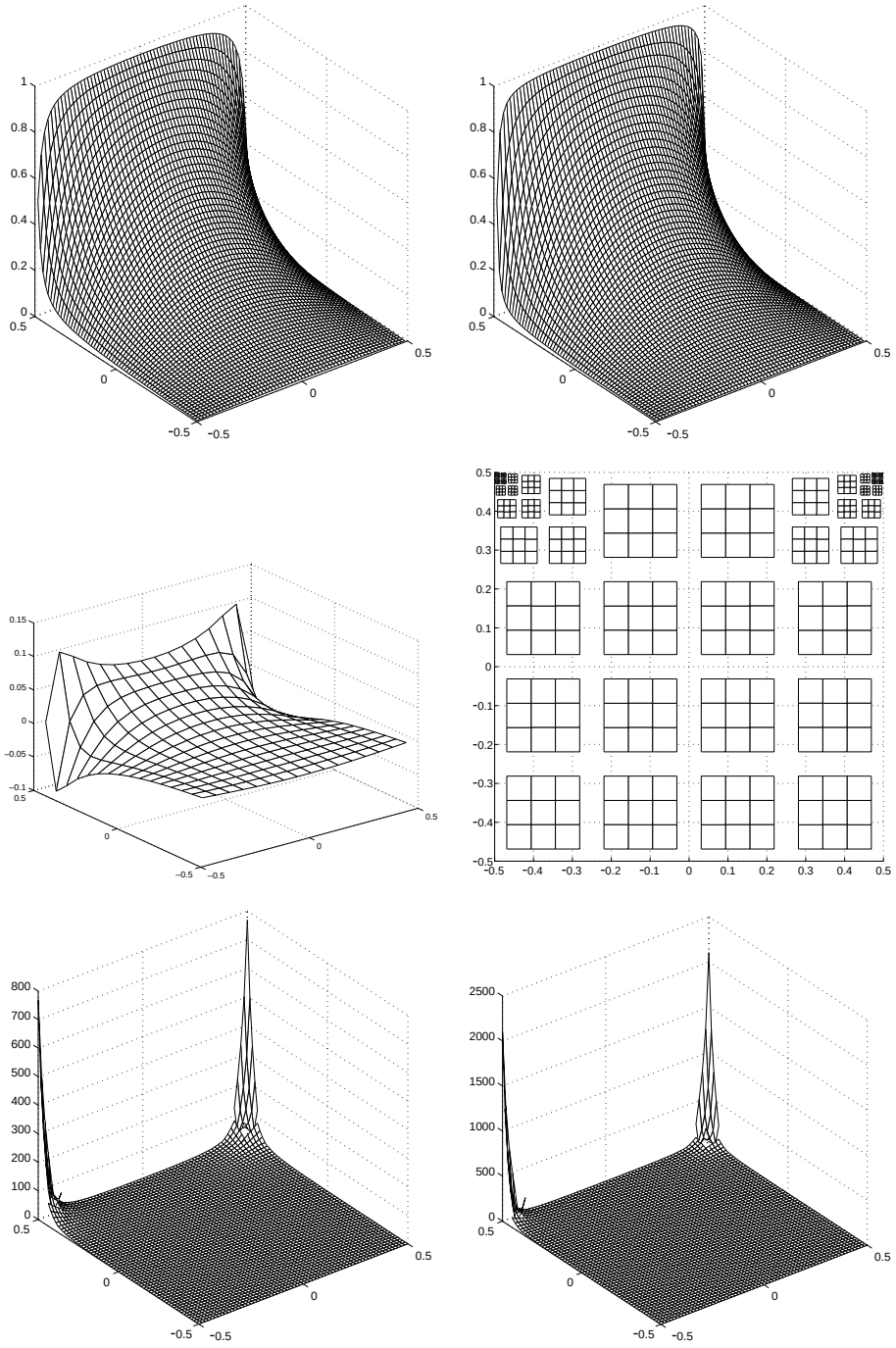

FIG. 5. *The upper left-hand figure shows the solution obtained by HWSCRT to Example* 4.2 *and the upper right-hand figure shows the solution obtained using our solver. The middle figures show the error in the solution obtained by HWSCRT and the adaptive grid use by our solver, respectively. The lower figures show the electrostatic energy* $\|\nabla\psi\|^2$ *obtained from HWSCRT and our solver, respectively.*

