JOURNAL OF COMPUTATIONAL PHYSICS 99, 251–270 (1992)

Fast Potential Theory. II. Layer Potentials and Discrete Sums*

JOHN STRAIN†

Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, New York 10012

Received March 20, 1990; revised May 24, 1991

We present three new families of fast algorithms for classical potential theory, based on Ewald summation and fast transforms of Gaussians and Fourier series. Ewald summation separates the Green function for a cube into a high-frequency localized part and a rapidly-converging Fourier series. Each part can then be evaluated efficiently with appropriate fast transform algorithms. Our algorithms are naturally suited to the use of Green functions with boundary conditions imposed on the boundary of a cube, rather than free-space Green functions. Our first algorithm evaluates classical layer potentials on the boundary of a d-dimensional domain, with d equal to two or three. The quadrature error is $O(h^m)+\epsilon$, where h is the mesh size on the boundary and m is the order of quadrature used. The algorithm evaluates the discretized potential using N elements at $O(N)$ points in $O(N \log N)$ arithmetic operations. The constant in $O(N \log N)$ depends logarithmically on the desired error tolerance. Our second scheme evaluates a layer potential on the domain itself, with the same accuracy. It produces M^d values using N boundary elements in $O((N+M^d)\log M)$ arithmetic operations. Our third method evaluates a discrete sum of values of the Green function, of the type which occur in particle methods. It attains error ϵ at a cost $O(N^a \log N)$, where $a=2/(1+D/d)$ and D is the Hausdorff dimension of the set where the sources concentrate in the limit $N\to\infty$. Thus it is $O(N\log N)$ when the sources do not cluster too much and close to $O(N \log N)$ in the important practical case when the points are uniformly distributed over a hypersurface. We also sketch an $O(N\log N)$ algorithm based on special functions. Two-dimensional numerical results are presented for all three algorithms. Layer potentials are evaluated to second-order accuracy, in times which exhibit considerable speedups even over a reasonably sophisticated direct calculation. Discrete sum calculations are speeded up astronomically; our algorithm reduces the CPU time required for a calculation with 40,000 points from six months to one hour. © 1992 Academic Press, Inc.

1. INTRODUCTION

A stable and accurate approach to the numerical solution of the Laplace equation

$$-\Delta u = f \qquad \text{in} \quad \Omega \subset \mathbf{R}^d \tag{1}$$

$$\alpha u + \beta \frac{\partial u}{\partial n} = g \qquad \text{on} \quad \Gamma = \partial\Omega \tag{2}$$

is provided by the integral equations of classical potential theory. In this approach, we use a known Green function $K(x, x')$ for a simple region containing Ω to form layer potentials

$$S\mu(x) = \int_\Gamma K(x, x')\,\mu(x')\,dx'$$

$$D\mu(x) = \int_\Gamma \frac{\partial K}{\partial n}(x, x')\,\mu(x')\,dx'$$

and the volume potential

$$Vf(x) = \int_\Omega K(x, x') f(x')\,dx'.$$

Classically, the free-space Green function has been the most popular [21], but when Ω is bounded, we will see that there are significant computational advantages to using the Green function for a cube B containing Ω, with boundary conditions imposed on ∂B. With any Green function, we can seek a solution u as an appropriate linear combination of volume and layer potentials; such an ansatz has the right Laplacian in Ω, and the boundary conditions will be satisfied if we choose the density μ on Γ properly. Usually this requires the solution of a second-kind integral equation on Γ, with an operator combining the nonsingular parts of the double layer potential and the normal derivative of the single layer potential. Numerical solution of these boundary integral equations and the resulting boundary element methods have been extensively studied; see [7, 22, 29], for example.

There are also interface problems occurring in crystal growth [27, 28], in which (2) is replaced by a jump condition

$$\alpha u + \beta \left[\frac{\partial u}{\partial n}\right] = g \qquad \text{on} \quad \Gamma$$

and the problem is augmented by a boundary condition on a cube B containing Ω, say

$$u = u_B \qquad \text{on} \quad B.$$

* Supported by DARPA/AFOSR Contract No. F-49620-87-C-0065 and a NSF Mathematical Sciences Postdoctoral Research Fellowship.

† Current address: Department of Mathematics, Princeton University, Fine Hall, Washington Road, Princeton, NJ 08544.

The Laplace equation is to hold everywhere in $B\backslash\Gamma$. These problems reduce to an integral equation

$$\beta\mu + \alpha S\mu = g \qquad \text{on} \quad \Gamma,$$

with an integral operator which is the single layer potential restricted to Γ.

