A Direct Adaptive Poisson Solver of Arbitrary Order Accuracy

Leslie Greengard* and June-Yub Lee*

March, 1995

Abstract

We present a direct, adaptive solver for the Poisson equation which can achieve any prescribed order of accuracy. It is based on a domain decomposition approach using local spectral approximation, as well as potential theory and the fast multipole method. In two space dimensions, the algorithm requires $O(NK)$ work where N is the number of discretization points and K is the desired order of accuracy.

1 Introduction

A variety of problems in computational physics require the solution of the Poisson equation in situations where the source distribution (the right-hand side) is locally smooth but has complicated structure involving oscillations, internal layers, etc. Such problems require adaptive discretizations to which standard direct solvers [8] do not apply.

To simplify the ensuing discussion, we will restrict our attention to the solution of the Poisson equation

$$\Delta u = f \tag{1}$$

in the plane, in the absence of physical boundaries. The imposition of boundary conditions in domains with complicated geometry can be achieved in a

*Courant Institute of Mathematical Sciences, New York University, New York, New York 10012. This work was supported by the Applied Mathematical Sciences Program of the U.S. Department of Energy under Contract DEFGO288ER25053, by the Office of Naval Research under Contract N00014-91-J-1312, by a NSF Presidential Young Investigator Award to L.G. and by a Packard Foundation Fellowship to L.G.

subsequent step by using standard potential theory. Once the solution $u(\mathbf{x})$ to (1) has been obtained, one need only solve an auxiliary Laplace equation using a boundary integral method [16, 29]. We refer the reader to [24] for a detailed description of such a solver. For our purposes here, it is sufficient to note that the method of [24] uses the algorithm of [22] for evaluating the volume integral. The latter algorithm is fast, achieves second or fourth order accuracy, and allows for discontinuities in the right-hand side, but it relies on a uniform underlying mesh. We would like to allow more complex volume discretizations.

Almost all currently available methods are based on iterative techniques using multigrid [7, 23], domain decomposition [11], or some other preconditioning strategy. Unfortunately, while multilevel iterations can achieve optimal efficiency in theory, they require an appropriate hierarchy of coarse grids which are not provided in many practical situations. There has, however, been significant progress made in this direction over the last few years. Good coarsening strategies can be found in [1, 27] for locally uniform meshes based on adaptive mesh refinement [5, 6]. Other useful schemes have been designed for composite overlapping grids [4, 13], quad-trees [14, 25], and unstructured triangulations [3, 12, 21].

Leaving aside finite difference and finite element discretizations, one could also solve (1) by direct evaluation of the exact solution in the form of a volume integral

$$u(\mathbf{x}) = \frac{1}{2\pi} \int_{\mathbf{R}^2} \log |\mathbf{x} - \mathbf{y}| f(\mathbf{y})\, d\mathbf{y}. \tag{2}$$

Such an approach has been developed for the case of unstructured triangulations by Russo and Strain [31]. They have produced a robust second-order accurate solver, based on applying the fast multipole method (FMM) [10, 17] directly to a quadrature approximation of the expression in (2). The implementation is nearly optimal in terms of asymptotic computational complexity, but requires a significant amount of work per gridpoint.

In this paper, we present a kind of domain decomposition or spectral element method [9, 28], which is fast and direct and assumes only that the right-hand side is defined on the leaf nodes of an adaptive quad-tree data structure. A related, but nonadaptive, method based on Fourier analysis is described in [2]. We proceed by solving local Poisson problems on each subregion using a spectral method and then coupling all the local solutions together, using potential theory and the FMM. The advantages of this approach over applying the FMM to the volume integral (2) directly are two-

fold. First, it is easy to obtain high order accuracy, and second, the CPU requirements are dramatically diminished. In fact, the algorithm presented here requires, for sixteenth order accuracy, about 500 floating point operations per grid point. Problems with one million unknowns require between 10 and 20 minutes to solve on a SPARCstation 2.

2 Mathematical Preliminaries

We assume that the source distribution f in (1) or (2) is supported inside a square D, on which is superimposed a hierarchy of refined grids. 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. The leaf nodes on which the source distribution is assumed to be given will be denote by D_i.

Our assumption that f is locally smooth is taken to mean that it is accurately represented on each leaf node by a local polynomial approximation. The basic strategy of our algorithm is to solve uncoupled Poisson problems on the leaf nodes and then to patch the solutions together analytically, using layer potentials.

2.1 The local solver

Consider now an arbitrary leaf node D_i in a given quad-tree refinement of D (Fig. 1). We define the local problem by the equation

$$\begin{aligned} \Delta u_i(\mathbf{x}) &= \begin{cases} f_i(\mathbf{x}) & \text{if } \mathbf{x} \in D_i \\ 0 & \text{if } \mathbf{x} \notin D_i \end{cases} \\ u_i(\mathbf{x}) &= O(\log|\mathbf{x}|) \quad \text{as } \mathbf{x} \to \infty, \end{aligned} \tag{3}$$

where f_i denotes the restriction of f to D_i.

Computing the solution to (3) is a non-trivial task, since u_i is weakly singular along the boundary of D_i, where the right-hand side is discontinuous. To overcome this difficulty, we have developed a rather nonstandard "local solver," based on Chebyshev approximation. By way of brief review, we note that the Chebyshev polynomial of degree k on $[-1,1]$ can be defined

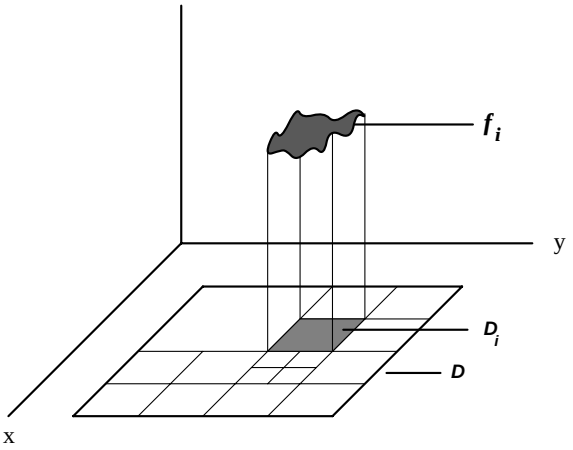

Figure 1: **The local solver.** For each leaf node D_i in a quad-tree refinement of D, the local Poisson problem defined by eq. (3) involves a discontinuous right-hand side.