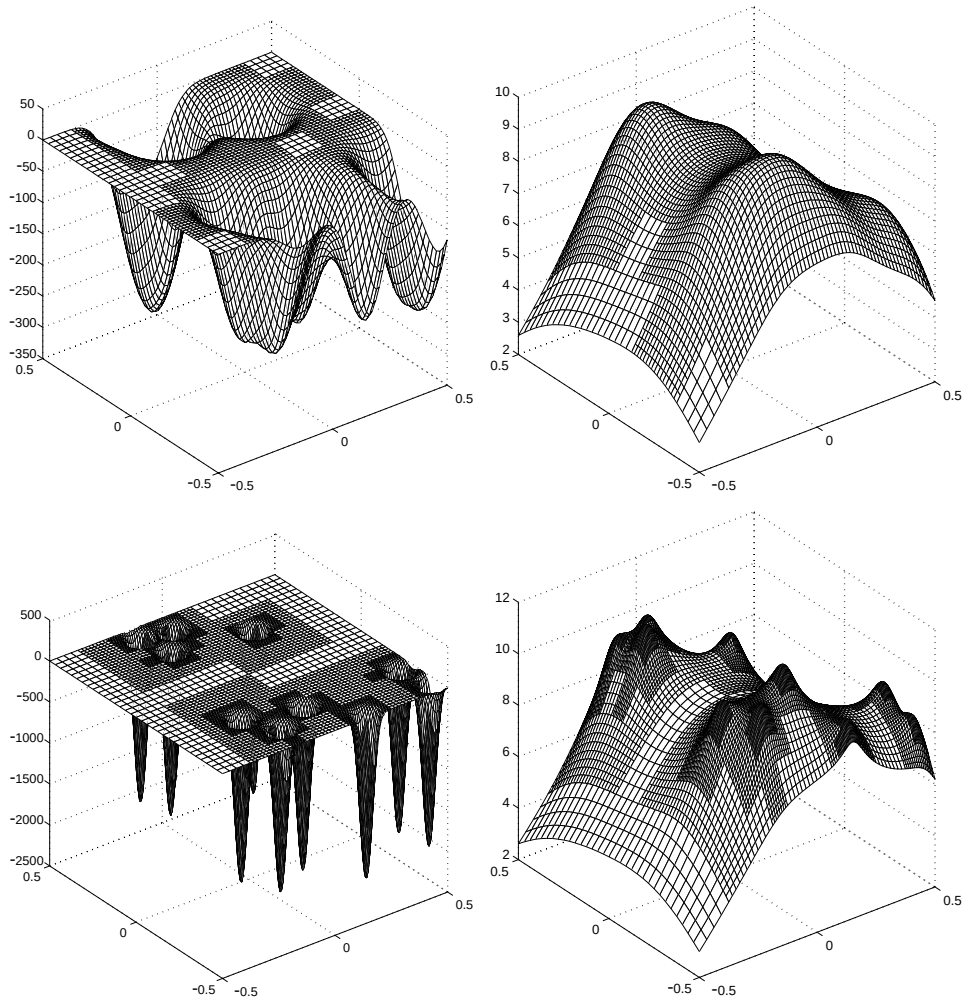

FIG. 6. *The left-most figure shows a surface plot of the right-hand side for* 10 *randomly placed Gaussians as described in* (33) *with* $\alpha = 100$ *and the figure to the right of it shows the corresponding solution. The two lower figures show both the right-hand side and corresponding solution for* $\alpha = 1000$.

accurate because of the corner singularities.

Example 4.3. In order to evaluate the performance of our code with widely varying degrees of adaptivity, we consider the source distribution

$$\Delta\psi(\mathbf{x}) = \sum_{i=1}^{10} -2\alpha e^{-\alpha\|\mathbf{x}-\mathbf{x_i}\|^2} \tag{33}$$

in free space. With $\alpha = 100$, the distribution is fairly smooth, while with $\alpha = 1000$, the Gaussians fall off sharply and require many levels of refinement near the centers (see Figure 6). The experiment was run using the fourth order code and $\epsilon_{FMM} = \epsilon_{RHS} = 10^{-6}$. Table 3 lists our timing data for various values of α.

In addition to comparing various levels of adaptivity, it is also worth noting that

TABLE 3
Timings for the FMM-based solver in Example 4.3.

M	N	T_{FMM}	Rate	N	T_{FMM}	Rate
	$\alpha = 100$			$\alpha = 250$		
5	43969	0.44	$9.9\ 10^4$	73840	0.76	$9.7\ 10^4$
10	70864	0.68	$1.0\ 10^5$	138640	1.43	$9.7\ 10^4$
25	119296	1.33	$8.9\ 10^4$	212800	2.06	$1.0\ 10^5$
100	233296	2.43	$9.6\ 10^4$	262144	2.60	$1.0\ 10^5$
	$\alpha = 500$			$\alpha = 1000$		
5	97648	1.08	$9.0\ 10^4$	108544	1.17	$9.3\ 10^4$
10	196336	2.09	$9.4\ 10^4$	225616	2.40	$9.4\ 10^4$
25	377248	4.03	$9.4\ 10^4$	460912	4.96	$9.3\ 10^4$
100	785632	8.39	$9.4\ 10^4$	916336	9.44	$9.7\ 10^4$