The advantages to integral equation formulations of these problems are efficiency, stability, and accuracy. The integral equation approach is stable and accurate because integral operators are bounded and even smoothing on appropriate function spaces; thus discrete approximations can have bounded condition numbers as the mesh is refined [19]. (This contrasts with usual finite difference and finite element approximations, which approximate unbounded operators and therefore may have very large condition numbers when the mesh is very fine.) It is efficient because one reduces a d-dimensional problem (to be solved on the d-dimensional domain Ω) to a $(d-1)$-dimensional problem to be solved on Γ. The price one pays for this reduction in dimensionality is loss of sparsity in the linear systems one has to solve. This can be overcome, however, by introducing "fast algorithms" which apply or invert the discretized operators of classical potential theory in essentially optimal amounts of CPU time. This has been done for various special cases in [24, 23] and with some generality in [3, 4].

In this paper, we present new fast algorithms for the approximate evaluation of classical layer potentials formed with the Green function for a square (if $d=2$) or cube (if $d=3$). Our methods use Ewald summation [1, 11] to split the potential into a high-frequency localized part and a low-frequency part with separated variables. The localized part can be evaluated efficiently, because it decays very fast away from the source. The low-frequency part is a rapidly convergent Fourier series, which can be evaluated by non-equidistant fast Fourier transform techniques. Balancing the work and error involved by adjusting the splitting parameter leads to a fast $O(N \log N)$ algorithm.

Our algorithms are based on different principles from earlier fast potential theories such as the fast multipole method [5] or the method of local corrections [2]. These schemes were intended to evaluate discrete convolution sums of the form

$$S_i = \sum_{j \neq i} W_j K(x_i - x_j) \tag{3}$$

which appear in vortex methods [6]. Here $x_j \in \mathbf{R}^d$, $d=2$ or 3, and K is the *free-space* Green function for $-\Delta$ or its derivative. The fast multipole method, for example, is based on multipole expansions (separation of variables), recursion, and data structures. It evaluates (3) in $O(N \log N)$ work if there are N of the points x_j and N values of S_i are desired, within an error tolerance ε specified by the user. The constant in $O(N \log N)$ depends on ε and is quite reasonable in two dimensions, where the method takes advantage of complex analysis.

We also present fast algorithms for evaluating (3) with K the Green function for a box B; these are much simpler than our method for evaluating layer potentials, because they need not address the issue of quadrature error. Our first method for evaluating (3) is very fast, but its cost is not always $O(N \log N)$; however, the deviation from $O(N \log N)$ is small if the points x_j are not too clustered. We sketch a second method which is $O(N \log N)$, but we have not implemented it.

The outline of the paper is as follows. In Section 2, we derive the Ewald summation formulas for evaluation of the Green function for $-\Delta$ with Dirichlet boundary conditions imposed on the boundary of a d-dimensional cube. The only analytical tool necessary is the Poisson summation formula. In Section 3, we discuss quadrature errors in discretizing layer potentials in $d=2$ or 3 dimensions, using Ewald summation, Gaussian quadrature and product integration. In Section 4, we present some background material on subsidiary fast algorithms which we use in this paper. We give brief descriptions of Rokhlin's non-equidistant fast Fourier transform, a scheme for evaluating Fourier series, and the fast Gauss transform. Section 5 presents fast evaluation schemes for evaluating layer potentials both on and off Γ. These schemes evaluate the discretizations developed in Section 3, to arbitrary accuracy and in $O(N \log N)$ time. They also allow us in principle to make the quadrature error arbitrarily high order if Γ and μ are smooth enough. Section 6 discusses how to use Ewald summation methods to evaluate discrete sums like (3). The algorithm we present is optimal only when sources are roughly uniformly distributed, but is usually very fast. We also sketch an optimal algorithm for solving this problem. Section 7 discusses some generalizations of our method—other potentials, other Green functions, other equations—Section 8 presents numerical results for two-dimensional versions of three of the algorithms, and Section 9, our conclusions.

2. THE GREEN FUNCTION FOR A CUBE

This section presents derivations of the Ewald summation formula for the Green function $K(x, x')$ of the Laplace equation

$$\begin{aligned} -\Delta u &= f \qquad \text{in} \quad B = [0,1]^d \\ u &= 0 \qquad \text{on} \quad \partial B \end{aligned} \tag{4}$$

in a d-dimensional cube B, with Dirichlet boundary conditions specified on the boundary ∂B of B. The choice of Dirichlet boundary conditions is arbitrary; we could just as well use Neumann, periodic or mixed (but separable) boundary conditions on ∂B. Our strategy is to relate (4) to the heat equation and use a well-known transformation of the heat kernel for B.