by the formula

$$T_k(\cos\theta) = \cos(k\theta), \tag{4}$$

or by the recurrence

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{k+1}(x) &= 2xT_k(x) - T_{k-1}(x) \quad \text{for } k \geq 1\ . \end{aligned}$$

Given a smooth function f on $[-1,1] \times [-1,1]$, we define the Chebyshev expansion of f by

$$f(x_1, x_2) = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} f_{n,m} T_n(x_1) T_m(x_2),$$

where

$$f_{n,m} = \frac{4}{c_n\, c_m} \int_0^{\pi} \int_0^{\pi} f(\cos\theta_1, \cos\theta_2) \cos(n\theta_1)\ \cos(m\theta_2)\, d\theta_1\, d\theta_2. \tag{5}$$

In the preceding expression, $c_0 = 2$ and $c_j = 1$ for $j \geq 1$. See [15] for a more complete discussion.

Remark 2.2: The truncated Chebyshev expansion

$$f(x_1, x_2) \approx \sum_{n=0}^{K} \sum_{m=0}^{K-n} f_{n,m} T_n(x_1) T_m(x_2) \tag{6}$$

is an approximation of order $K+1$, since it includes all polynomials of degree less than or equal to K.

Remark 2.3: It is possible to compute the coefficients $f_{n,m}$ of the truncated Chebyshev expansion in $O(K^2 \log K)$ operations using the fast cosine transform (see, for example, [9]).

Lemma 2.1 *Let $T_n(x_1)\, T_m(x_2)$ be a polynomial of degree $n+m$. Then there exists a polynomial $P_{n,m}(x_1, x_2)$ of degree $n+m+2$ such that*

$$\Delta P_{n,m}(x_1, x_2) = T_n(x_1)\, T_m(x_2).$$

Proof: Let $\mathcal{I}$ denote the the indefinite integration operator:

$$\mathcal{I} f(x) = \int_{-1}^{x} f(t)\, dt,$$

and let $\mathcal{D}$ denote the differentiation operator. If $n \geq m$, the following polynomial satisfies the desired conditions:

$$P_{n,m}(x_1, x_2) = \sum_{k=0}^{\lceil \frac{m}{2} \rceil} (-1)^k \left[\mathcal{I}^{2k+2} T_n\right](x_1) \left[D^{2k} T_m\right](x_2). \tag{7}$$

The construction for $n \leq m$ is analogous. □

Theorem 2.1 *Let $f : [-1,1] \times [-1,1] \to \mathbf{R}$ be a polynomial of degree K,*

$$f(x_1, x_2) = \sum_{n=0}^{K} \sum_{m=0}^{K-n} f_{n,m} T_n(x_1) T_m(x_2),$$

and let

$$u(x_1, x_2) = \sum_{n=0}^{K} \sum_{m=0}^{K-n} f_{n,m} P_{n,m}(x_1, x_2),$$

where $P_{n,m}$ is defined in Lemma 2.1. Then $u(x_1, x_2)$ is a polynomial of degree $K+2$ satisfying $\Delta u = f$ on $[-1,1] \times [-1,1]$.

Proof: The theorem follows immediately from Lemma 2.1. □

Remark 2.4: The construction of the preceding theorem need not depend on the precise dimensions of the box, since the Chebyshev polynomials can be rescaled to any interval. For a given square D_i, we will refer to the (scaled) Chebyshev expansion coefficients of the local source distribution f_i as $\{f^i_{n,m}\}$.

Definition 2.1 *For a given square D_i, we define the (piecewise) smooth part of the local solution u_i to be the function*

$$u_i^s(x_1,x_2) = \begin{cases} \sum_{n=0}^{K} \sum_{m=0}^{K-n} f^i_{n,m} P_{n,m}(x_1,x_2), & \text{if } \mathbf{x} \in D_i \\ 0 & \text{if } \mathbf{x} \notin D_i \end{cases} \tag{8}$$

where, with a slight abuse of notation, $P_{n,m}$ is the polynomial defined in Lemma 2.1 scaled to the proper dimensions of D_i.

While u_i^s satisfies the Poisson equation (3), it sustains jumps in both function value and normal derivative across the boundary of D_i. Fortunately, the difference

$$u_i^h = u_i - u_i^s$$

has a rather simple structure. It is harmonic both inside and outside the square, and given by standard potential theory [18, 20, 26] as a combination of single and double layer potentials:

$$u_i^h(\mathbf{x}) = \int_{\partial D_i} G(\mathbf{x},\mathbf{y}) \frac{\partial u_i^s}{\partial n}(\mathbf{y})\, dt_{\mathbf{y}} + \int_{\partial D_i} \frac{\partial G}{\partial n}(\mathbf{x},\mathbf{y}) u_i^s(\mathbf{y})\, dt_{\mathbf{y}}, \tag{9}$$

where $G(\mathbf{x},\mathbf{y}) = \frac{1}{2\pi}\log|\mathbf{x}-\mathbf{y}|$, $dt_{\mathbf{y}}$ is an element of arc length, and $\frac{\partial}{\partial n}$ denotes the outward normal derivative.

Remark 2.5: Note that the densities of the layer potentials, namely $\frac{\partial u_i^s}{\partial n}$ and u_i^s, are smooth functions. In fact, by Theorem 2.1, they are polynomials of degree $K+2$.

Definition 2.2 *The concatenation of smooth local solutions is defined by*

$$u^s(\mathbf{x}) = \begin{cases} u_i^s(\mathbf{x}) & \text{if } \mathbf{x} \in D_i \\ 0 & \text{if } \mathbf{x} \in \mathbf{R}^2 - D. \end{cases} \tag{10}$$

We may summarize the preceding discussion in the following theorem.