TABLE 4
Timings for the FMM-based solver in Example 4.3 *using free-space and periodic boundary conditions.*

M	N	T_{FMM}	Rate	N	T_{FMM}	Rate
	Free space			Periodic		
10	138640	1.43	$9.7\ 10^4$	139696	1.63	$8.6\ 10^4$
30	241168	2.38	$1.0\ 10^5$	241552	2.63	$9.2\ 10^4$
50	253696	2.41	$1.0\ 10^5$	253840	2.62	$9.7\ 10^4$
100	262144	2.60	$1.0\ 10^5$	262192	2.94	$8.9\ 10^4$

the periodic and free-space solvers execute in nearly the same time. To show this, we consider a right-hand side given by

$$\Delta\psi(\mathbf{x}) = \sum_{i=1}^{M} (-1)^i 2\alpha e^{-\alpha\|\mathbf{x}-\mathbf{x_i}\|^2}. \tag{34}$$

This ensures that the net charge in the periodic cell is zero. Table 4 compares the performance of the fourth order accurate code with either free-space or periodic boundary conditions with $\alpha = 250$ and $\epsilon_{FMM} = \epsilon_{RHS} = 10^{-6}$.

Example 4.4. Most of the preceding examples involve adaptive refinement around a "point-like" singularity. Here we consider the case of a singularity along a curve. We simply define a source distribution which takes the value 1 inside a circle of radius .25 and 0 outside. The right-hand side is shown in Figure 7 along with the computed solution. Using the fourth order solver with 7 levels or refinement and 7936 grid points, the L_2 and L_∞ errors are less than 10^{-3} and the solver executes at the rate $1.3\ 10^5$ points per second. Using the sixth order solver with 12 levels or refinement and 681,408 grid points, the L_2 and L_∞ errors are less than 10^{-6} and the solver executes at the rate $8.5\ 10^4$ points per second. Both of these timings are comparable to those in Example 4.1.

The following observations can be made from the preceding data.

1. The timings for the FMM-based solver grow linearly with the number of unknowns. For fourth order accuracy with a three-digit FMM tolerance, the present implementation achieves a processing speed between $5.9\ 10^4$ and $1.5\ 10^5$ points per second. The classical second order FFT-based solver processes $2.6\ 10^5$ points per second on a 1024×1024 grid.
2. For three-digit accuracy, the fourth order accurate code is fastest ($\approx 1.4\ 10^5$ points per second), while for six-digit accuracy, the sixth order accurate code

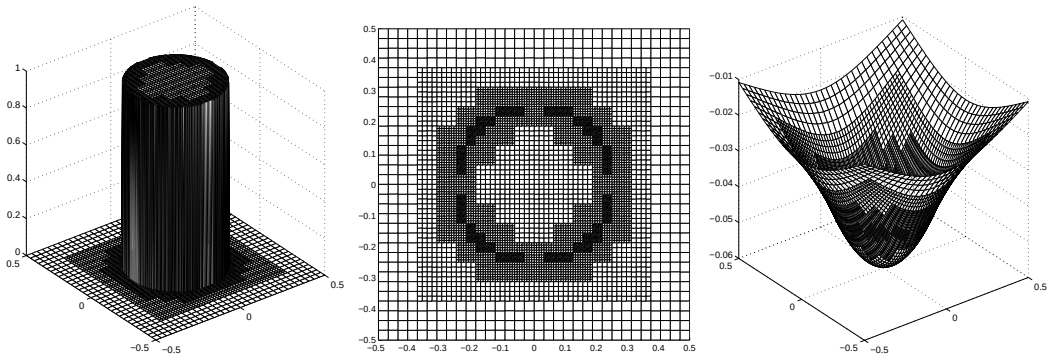

FIG. 7. *The left-hand figure shows the right-hand side. The middle figure shows the grid and the right figure shows the solution.*

is fastest ($\approx 1.1\ 10^5$ points per second). For twelve-digit accuracy, the sixth and eighth order codes are only about twice as slow ($\approx 6.3\ 10^4$ points per second).

5. Conclusions. We have developed a new adaptive, high order accurate solver for the Poisson equation in two dimensions. The method is direct, fast-multipole-based, and allows for the specification of a variety of boundary conditions on a unit square. These include free-space conditions, periodic boundary conditions, Dirichlet, Neumann, and a variety of mixed conditions. The amount of work scales linearly with the number of degrees of freedom in the computational domain and is competitive with classical FFT-based solvers in terms of *work per grid point*, despite the flexibility of adaptive mesh refinement.

In order to develop a black box Poisson solver of broad interest, of course, we need to allow for complex geometry. It would also be of value to be able to solve the Helmholtz and linearized Poisson–Boltzmann equations,

$$\Delta u + \lambda^2 u = f \qquad \text{and} \qquad \Delta u - \lambda^2 u = f,$$

with a similar approach. These extensions are underway and will be reported at a later date.