The heat kernel $G(x, x', t)$ for the corresponding parabolic problem

$$\partial_t v = \Delta v \quad \text{in} \quad B \tag{5}$$

$$v = 0 \quad \text{on} \quad \partial B$$

$$v = f \quad \text{at} \quad t = 0 \tag{6}$$

can be found by a d-dimensional Fourier since expansion: the result is

$$G(x, x', t) = 2^d \sum_{k \in N^d} e^{-\pi^2 |k|^2 t} \sin \pi k_1 x_1 \cdots \sin \pi k_d x_d \times \sin \pi k_1 x'_1 \cdots \sin \pi k_d x'_d, \tag{7}$$

where $k = (k_1, ..., k_d)$ runs over the set $\mathbf{N}^d$ of vectors with d strictly positive integer components and $|k|^2 = k_1^2 + \cdots + k_d^2$. This series converges exponentially fast when t is large. The Poisson summation formula [9] (or the method of images [20]) gives the complementary formula

$$G(x, x', t) = (4\pi t)^{-d/2} \sum_{k \in \mathbf{Z}^d} \sum_{\sigma_i = \pm 1} \sigma_1 \cdots \sigma_d e^{-|x - \sigma x' - 2k|^2/4t}, \tag{8}$$

which converges exponentially fast when t is small. (Here $\sigma x' = (\sigma_1 x'_1, ..., \sigma_d x'_d)$ and k runs over the set $\mathbf{Z}^d$ of vectors with d arbitrary integer components.)

We can integrate (5) from $t = 0$ to $t = \infty$ and use (6) to obtain

$$-\Delta \left(\int_0^\infty v(x, t)\, dt \right) = f(x).$$

Thus $u(x) = \int_0^\infty v(x, t)\, dt$ is the solution to the Laplace equation (4). It follows that the Green function $K(x, x')$ for (4) is given by

$$K(x, x') = \int_0^\infty G(x, x', t)\, dt. \tag{9}$$

This translates—into the language of kernels—the operator identity

$$(-\Delta)^{-1} = \int_0^\infty e^{t\Delta}\, dt.$$

This connection between the heat and Laplace equations was used in [20], and doubtless in many other places. The next step in our derivation is essentially equivalent to what is known as "Ewald summation" in the physical literature [1, 11]. Split the time integral (9) at a cutoff time δ, substitute the Fourier series (7) for $G(x, x', t)$ when $t \geqslant \delta$, and replace $G(x, x', t)$ by (8) when $0 \leqslant t \leqslant \delta$. Thus we use each of our two formulas for $G(x, x', t)$ in an interval of the time axis where it converges exponentially fast. The time integrals can be evaluated exactly, and the result is the following formula for $K(x, x')$:

$$K(x, x') = K_F(x, x') + K_L(x, x'), \tag{10}$$

where

$$K_F(x, x') = 2^d \sum_{k \in N^d} \frac{e^{-\pi^2 |k|^2 \delta}}{\pi^2 |k|^2} \sin \pi k_1 x_1 \cdots \sin \pi k_d x_d \times \sin \pi k_1 x'_1 \cdots \sin \pi k_d x'_d \tag{11}$$

and

$$K_L(x, x') = \frac{1}{4\pi^{d/2}} \sum_{k \in \mathbf{Z}^d} \sum_{\sigma_i = \pm 1} \sigma_1 \cdots \sigma_d |x - \sigma x' - 2k|^{2-d} \times \Gamma(d/2 - 1, |x - \sigma x' - 2k|^2/4\delta). \tag{12}$$

Here

$$\Gamma(a, z) = \int_z^\infty e^{-s} s^{a-1}\, ds \tag{13}$$

is the incomplete gamma function. Its properties are discussed in [10]; we only need smoothness away from zero and exponential decay: $|\Gamma(a, z)| \leqslant 2z^{a-1} e^{-z}$ for $a \leqslant 1$ and $z > 0$.

The Fourier series for K_F converges exponentially fast: If we drop all terms in which some k_i is greater than a truncation parameter p, we incur an error E_F bounded by

$$\begin{aligned} |E_F| &\leqslant 2^d d \sum_{k_1=1}^{\infty} \cdots \sum_{k_{d-1}=1}^{\infty} \sum_{k_d=p+1}^{\infty} \frac{e^{-\pi^2 |k|^2 \delta}}{\pi^2 |k|^2} \\ &\leqslant \frac{2^d d}{\pi^2 p^2} \left(\int_0^\infty e^{-\pi^2 k^2 \delta}\, dk \right)^{d-1} \left(\int_p^\infty e^{-\pi^2 k^2 \delta}\, dk \right) \\ &\leqslant \frac{2^d d}{\pi^2 p^2} (4\pi\delta)^{-(d-1)/2} \frac{e^{-\pi^2 p^2 \delta}}{\pi^2 p \delta} \end{aligned}$$

if $\pi^2 p^2 \delta \geqslant 1$. This can be summarized for $d = 2, 3$ as a Fourier series truncation error bound:

$$|E_F| \leqslant \frac{e^{-\pi^2 p^2 \delta}}{50 p^3 \delta^{(d+1)/2}}. \tag{14}$$