Theorem 2.2 *Let the function $f(\mathbf{x})$ be supported in a square domain D, on which is superimposed an adaptive quad-tree subdivision of space with leaf nodes D_i for $i = 1, \ldots, M$. Suppose, further, that on each square D_i, $f(\mathbf{x})$ is given by a polynomial of degree K. Then, for $\mathbf{x} \in \mathbf{R}^2$, the solution to the Poisson equation (1) is given by*

$$u(\mathbf{x}) = u^s(\mathbf{x}) + \sum_{i=1}^{M} u_i^h(\mathbf{x}). \tag{11}$$

2.2 Modifications of the Fast Multipole Method

In this section, we describe the changes which need to be made to the fast multipole method (FMM) [10, 17] in order to evaluate $\sum_{i=1}^{M} u_i^h(\mathbf{x})$, but leave a detailed description of that method to the original papers.

Consider a typical leaf node D_i in the quad-tree refinement of D. Then, outside the region covered by its neighbors, the function u_i^h can be accurately represented as a multipole expansion about the box center, using only a small number of terms. To obtain a formal expansion, we first observe from equation (9) that

$$u_i^h(z) = Re\left(\frac{1}{2\pi i}\int_{\partial D_i} \frac{\partial u_i^s}{\partial n}(w)\log(w-z)\,dt_w + \frac{1}{2\pi i}\int_{\partial D_i} \frac{u_i^s(w)}{w-z}\,dw\right), \tag{12}$$

where we have equated the points $\mathbf{x}$ and $\mathbf{y}$ in $\mathbf{R}^2$ with the points z and w in the complex plane. It is straightforward to prove the following result.

Lemma 2.2 *Let D_i be a square centered at the origin of dimension $L \times L$ and let $z = x + iy$ be a point in the plane with $|x| \geq 3L/2$ or $|y| \geq 3L/2$ (or both). Then*

$$u_i^h(z) = Re\left(a_0 \log z - \sum_{l=1}^{\infty} \frac{a_l}{z^l}\right), \tag{13}$$

where

$$a_0 = \frac{1}{2\pi i}\int_{\partial D_i} \frac{\partial u_i^s}{\partial n}(w)\,dt_w \tag{14}$$

and

$$a_l = \frac{1}{2\pi i}\left(\int_{\partial D_i} \frac{w^l}{l}\frac{\partial u_i^s}{\partial n}(w)\,dt_w + \int_{\partial D_i} w^{l-1}\,u_i^s(w)\,dw\right) \tag{15}$$

for $l \geq 1$. The error in truncating the expansion after p terms is given by

$$\begin{aligned} E_p &= \left| u_i^h(z) - Re\left(a_0 \log z - \sum_{l=1}^{p} \frac{a_l}{z^l} \right) \right| \\ &\leq \left(A + \frac{B}{|z|} \right) \left(\frac{\sqrt{2}}{3} \right)^p, \end{aligned} \tag{16}$$

where

$$\begin{aligned} A &= \frac{1}{2\pi} \int_{\partial D_i} \left| \frac{\partial u_i^s}{\partial n}(w) \right| \, dt_w \\ B &= \frac{1}{2\pi} \int_{\partial D_i} |u_i^s(w)| \, dt_w. \end{aligned}$$

Since the single and double layer densities $\frac{\partial u_i^s}{\partial n}$ and u_i^s are polynomials of degree $K+2$, the multipole coefficients in equations (14) and (15) can be computed exactly. The most important thing to note, however, is that after forming the multipole expansions for all leaf nodes, the process of

(1) merging expansions at coarser refinement levels,
(2) converting multipole expansions to local expansions,
(3) shifting local expansions from parents to children,

is carried out just as in a standard implementation of the FMM.

It remains only to say something about computing the influence of a leaf node D_i on itself and its neighbors, corresponding to the "direct interaction" step in a standard FMM. For this, we will require some additional notation. Let $D_i = [t_1, t_2] \times [t_3, t_4]$ and let $d = N, S, E$ or W refer to the north, south, east, or west side of the boundary, respectively. The individual boundary segments themselves will be denoted by ∂D_i^d. We can then define

$$\sigma_i^d(t) = \frac{\partial u_i^s}{\partial n}(\mathbf{y}_d(t)), \quad \mu_i^d(t) = u_i^s(\mathbf{y}_d(t)), \tag{17}$$

where

$$\mathbf{y}_N(t) = (t, t_4),\ \mathbf{y}_S(t) = (t, t_3),\ \mathbf{y}_E(t) = (t_2, t),\ \mathbf{y}_W(t) = (t_1, t).$$

Since all these functions are polynomials of degree $K+2$, we may write

$$\sigma_i^d(t) = \sum_{k=0}^{K+2} \sigma_{i,k}^d \, \tilde{T}_k(t), \quad \mu_i^d(t) = \sum_{k=0}^{K+2} \mu_{i,k}^d \, \tilde{T}_k(t),$$

where $\tilde{T}_k(t)$ is the Chebyshev polynomial scaled to the proper dimensions of D_i. But then u_i^h in equation (9) can be written in the form

$$\begin{aligned} u_i^h(\mathbf{x}) &= \sum_{k=0}^{K+2} (S_k^N \sigma_{i,k}^N + S_k^S \sigma_{i,k}^S + S_k^E \sigma_{i,k}^E + S_k^W \sigma_{i,k}^W) \\ &+ \sum_{k=0}^{K+2} (D_k^N \mu_{i,k}^N + D_k^S \mu_{i,k}^S + D_k^E \mu_{i,k}^E + D_k^W \mu_{i,k}^W), \end{aligned} \tag{18}$$

where

$$\begin{aligned} S_k^d = S_k^d(\mathbf{x}) &= \int_{\partial D_i^d} G(\mathbf{x}, \mathbf{y}_d(t)) \tilde{T}_k(t)\, dt \\ D_k^d = D_k^d(\mathbf{x}) &= \int_{\partial D_i^d} \frac{\partial G}{\partial n}(\mathbf{x}, \mathbf{y}_d(t)) \tilde{T}_k(t)\, dt. \end{aligned}$$

Thus, for each target point of interest $\mathbf{x}$, the function u_i^h can be evaluated using $8(K+3)$ operations, assuming that we have precomputed the coefficients $\{S_k^d, D_k^d\}$.