REFERENCES

[1] A. S. Almgren, J. B. Bell, P. Colella, and L. H. Howell, *An adaptive projection method for the incompressible Euler equation*, in Proceedings of the Eleventh AIAA Computational Fluid Dynamics Conference, 1993, pp. 530–539.

[2] C. Anderson, *Domain decomposition techniques and the solution of Poisson's equation in infinite domains*, in Proceedings of the Second International Symposium on Domain Decomposition Methods, 1988, pp. 129–139.

[3] M. J. Berger and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics*, J. Comput. Phys., 53 (1989), pp. 484–512.

[4] C. L. Berman and L. Greengard, *A renormalization method for the evaluation of lattice sums*, J. Math. Phys., 35 (1994), pp. 6036–6048.

[5] A. Brandt, *Multi-level adaptive solutions to boundary value problems*, Math. Comp., 31 (1977), pp. 330–390.

[6] B. L. Buzbee, G. H. Golub, and C. W. Nielson, *On direct methods for solving Poisson's equations*, SIAM J. Numer. Anal., 7 (1970), pp. 627–656.

[7] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, New York, 1988.
[8] 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.
[9] T. F. Chan, R. Glowinski, J. Periaux, and O. B. Widlund, eds., *Domain Decomposition Methods*, SIAM, Philadelphia, 1989.
[10] T. Chan and B. Smith, *Domain decomposition and multigrid algorithms for elliptic problems on unstructured meshes*, Electron. Trans. Numer. Anal., 2 (1994), pp. 171–182.
[11] G. Chesshire and W. D. Henshaw, *Composite overlapping meshes for the solution of partial differential equations*, J. Comput. Phys., 90 (1991), pp. 1–64.
[12] Z. Gimbutas, L. Greengard, and M. Minion, *Coulomb interactions on planar structures: Inverting the square root of the Laplacian*, SIAM J. Sci. Comput., 22 (2001), pp. 2093–2108.
[13] D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications*, SIAM, Philadelphia, 1977.
[14] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems*, MIT Press, Cambridge, MA, 1988.
[15] L. Greengard and J. Y. Lee, A direct adaptive Poisson solver of arbitrary order accuracy, J. Comput. Phys., 125 (1996), pp. 415–424.
[16] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.
[17] L. Greengard and V. Rokhlin, *A new version of the fast multipole method for the Laplace equation in three dimensions*, Acta Numer., 6 (1997), pp. 229–269.
[18] R. B. Guenther and J. W. Lee, *Partial Differential Equations of Mathematical Physics and Integral Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[19] T. Hrycak and V. Rokhlin, *An improved fast multipole algorithm for potential fields*, SIAM J. Sci. Comput., 19 (1998), pp. 1804–1826.
[20] H. Johansen and P. Colella, *A Cartesian grid embedded boundary method for Poisson's equation on irregular domains*, J. Comput. Phys., 147, (1998), pp. 60–85.
[21] S. McCormick, ed., *Multigrid Methods*, SIAM, Philadelphia, 1987.
[22] A. McKenney, L. Greengard, and A. Mayo, *A fast Poisson solver for complex geometries*, J. Comput. Phys., 118 (1995), pp. 348–355.
[23] M. L. Minion, *A projection method for locally refined grids*, J. Comput. Phys., 127 (1996), pp. 158–177.
[24] A. T. Patera, *A spectral element method for fluid dynamics: Laminar flow in a fluid expansion*, J. Comput. Phys., 54 (1984), pp. 468–488.
[25] Lord Rayleigh, *On the influence of obstacles arranged in a rectangular order upon the properties of the medium*, Philos. Mag., 34 (1892), p. 481.
[26] V. Rokhlin, *Rapid solution of integral equations of classical potential theory*, J. Comput. Phys., 60 (1985), pp. 187–207.
[27] P. N. Swarztrauber, *The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle*, J. Comput. Phys., 15 (1974), pp. 46–54.
[28] P. Swarztrauber and R. Sweet, *Efficient Fortran Subprograms for the Solution of Elliptic Partial Differential Equations*, NCAR Technical Note NCAR-TN/IA-109, 1975, pp. 135–137.
[29] G. Russo and J. Strain, *Fast triangulated vortex methods for the 2D Euler equations*, J. Comput. Phys., 111 (1994), pp. 291–323.
[30] I. Stakgold, *Boundary Value Problems of Mathematical Physics*, Macmillan, New York, 1968.
[31] N. Yarvin and V. Rokhlin, *Generalized Gaussian quadratures and singular value decompositions of integral operators*, SIAM J. Sci. Comput., 20 (1999), pp. 699–718.