The usefulness of K_L, on the other hand, is not so much that the series (12) converges exponentially fast—though it does—but that the sum K_L is exponentially localized in space. Indeed, if x and x' are inside $B = [0, 1]^d$, then one commits an error which is $O(e^{-1/\delta})$ in keeping only 3^d terms of the sum (12), corresponding to the nearest images of B. If either x or x' stays a distance D from ∂B, then K_L is approximated by one term with an error which is $O(e^{-D^2/4\delta})$ as $\delta \to 0$. This term is then bounded by

$$\left| \frac{1}{4\pi^{d/2}} |x - x'|^{2-d} \Gamma(d/2 - 1, |x - x'|^2/4\delta) \right|$$
$$\leqslant \frac{2^{3-d}}{\pi^{d/2}} \frac{\delta^{2-d/2}}{|x - x'|^2} e^{-|x-x'|^2/4\delta}.$$

This is exponentially small as soon as $|x - x'| \geqslant O(\sqrt{\delta})$ in two dimensions and $|x - x'| \geqslant O(\sqrt{\delta} \, |\log \delta|)$ in three dimensions.

Finally, we explain why Ewald summation is useful. If we had computed $K(x, x')$ by a direct Fourier expansion, we would have found

$$K(x, x') = 2^d \sum_{k \in N^d}^{\infty} \frac{1}{\pi^2 |k|^2} \sin \pi k_1 x_1 \cdots \sin \pi k_d x_d$$
$$\times \sin \pi k_1 x'_1 \cdots \sin \pi k_d x'_d. \qquad (15)$$

This series either diverges (if $d \geqslant 3$) or converges slowly (if $d \leqslant 2$), so it is almost useless for evaluating the kernel. This is because we are expressing high-frequency information and low-frequency information alike as a Fourier series, so we have to include many terms. If we had computed $K(x, x')$ by the method of images, we would also have gotten a useless expansion, because we would be trying to convey global information by point evaluation. Instead, we have constructed formulas which give $K(x, x')$ as a sum of two series, K_F and K_L. The local information is carried by the rapidly decaying local part K_L and the global low-frequency information is expressed in the Fourier series for K_F. The cutoff δ indexes a one-parameter family of formulas for $K(x, x')$, the Fourier series appears when $\delta = 0$, and the method of images sum occurs in the limit $\delta \to \infty$.

Our fast algorithms are based on this splitting of $K(x, x')$. Global information is encoded in the rapidly converging Fourier series for K_F, which can be evaluated rapidly because the variables are separated. Local high-frequency information is carried in K_L, and K_L decays very rapidly away from its singularity—it decays like a Gaussian with small variance. Hence convolution with K_L is an almost local operator, which can be applied rapidly.

3. DISCRETIZATION OF LAYER POTENTIALS

We now have the basic Ewald summation formulas we need to evaluate layer and volume potentials. For simplicity, we deal in detail only with the single layer potential, in dimensions $d = 2$ and 3. We describe how to evaluate $S\mu$ on and off Γ to accuracy $O(h^m)$ in the size h of the mesh on Γ. We are mostly interested in the cases $m = 2$ and $m = 4$.

Our basic approach is as follows. We allow two types of error. The first, quadrature error, occurs with all approximate evaluation of integrals. It is $O(h^m)$ as the size h of the mesh on Γ decreases, with a constant in $O(h^m)$ which is allowed to depend on derivatives of the density μ up to some order M, possibly larger than m. This error has to do with the discretization error in evaluating $S\mu$ with Ewald summation, independent of the speed with which $S\mu$ is evaluated. It does, however, depend on the splitting parameter δ and therefore on the number of terms p kept in the Fourier series representation of $S_F\mu$. The second type of error is the price we pay for the fast evaluation scheme; it is $O(\varepsilon)$, where ε is a user-specified tolerance and the constant in $O(\varepsilon)$ is *not* allowed to depend on derivatives of μ or Γ.