3 The Numerical Method

We begin with some notation concerning the adaptive data structure. We assume that we are given a square D, which contains the support of the right hand side f, as well as a quad-tree refinement of D which respects the condition that two leaf nodes in the tree which share a boundary segment live at most one refinement level apart (Fig. 2).

Definition 3.1
a) For each square S, the neighbors *consist of those squares at the same refinement level with which it shares a boundary point, as well as leaf nodes at coarser levels with which it shares a boundary point.*
b) For each square S, the interaction region *consists of the area covered by the neighbors of S's parent, excluding the neighbors of S. The* interaction list *consists of those squares in the interaction region which are at the same refinement level, as well as leaf nodes in the interaction list which are at coarser levels.*

Letting M be the number of leaf nodes and K be the desired order of accuracy, we construct a (scaled) $K \times K$ Chebyshev mesh on each leaf node

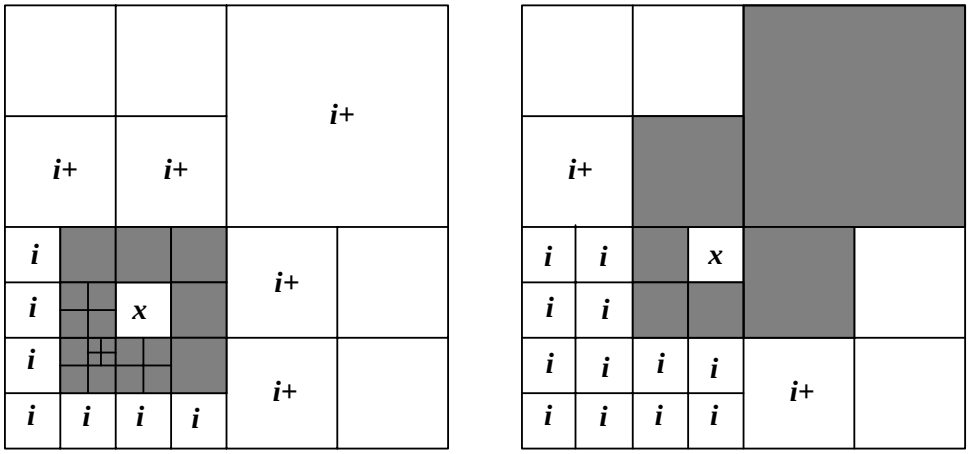

Figure 2: **Adaptive subdivision of a square domain** D**.** In (a), all leaf nodes are visible and the neighbors of the square marked by an **x** are indicated by shading. The elements of the interaction list are indicated by an **i** or an **i**+, depending on whether they are at the same refinement level or at a coarser one. Note that some of the neighbors at the same refinement level are further subdivided, resulting in a somewhat complex local structure. In (b), the neighbors and interaction list of the square marked by an **x** are again indicated by shading or by the labels **i** and **i**+. We have omitted the refinements of some of the members of the interaction list in (b), since those refinements are of no consequence to the marked square under consideration.

D_i for $i = 1, \ldots, M$. The total number of discretization points is given by $N = MK^2$. We will also use the following definitions:

- $S_{0,0} =$ original domain D.
- $S_{l,k} =$ k^{th} square at refinement level l.
- $\Phi_{l,k} =$ multipole expansion describing the far field of $\sum_j u_j^h$, where the index of summation runs over all leaf nodes D_j contained within $S_{l,k}$.
- $\Psi_{l,k} =$ local expansion describing the field $\sum_j u_j^h$, where the index of summation runs over all leaf nodes outside the neighbors of $S_{l,k}$.

The quad-tree structure is obtained by dividing a square subdomain $S_{l-1,j}$ into four smaller subdomains $S_{l,4j+k}$, for $k = 0, 1, 2, 3$ (subject to the constraint that two distinct leaf nodes which share a boundary segment live at most one level apart).

Remark 3.1: In the preceding discussion, leaf nodes have two characterizations, as squares D_i and as tree elements $S_{l,k}$ for some l and k. When it is clear from the context, we will refer to the multipole and local expansions for D_i by Φ_i and Ψ_i, repectively.

Remark 3.2: Note that the function $\sum_{i=1}^{M} u_i^h(\mathbf{x})$ depends only on the boundary values of u_i^s and $\frac{\partial u_i^s}{\partial n}$. Note also that on the boundary of each leaf node ∂D_i, the exact solution is given by

$$u_{\partial D_i}(\mathbf{x}) = u_i^s(\mathbf{x}) + \sum_{j=1}^{M} u_j^h(\mathbf{x}).$$

Once the boundary values $u_{\partial D_i}$ are known, the local Poisson problem

$$\begin{aligned} \Delta u_i(\mathbf{x}) &= f_i(\mathbf{x}) \quad \text{in } D_i & (19) \\ u_i(\mathbf{x}) &= u_{\partial D_i}(\mathbf{x}) \quad \text{on } \partial D_i & (20) \end{aligned}$$

has a smooth solution which is, in fact, the restriction to D_i of the desired solution u.

Algorithm

Initialization

Comment [We assume we are given a square domain $D = S_{0,0}$, on which is superimposed a hierarchical quad-tree structure. Internal nodes will be referred to as $S_{l,k}$, while leaf nodes will be referred to as D_i for $i = 1, \ldots, M$.]

a) Choose order K of polynomial approximation on each leaf node.
b) Choose order p of multipole expansions
($p \approx \log_2 \epsilon$, where ϵ is the desired accuracy).

Step I: Solve local problems in free space

do $i = 1, ..., M$
a) for leaf node D_i, compute the Chebyshev expansion of the right-hand side f_i via eq. (5).
b) for leaf node D_i, evaluate u_i^s and $\frac{\partial u_i^s}{\partial n}$ on the boundary via definition 2.1 and theorem 2.1.
c) form the multipole expansion $\Phi_i(\mathbf{x})$ for $u_i^h(\mathbf{x})$ according to formulae (13), (14), (15).
end

Cost [Step a) requires approximately $10MK^2 \log K$ operations, while b) requires $8MK^3$ operations. c) requires $8MKp$ operations.]