It is unrealistic in many applications to assume that Γ is known exactly, say as a smooth parametrized surface. In moving boundary problems, for example, we have to approximate Γ by an object with finitely many degrees of freedom. Thus our code has been written to operate on Γ and μ given as a union of elements Γ_j, with maximum side length bounded by h. In two dimensions, Γ is a union of line segments if $m = 2$ and a union of cubic curves if $m = 4$, say cubic Hermite interpolants to points on Γ and the derivatives of Γ at those points, for example. In three dimensions, we take Γ to be a union of quadrilaterals if $m = 2$ and if $m = 4$ we use images of rectangles under bicubics, say Hermite interpolants or splines. See [14] for background on surface representation and [29] for the effect on finite element methods. On each element, μ will be given as a polynomial of degree $m - 1$, to accuracy $O(h^m)$ if Γ and μ are of class C^m. (We use quadrilateral elements in three dimensions, rather than the more popular and versatile triangular elements, mainly for convenience of exposition. We need to take advantage of Gaussian integration rules with their superalgebraic convergence. Such rules do exist for triangles, for example conical Gaussian rules, so our analysis can presumably be extended to triangular elements.) We can treat many more general situations using the techniques reported here, but have preferred simplicity of exposition over generality for the time being. We aim to keep the discussion on a concrete and practical level, with a minimum of abstraction. If Γ and μ are smooth, we have then

$$S\mu(x) = \sum_{j=1}^{N} \int_{\Gamma_j} K(x, x') \, \mu(x') \, dx' + O(h^m), \qquad (16)$$

where $O(h^m)$ depends on derivatives of Γ and μ of order up to m; see [22, 29] for error analysis. We now introduce the Ewald splitting

$$S\mu(x) = S_F\mu(x) + S_L\mu(x)$$
$$= \int_\Gamma K_F(x, x')\,\mu(x')\,dx' + \int_\Gamma K_L(x, x')\,\mu(x')\,dx',$$

where S_F is a rapidly converging Fourier series and S_L is highly localized. We consider discretization of $S_F\mu$ and $S_L\mu$ separately, and also separate the evaluation of $S_L\mu$ on Γ from its evaluation at an arbitrary point $x \in B$, which is not known to lie on Γ. These two situations require completely different strategies.

3.1. *The Fourier Part*

First we consider the evaluation of $S_F\mu$. From (10), we have

$$S_F\mu(x) = \int_\Gamma 2^d \sum_{k \in N^d} \frac{e^{-\pi^2|k|^2\delta}}{\pi^2\,|k|^2} \sin \pi k_1 x_1$$
$$\cdots \sin \pi k_d x_d \sin \pi k_1 x'_1 \cdots \sin \pi k_d x'_d \mu(x')\,dx'$$
$$= 2^d \sum_{k \in N^d} \frac{e^{-\pi^2|k|^2\delta}}{\pi^2\,|k|^2} \hat{\mu}(k) \sin \pi k_1 x_1 \cdots \sin \pi k_d x_d,$$

where the Fourier coefficients $\hat{\mu}(k)$ of the measure on B with density μ on Γ are given by

$$\hat{\mu}(k) = \int_\Gamma \sin \pi k_1 x'_1 \cdots \sin \pi k_d x'_d \mu(x')\,dx'.$$

The error in truncation the Fourier series for $S_F\mu$ after terms with $k_i = p$ is bounded by (Section 2)

$$|E_F| \leqslant \frac{e^{-\pi^2 p^2 \delta}}{50 p^3 \delta^{(d+1)/2}} |\mu|_1,$$

where $|\mu|_1 = \int_\Gamma |\mu(x')|\,dx'$ and $\pi^2 p^2 \delta \geqslant 1$. If this error is to be bounded, it is clearly necessary to have $p^2\delta$ bounded away from zero as $\delta \to 0$ and $p \to \infty$. If $d = 2$ this condition is sufficient; in $d =$ three dimensions, $p^2\delta$ has to increase logarithmically as $\delta \to 0$ and $p \to \infty$, in order to balance an additional power of δ in the denominator. In two dimensions, for example, this bound is less than $10^{-6}\,|\mu|_1$ when $p^2\delta \geqslant 1$ and less than $10^{-11}\,|\mu|_1$ when $p^2\delta \geqslant 2$.

Next, consider the error in evaluating $\hat{\mu}(k)$ by product q-point Gauss–Legendre quadrature over each element Γ_j. Generally, the error in evaluating the integral $\int_0^1 f(s)\,ds$ by such a rule is bounded by (see page 98 of [8])

$$|E| = \left| \int_0^1 f(s)\,ds - \sum_{j=1}^{q} f(s_j)\,w_j \right|$$
$$\leqslant \frac{q!^4}{(2q+1)\,2q!^3} |f^{(2q)}|_\infty$$
$$= O\left(\frac{e}{8q}\right)^{2q} |D^{2q} f|_\infty,$$

where s_j are the points and w_j are the weights of the Gauss–Legendre formula, D denotes differentiation, and $|f|_\infty = \max |f(s)|$. The second equality follows from Stirling's formula.