Step II: Multipole Sweep

Upward pass
for all internal nodes $S_{l,j}$
form the multipole expansion $\Phi_{l,j}$ by merging the mulitpole expansions of its four children.
end
Downward pass
for all nodes $S_{l,j}$

form the local expansion $\Psi_{l,j}$ by adding the
local expansion of its parent to the contributions from the
multipole expansions of all elements of its interaction list.
end

Cost [The upward pass requires approximately Mp^2 work, while the downward pass requires approximately $27Mp^2$ work (see Remark 3.3 below).]

Step III: Leaf node processing

Comment [At this point, for each leaf node D_i, $\Psi_i(\mathbf{x}) = \sum_j u_j^h(\mathbf{x})$, where the index of summation runs over all leaf nodes D_j outside the neighbors of D_i.]

do $i = 1, ..., M$
for each boundary point of the $K \times K$ Chebyshev mesh on D_i,
a) evaluate the local expansion $\Psi_i(\mathbf{x})$
b) evaluate u_j^h induced by each neighboring leaf node D_j
c) evaluate u_i^h
d) add the values from a), b), c) to the local solution u_i^s.
end
end

Comment [We have now computed the correct boundary values $u_{\partial D_i}$ on each leaf node.]

Cost [Step a) requires approximately $4MKp$ operations, while b) requires approximately $24MK^2$ operations. c) requires $2MK$ operations.]

Step IV: Solve local Dirichlet problems

do $i = 1, ..., M$
a) Solve the local problem (19), (20) with K^{th} order spectral method.
end

Cost [By precomputation of the LU factorization of the $K^2 \times K^2$ system matrix, the cost per leaf node is K^4 operations.]

Remark 3.1: It is, in fact, somewhat difficult to estimate the costs of Steps II and III precisely, since they depend on the actual structure of the adaptive quad-tree. The stated cost is accurate in the case of a uniform mesh and a good, often pessimistic, approximation in more general settings. (Note, for example, that if a leaf node D_i has subdivided neighbors, more direct interactions will be needed to evaluate the field on the boundary of D_i. This increase is offset by a decrease in the direct interactions needed by the neighboring, finer level leaf nodes themselves.)

In summary, the CPU requirements of the algorithm are approximately

$$N\left(10\log K + 8K + 2\frac{p}{K} + \frac{27p^2}{K^2} + 2\frac{p}{K} + 24 + K^2\right).$$

The cost of Step 4 could be reduced by the use of a more sophisticated local solver. There are direct methods requiring $O(K^3)$ work per leaf node [15, 9] as well as a recently-developed high order direct solver [32], which requires $O(K^2 \log K)$ work. Since we are primarily interested in $K \leq 16$, we have chosen the simplest scheme, which requires $O(K^4)$ work per leaf node. For $K > 16$, we recommend switching to one of these other methods, which would bring the asymptotic CPU requirements to $O(NK)$.

4 Numerical Results

The algorithm described above has been implemented in double precision using a combination of C++ and Fortran, C++ for the adaptive bookkeeping and Fortran for the basic numerical modules. In this section, we illustrate its performance using a Sun SPARCstation 2 on a suit of test problems.

Example 1 (A Gaussian distribution on a uniform grid).

We first consider the equation

$$\Delta u = (400^2 r^2 - 800)\, e^{-400r^2/2}, \tag{21}$$

where $r^2 = x^2 + y^2$, for which the exact solution is given by

$$u = e^{-400r^2/2}. \tag{22}$$

The source distribution is supported, with an exponentially small error, in the box $[-0.5, 0.5]^2$, which we use as our computational domain D. In order to first study the rate of convergence without the complications introduced by adaptivity, we decomposed D uniformly into leaf nodes. Our results are summarized in Tables 1 and 2, the first corresponding to eighth order accuracy ($K = 8$) and the second to sixteenth order accuracy ($K = 16$). In each table, the first column indicates the number of refinement levels and the second column indicates the total number of grid points. Columns 3 and 4 show the relative error in the L^2 norm of the computed solution, using $p = 21$ or $p = 42$ multipole moments, while columns 5 and 6 indicate the amount of CPU time required.

# of levels	# of grid points	Relative Error		CPU Time (sec)	
		p=21	p=42	p=21	p=42
0	64	$2.24{\cdot}10^{1}$	$2.24{\cdot}10^{1}$	0.1	0.1
1	256	$1.16{\cdot}10^{-1}$	$1.16{\cdot}10^{-1}$	0.2	0.3
2	1024	$5.40{\cdot}10^{-3}$	$5.40{\cdot}10^{-3}$	0.7	1.0
3	4096	$4.86{\cdot}10^{-5}$	$4.86{\cdot}10^{-5}$	2.7	4.4
4	16384	$1.79{\cdot}10^{-7}$	$1.79{\cdot}10^{-7}$	10.5	18.4
5	65536	$4.30{\cdot}10^{-10}$	$3.14{\cdot}10^{-10}$	42.4	74.5
6	262144	$1.86{\cdot}10^{-10}$	$6.34{\cdot}10^{-13}$	171.1	301.2
7	1048576	$1.51{\cdot}10^{-10}$	$1.63{\cdot}10^{-15}$	687.2	1217.0

Table 1: Performance of the eighth order method ($K = 8$) for Example 1.

For $p = 21$, we see that the error is dominated by the multipole approximation once the discretization error falls below about 10^{-10}. For $p = 42$, full double precision accuracy is achieved. We have also broken down the timings for two representative calculations, to see how costly the various steps of the algorithm are in actual practice (Table 3). Note that at $K = 16$, even requiring fourteen digit multipole accuracy, the most expensive part of the algorithm is step 4.

Example 2 (Random distribution of Gaussian sources).

We have selected for our second example a distribution of Gaussian sources,

$$\Delta u = \Sigma_{i=1}^{14} e^{-\omega_i r_i^2}, \tag{23}$$

where $r_i^2 = (x - x_i)^2 + (y - y_i)^2$, the centers (x_i, y_i) are randomly located

# of levels	# of grid points	Relative Error		CPU Time (sec)	
		p=21	p=42	p=21	p=42
0	256	2.31	2.31	2.6	2.7
1	1024	$5.11 \cdot 10^{-4}$	$5.11 \cdot 10^{-4}$	3.1	3.2
2	4096	$7.86 \cdot 10^{-7}$	$7.86 \cdot 10^{-7}$	5.3	5.6
3	16384	$4.98 \cdot 10^{-10}$	$4.71 \cdot 10^{-11}$	13.6	15.5
4	65536	$5.39 \cdot 10^{-10}$	$3.77 \cdot 10^{-15}$	47.7	55.9
5	262144	$2.94 \cdot 10^{-10}$	$3.24 \cdot 10^{-15}$	184.2	216.7

Table 2: Performance of the sixteenth order method ($K = 16$) for Example 1.

p	K	N	init.	step 1	steps 2 & 3	step 4	misc (sec)
21	8	65536	0.20	5.21	24.34	8.49	3.18
42	16	262144	5.57	28.98	80.33	84.17	17.65

Table 3: Breakdown of timings for Example 1 using 1024 leaf nodes.

in the box $[-0.4, 0.4]^2$, and $\omega_i \in [1024, 16384]$. The computational domain D is again the box $[-0.5, 0.5]^2$. An adaptive discretization is obtained by requiring that the right-hand side be resolved to a specified tolerance on each leaf node (Fig. 3). We achieve this by examining the local Chebyshev coefficients of the right-hand side and subdividing those nodes on which the tail of the series has not decayed sufficiently, according to the criterion

$$\sum_{n=0}^{K} |f^i_{n,K-n}| < \quad \text{discretization tolerance.} \tag{24}$$

In Table 4, we show the discretization tolerance and the multipole tolerance, as well as the number of grid points and CPU times required by the eighth order and sixteenth order schemes. One can see that the eighth order method is more efficient if fewer thah 8 digits of accuracy are required, but that the sixteenth order method is preferable for higher precision.

Example 3 (Sharp transition layers).

To test the algorithm on a source distribution with internal layers, we let

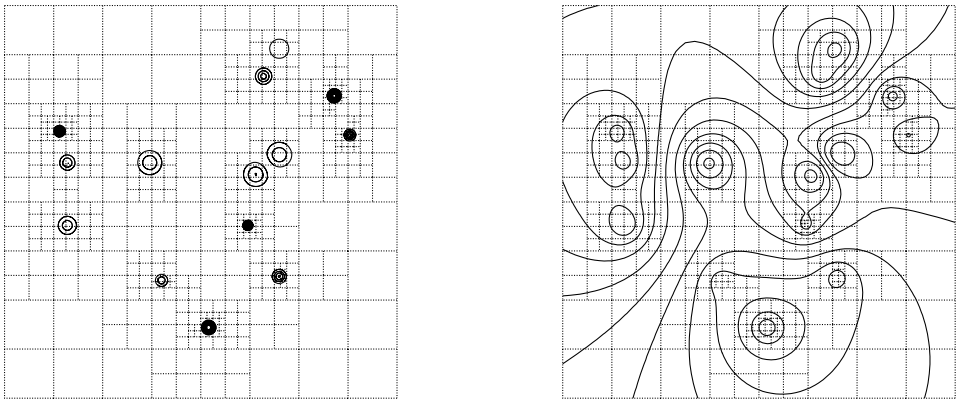

Figure 3: The left-hand figure shows a random distribution of Gaussian sources and the superimposed adaptive quad-tree. On the right is a contour plot of the solution.

discretization tolerance	multipole tolerance	K=8		K=16	
		N	CPU time	N	CPU time
10^{-4}	10^{-4}	13120	16.4	23296	32.3
10^{-6}	10^{-6}	31168	42.0	47872	64.7
10^{-8}	10^{-8}	67456	99.3	75520	103.2
10^{-10}	10^{-10}	146560	234.1	123136	173.9
10^{-12}	10^{-12}	355072	615.4	197632	283.5
10^{-14}	10^{-14}	866468	1648.9	301312	448.0

Table 4: Performance of the eighth and sixteenth order methods in Example 2. N denotes the number of points and CPU times are listed in seconds.

$$\begin{aligned} g(n) &= \operatorname{erf}(\sigma\,\tfrac{(n+1)}{4}) + \operatorname{erf}(\sigma\,\tfrac{(n-1)}{4}) - \operatorname{erf}(\sigma\,n) \\ h(p) &= \operatorname{erf}(\sigma\,\tfrac{(p+1)}{8}) + \operatorname{erf}(\sigma\,\tfrac{(p-1)}{8}) \\ f(x,y) &= g(\tfrac{y-x}{\sqrt{2}}) \cdot h(\tfrac{y+x}{2}) \end{aligned}$$

This function is approximately equal to 1 in the upper left-hand rectangle visible in Figure 4 and -1 in the lower right-hand rectangle. We let $\sigma = 256$ so that the width of the transition region is of the order $1/256$. 610 boxes are required using $K = 8$ to enforce a discretization tolerance of 10^{-6} according to the local criterion (24). The corresponding number of points is 39,040 and the solver requires 45 seconds. The relative error of the computed solution in the L^2 norm is 10^{-8}, two orders of magnitude smaller than the tolerance criterion.

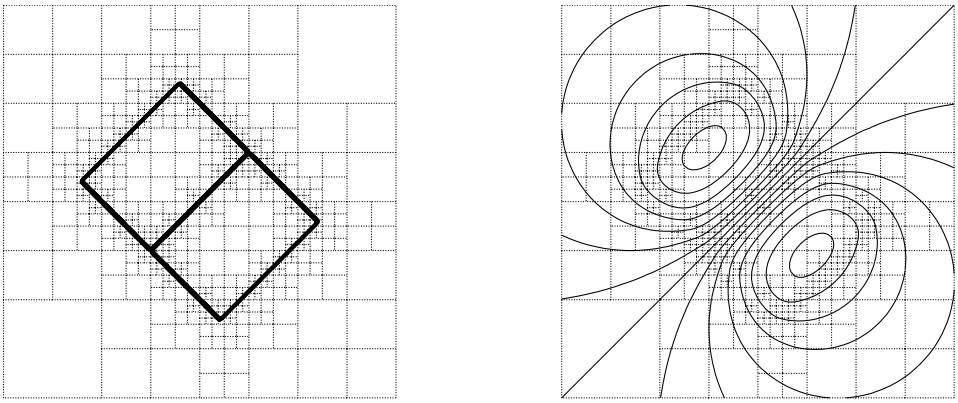

Figure 4: In the left-hand figure, an adaptive discretization of the function defined in example 3 is shown. The right-hand figure superimposes a contour plot of the solution.