Our task now is to split this estimate, with f equal to a product of sines times μ, into $O(h^m)$ and $O(\varepsilon)$ parts, as described above. By Leibniz' rule, we have

$$D^{2q} f(x') = \sum_{j=0}^{2q} \binom{j}{2q} D^{2q-j}$$
$$\times (\sin \pi k_1 x'_1 \cdots \sin \pi k_d x'_d)\, D^j \mu(x'),$$

where D denotes differentiation. The two dominant terms in this estimate are the endpoints, where all the derivatives go onto one factor or the other. (The intermediate terms can be bounded in terms of the endpoints by interpolation estimates for intermediate derivatives and Hölder's inequality.) Thus the $2q$th derivative of f can be estimated by

$$O(\pi d p h)^{2q}\,|\mu|_\infty + O(h^{2q}),$$

where the first comes from differentiating the sines and the second term depends on derivatives of μ up to order $2q$ but is independent of p. We will choose p and q depending on h to make the first term less than ε and the second term $O(h^m)$, in Section 5.

We then will have each Fourier coefficient $\hat{\mu}(k)$ for $1 \leqslant k_i \leqslant p$, with error E_k. Thus evaluating $S_F\mu$ gives an error bounded by

$$|E| \leqslant \max |E_k| \sum_{k \in N^d} \frac{e^{\pi^2|k|^2\delta}}{\pi^2\,|k|^2}$$
$$\leqslant \max |E_k|\, C_d \int_0^\infty e^{-\pi^2 r^2 \delta} r^{d-3}\,dr$$
$$\leqslant \max |E_k|\, C_d \delta^{1-d/2}.$$

3.2. *The Local Part Evaluated on Γ*

We turn now to the evaluation of the local part

$$S_L\mu(x) = \int_\Gamma K_L(x, x')\, \mu(x')\, dx'$$

for $x \in \Gamma$. More care is required in this case, because K_L is infinite when $x = x'$. But we can use the known form of the singular kernel to transform the integral in a manner convenient for evaluation. Write

$$K_L(x, x') = \int_0^\delta K(x, x', t)\, dt$$
$$= \int_0^\delta \frac{e^{-|x-x'|^2/4t}}{(4\pi t)^{d/2}}\, dt + O(e^{-D^2/4\delta}).$$

Here we assume, for simplicity, that $\operatorname{dist}(\Gamma, \partial B) \geqslant D > 0$ and we can thus drop images when δ is small. The images are nonsingular, and therefore represent only a notational complication. Thus we have

$$S_L\mu(x) = \int_0^\delta \int_\Gamma \frac{e^{-|x-x'|^2/4t}}{(4\pi t)^{d/2}}\, \mu(x')\, dx'\, dt$$
$$+ O(e^{-D^2/4\delta}).$$

This is a single layer *heat* potential with density μ independent of time. Write, for convenience,

$$g(x, t) = \int_\Gamma \frac{e^{-|x-x'|^2/4t}}{(4\pi t)^{(d-1)/2}}\, \mu(x')\, dx'$$
$$S_L\mu(x) = \int_0^\delta \frac{1}{\sqrt{4\pi t}}\, g(x, t)\, dt + O(e^{-D^2/4\delta}).$$

Then it can easily be shown [28] that

$$g(x, t) \to \mu(x) \qquad \text{as} \quad t \to 0 \text{ for } x \in \Gamma$$

and g is a smooth function of t. Hence the only remaining singularity is the square-root singularity in the time integral, and this is *independent* of x. Thus we can make a product integration formula [18]

$$\int_0^\delta \frac{1}{\sqrt{4\pi t}}\, g(x, t)\, dt = \sqrt{\delta/\pi} \sum_{j=0}^{n} w_j\, g(x, \tau_j)$$
$$+ O(\delta^{n+3/2}),$$

with $\tau_j = j\delta/n$, with weights w_j determined by requiring the formula to be exact whenever g is a polynomial in t of degree $\leqslant n$. Thus we have, from the definition of g,

$$S_L\mu(x) = \sqrt{\delta/\pi}\, w_0\mu(x) + \sqrt{\delta/\pi} \sum_{j=1}^{n} \frac{w_j}{(4\pi\tau_j)^{(d-1)/2}}$$
$$\times G_{4\tau_j}\mu(x) + O(e^{-D^2/4\delta}),$$

where the Gauss transform $G_\tau\mu(x)$ is defined by

$$G_\tau\mu(x) = \int_\Gamma e^{-|x-x'|^2/\tau}\, \mu(x')\, dx'.$$

Thus we need to evaluate $n-1$ Gauss transforms with $\tau = O(\delta)$. Given $G_\tau\mu$ with error bounded by E, we get $S_L\mu$ on Γ with error bounded by

$$|E_L| \leqslant C_1\delta^{1-d/2}E + O(\delta^{n+3/2}) + O(e^{-D^2/4\delta}),$$

where C_1 is a constant of order unity.