Example 4 (Discontinuous distributions).

In Figure 5, we show a rather complicated but rectilinear piecewise constant distribution. Note that the smooth part of the local solution u_s^i is computed exactly on each leaf node and that the evaluation of u on the boundary of each leaf node in Step III is limited only by the multipole tolerance. Step 4 of the algorithm, however, does not provide K^{th} order accuracy, since the exact solution is no longer smooth. On the other hand, it is straightforward to modifiy the algorithm so that full machine precision

can be obtained. Here, we use a nonadaptive refinement with 16×16 leaf nodes and $K = 8$. There are 16,384 grid points and the solver requires 10 seconds.

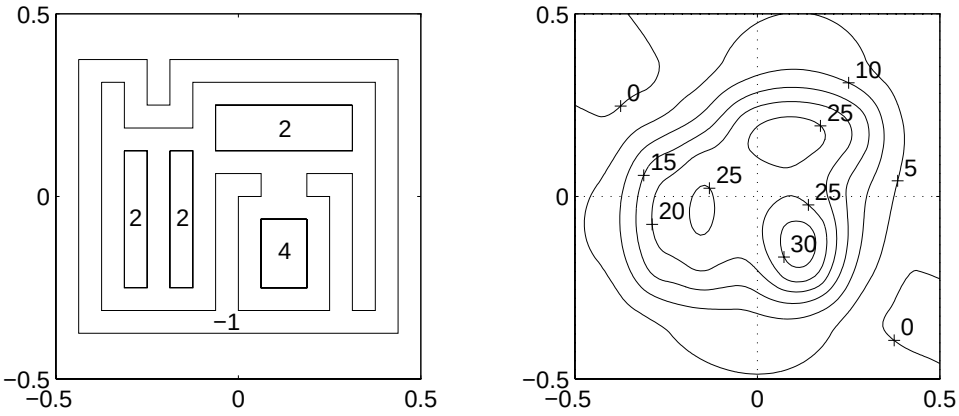

Figure 5: A piecewise constant source distribution (example 4). The left-hand figure shows the supports and values of five different patches. The right-hand figure shows a contour plot of the solution.

Example 5 (Helmholtz decomposition.)

It is well-known that any vector field $\mathbf{u}$ can be written as the sum of two terms

$$\mathbf{u} = \mathbf{u}_p + \mathbf{u}_s, \tag{25}$$

where the irrotational part $\mathbf{u}_p$ has zero curl and the solenoidal part $\mathbf{u}_s$ has zero divergence [19]. This Helmholtz decomposition can be constructed explicitly in terms of $\mathbf{u}$ as follows:

$$\mathbf{u}_p(\mathbf{x}) = \nabla \frac{1}{2\pi} \int_{\mathbf{R}^2} \log|\mathbf{x} - \mathbf{y}| \nabla \cdot \mathbf{u}(\mathbf{y})\, d\mathbf{y}$$
$$\mathbf{u}_s(\mathbf{x}) = \nabla \times \nabla \times \frac{1}{2\pi} \int_{\mathbf{R}^2} \log|\mathbf{x} - \mathbf{y}| \mathbf{u}(\mathbf{y})\, d\mathbf{y}.$$

One of the consequences of having a high-order, adaptive, and well-conditioned Poisson solver is that the projection of $\mathbf{u}$ onto divergence-free fields can be computed stably and accurately. We present only a simple example in the present paper, for which

$$\mathbf{u} = e^{-r^2}(1 - 2x^2 - y,\, x - 2xy). \tag{26}$$

The computation is done on the square $[-4,4]^2$ and the solution is plotted on $[-2,2]^2$. A uniform refinement with 16,384 points was used, requiring 10 seconds of CPU time (Figure 6).

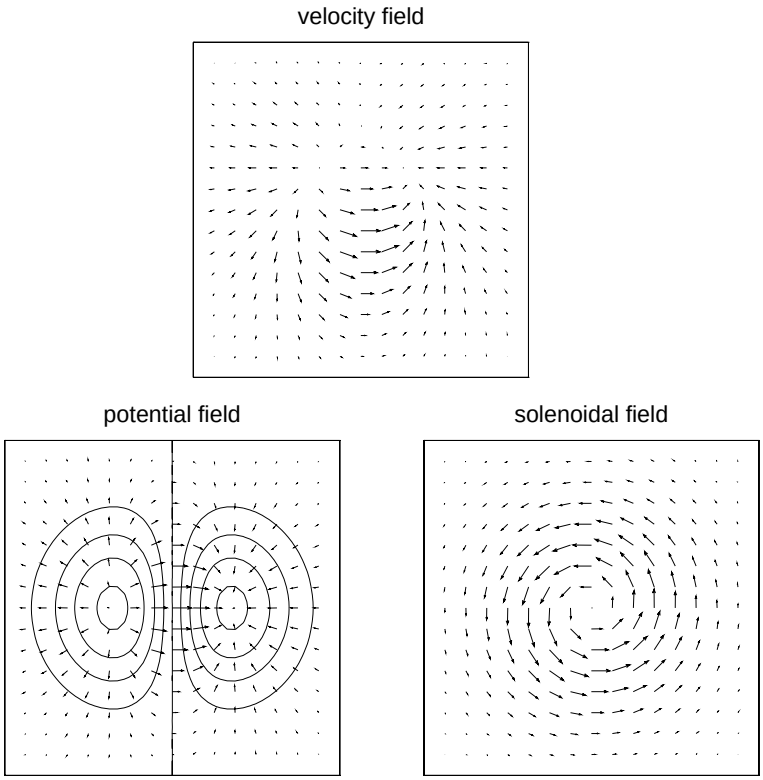

Figure 6: Direct computation of the Helmholtz decomposition

5 Conclusions

We have developed a robust fast solver for the Poisson equation in free space which is adaptive, high-order accurate and direct. Our algorithm requires only that the source distribution have bounded support and that it be smooth on the scale of the leaf nodes of an adaptive quad-tree data structure. The solution procedure is essentially analytic, involving only two kinds of numerical approximation:

1. f is discretized on each leaf node as a high-order Chebyshev expansion, and

2. far field interactions are computed via multipole expansions to within a tolerance of ϵ

We can ignore the second contribution to the error, by simply setting the tolerance ϵ sufficiently low.