Now consider the evaluation of the Gauss transform. We have, as in (16),

$$G_\tau\mu(x) = \sum_{j=1}^{N} \int_{\Gamma_j} e^{-|x-x'|^2/\tau}\, \mu(x')\, dx' + O(h^m)$$

for mth-order elements and interpolation. We approximate each integral over Γ_j by product Gauss–Legendre quadrature using q points per dimension. To estimate the quadrature error, we need the $2q$th derivative of the integrand. A calculation with Hermite functions shows that roughly, we can estimate such derivatives by

$$|f^{(2q)}|_\infty \leqslant C_0 \left(\frac{h}{\sqrt{\tau}}\right)^{2q} \sqrt{2q!}.$$

With Stirling's formula, we find that the quadrature error satisfies

$$|E| \leqslant O\left(\frac{e}{32q}\frac{h^2}{\tau}\right)^q |\mu|_\infty.$$

Note that this estimate is essentially the same as (14) since $\tau \geqslant \delta/n$ and $p^2\delta \geqslant$ constant. Finally, the quadrature error involved in evaluating $S_L\mu$ on Γ is thus bounded roughly by

$$|E_L| \leqslant C\delta^{1-d/2} O\left(\frac{e}{32q}\frac{nh^2}{\delta}\right)^q |\mu|_\infty + O(\delta^{n+3/2}).$$

As in the calculation of the Fourier coefficients, we have not yet enough information to use this estimate, so we will return to the evaluation of $S_L\mu$ on Γ after introducing the fast Gauss transform in Section 4 and balancing the work estimates in Section 5.

3.3. *The Local Part Evaluated Off Γ*

When x is not known to lie on Γ, the integrand in the integral

$$S_L\mu(x) = \int_\Gamma K_L(x, x')\,\mu(x')\,dx'$$

may not be singular. It is smooth when x is not on Γ, but blows up when x approaches Γ. Paradoxically, this possible *lack* of a singularity is quite troublesome when evaluating $S_L\mu$. This is because we cannot use a product integration formula in time which is independent of x, and therefore cannot express $S_L\mu$ as a sum of Gauss transforms.

We use instead a *spatial* product integration method to evaluate $S_L\mu$ off Γ. This is particularly convenient in the second-order case in two dimensions where the integrals involved are fairly straightforward, so we give the details only in this case. When higher order accuracy is desired, product integration becomes difficult; however, the approach suggested in [19] is an attractive alternative. A local expansion as in [26, 13] could also be used effectively here, because K_L decays rapidly away from its singularity; there is no far field.

In two dimensions,

$$K_L(x, x') = \frac{1}{4\pi}\,\Gamma\left(0, \frac{|x-x'|^2}{4\delta}\right) + O(e^{-D^2/4\delta})$$

and Γ_j is the line segment connecting x_j and x_{j+1} if $m = 2$. Thus

$$\int_{\Gamma_j} K_L(x, x')\,\mu(x')\,dx'$$
$$= \frac{1}{4\pi}\int_0^1 \Gamma\left(0, \frac{|x - tx_{j+1} - (1-t)\,x_j|^2}{4\delta}\right)$$
$$\times (t\mu_{j+1} + (1-t)\,\mu_j)\,|x_{j+1} - x_j|\,dt.$$

We have

$$\Gamma(0, z) = -\log(z) + F_1(z),$$

where $F_1(z)$ is entire. Thus we integrate the logarithmic part of the kernel exactly over each line segment and apply Gauss–Legendre quadrature to the remaining integral involving $F_1(|x-x'|^2/4\delta)\,\mu(x')$. The integrand of the F_1 integral is an analytic function *scaled* by $1/\sqrt{\delta}$, so the $2q$th derivative grows no worse than $2q!\delta^{-q}$. Hence the error estimate for integrating the F_1 term looks no worse than

$$|E_1| \leqslant C\frac{q!^4}{2q!^3}\frac{2q!\,h^{2q}}{\delta^q} = O\left(\frac{h}{4\sqrt{\delta}}\right)^{2q}$$

by Stirling's formula. This is therefore the complete error involved in evaluating $S_L\mu$ in this case. Further analysis will have to be postponed until we know how h is related to δ.