The extension of this algorithm to three dimensions is straightforward, although the number of operations per grid point will grow due to the increase in work associated with both the local solver and the FMM. Furthermore, the algorithm can be extended to any problem where the governing Green's function is known, such as the Helmholtz equation, although the FMM required may differ [30].

The most salient feature of the present algorithm, however, is its speed. The sixteenth order accurate implementation requires about 500 floating point operations per grid point. It is only a few times more expensive than a second order nonadaptive method based on Fourier analysis or cyclic reduction. yet it can easily obtain machine precision for both the computed solution and its partial derivatives.

References

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

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

[3] J. D. Baum and R. Löhner (1992), *Numerical design of a passive shock deflector using an adaptive finite element scheme on unstructured grids*, AIAA-92-0448.

[4] J. A. Benek, J. L. Steger, and F. C. Dougherty (1983), *A flexible grid embedding technique with application to the Euler equations*, AIAA-83-1944, 6th Computational Fluid Dynamics Conf., Danvers, Mass.

[5] M.J. BERGER AND J. OLIGER (1984), *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comp. Phys. **53**, pp. 484–512.

[6] M.J. BERGER AND P. COLELLA (1989), *Local adaptive mesh refinement for shock hydrodynamics*, J. Comp. Phys. **53**, pp. 484–512.

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

[8] B. L. BUZBEE, G. H. GOLUB, AND C. W. NIELSON (1970), *On direct methods for solving Poisson's equation*, SIAM J. Numer. Anal. **7**, pp. 627–656.

[9] C. CANUTO, M.Y. HUSSAINI, A. QUARTERONI, AND T.A. ZANG (1988), *Spectral Methods in Fluid Dynamics*, Society for Industrial and Applied Mathematics, Philadelphia.

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

[11] T. CHAN, R. GLOWINSKI, J. PÉRIAUX, AND O. WIDLUND, EDS. (1989), *Domain Decomposition Methods*, Society for Industrial and Applied Mathematics, Philadelphia.

[12] T. CHAN AND B. SMITH (1994), *Domain decomposition and multigrid algorithms for elliptic problems on unstructured meshes*, Electronic Transactions on Numerical Analysis **2**, pp. 171–182.

[13] G. CHESSHIRE AND W.D. HENSHAW (1991), *Composite overlapping meshes for the solution of partial differential equations*, J. Comp. Phys. **90**, pp. 1–64.

[14] D. DEZEEUW AND K. POWELL (1991), *An adaptively refined Cartesian mesh solver for the Euler equations*, AIAA-91-1542, 10th Computational Fluid Dynamics Conf., Honolulu, Hawaii.

[15] D. GOTTLIEB AND S. ORSZAG (1977), *Numerical Analysis of Spectral Methods*, Society for Industrial and Applied Mathematics, Philadelphia.

[16] A. GREENBAUM, L. GREENGARD, AND G. B. MCFADDEN (1993), *Laplace's equation and the Dirichlet-Neumann map in multiply connected domains*, J. Comput. Phys. **105**, pp. 267–278.

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

[18] R. B. GUENTHER AND J. W. LEE (1988), *Partial Differential Equations of Mathematical Physics and Integral Equations,* Prentice-Hall, Englewood Cliffs, New Jersey.

[19] J. D. JACKSON *Classical Electrodynamics*, (Wiley, New York, 1975).

[20] O. D. KELLOGG, *Foundations of Potential Theory*, Dover, New York, 1953.

[21] D. J. MAVRIPLIS AND A. JAMESON (1987), *Multigrid solution of the Euler equations on unstrutured and adaptive meshes*, in Proc. Third Copper Mountain Conf. Multigrid Methods, Lecture Notes in Pure and Applied Mathematics.

[22] A. MAYO (1992), *The rapid evaluation of volume integrals of potential theory on general regions*, J. Comput. Phys. **100**, pp. 236–245.

[23] S. MCCORMICK (1987), *Multigrid Methods*, Society for Industrial and Applied Mathematics, Philadelphia.

[24] A. MCKENNEY, L. GREENGARD, AND A. MAYO (1994), *A fast Poisson solver for complex geometries*, DOE/CMCL Report 94-003, New York University (to appear in J. Comput. Phys.).

[25] A. MIGDAL AND A. LEBEDEV (1995), *Shaped trees and fast Laplace solvers*, preprint.

[26] S. G. MIKHLIN (1957), *Integral Equations*, Pergamon Press, New York.

[27] M. L. MINION (1994), *Two methods for the study of vortex patch evolution on locally refined grids*, PhD thesis, University of California, Berkeley.

[28] A. T. PATERA (1984), *A spectral element method for fluid dynamics: laminar flow in a fluid expansion*, J. Comput. Phys. **54**, pp. 468–488.

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

[30] V. ROKHLIN (1990), *Rapid solution of integral equations of scattering theory in two dimensions*, J. Comput. Phys. **86**, pp. 414–439.

[31] G. RUSSO AND J. STRAIN (1994), *Fast triangulated vortex methods for the 2D Euler equations*, J. Comput. Phys. **111**, pp. 291–323.

[32] L. VOZOVOI, M. ISRAELI, AND A. AVERBUCH (1995), *High order fast elliptic solver in rectangular regions* Technical Report 846, Technion - Israel Institute of Technology.