In three dimensions, a similar analysis applies. Only the details of evaluating the singular term exactly are different.

4. BACKGROUND MATERIAL

In this section, we describe three fast algorithms which we will use in the main body of this paper. First, we describe an unpublished algorithm suggested by Rokhlin [25], which evaluates discrete Fourier coefficients given function values at arbitrary points. Then we describe a simple method for evaluating a Fourier series at an arbitrary collection of points. Finally, we describe the fast Gauss transform [12] which evaluates a convolution sum of d-dimensional Gaussians. All three schemes are much faster than direct evaluation of the corresponding sums, as soon as problems of any reasonable size need to be solved.

4.1. *The Non-Equidistant Fast Fourier Transform*

Rokhlin's algorithm evaluates the sum

$$\hat{f}(k) = \sum_{j=1}^{N} e^{ia_jk} f_j \tag{17}$$

for $k = 0, 1, 2, ..., p$, given N points $a_j \in [-\pi, \pi]$ and N complex numbers f_j. Direct evaluation costs $O(Np)$ work, and the usual fast Fourier transform can be used only when a_j are equispaced. Rokhlin's algorithm evaluates this sum with accuracy εF in $O((N+p)\log p)$ work, with a constant depending on the user-specified precision ε and $F = \sum_{j=1}^N |f_j|$.

The evaluation of (17) amounts to finding the Fourier coefficients of the periodic distribution f defined by

$$f(x) = 2\pi \sum_{j=1}^{N} \delta(x - a_j) f_j \tag{18}$$

on $[-\pi, \pi]$. A natural approach, if f were a smooth function, would be to evaluate f on an equidistant mesh and apply a standard fast Fourier transform. This is impossible, of course, because we cannot evaluate $\delta(x - a_j)$ at a point. Thus we smooth each point mass into a Gaussian, apply the FFT, and undo the smoothing.

We define the smoothed function g approximating f by requiring its Fourier coefficients $\hat{g}(k)$ to be given by

$$\hat{g}(k) = e^{-\delta k^2}\hat{f}(k)$$
$$= \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{ikx} g(x)\,dx.$$

Thus

$$g(x) = \sum_{-\infty}^{\infty} e^{-\delta k^2} \hat{f}(k)\, e^{-ikx}. \tag{19}$$

Since

$$\hat{f}(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{ikx} f(x)\, dx,$$

we have

$$g(x) = \frac{1}{2\pi} \int_{-\pi}^{\pi} K(x-x') f(x')\, dx'$$

where

$$K(x) = \sum_{-\infty}^{\infty} e^{-\delta k^2} e^{ikx} = \sqrt{\pi/\delta} \sum_{-\infty}^{\infty} e^{-(x-2k\pi)^2/4\delta}.$$

The second equality is a well-known consequence of the Poisson summation formula [9]. From the definition of f, we have

$$g(x) = \sqrt{\pi/\delta} \sum_{j=1}^{N} f_j \sum_{-\infty}^{\infty} e^{-(x-a_j-2k\pi)^2/4\delta}. \tag{20}$$

Since δ will be small, we need only a few terms of the infinite sum over k in (20): the error in keeping only three terms is bounded by $\sqrt{\pi/\delta}\, e^{-\pi^2/\delta}$ as long as $|x| \leqslant \pi$.

Next we evaluate g on the equidistant grid $x = jh$ with $-q \leqslant j \leqslant q$, $h = \pi/q$. If we evaluate three Gaussians for each j at each grid point, we do $O(Nq)$ work, and we expect $q \geqslant p$, so this costs too much. The rapid spatial decay of the Gaussian, however, means that we need to evaluate the Gaussian centered at a_j only for $|x - a_j| \leqslant R$, where the range R depends on δ. The error in this truncation is bounded by $3F\sqrt{\pi/\delta}\, e^{-R^2/4\delta}$. If $R = Lh$, this evaluation will cost $O(LN)$ work. We now have the values $g(jh)$ on an equidistant grid, so we can use the standard FFT to evaluate the discrete Fourier coefficients

$$\tilde{g}(k) = \sum_{-q}^{q} e^{2\pi ikj/2q} g(jh) \tag{21}$$

in $O(q \log q)$ work. However, we really wanted the continuous Fourier coefficients

$$\hat{g}(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{ikx} g(x)\, dx,$$

not the trapezoidal sums (21). Fortunately, the explicit formula (19) allows us to bound the quadrature error in replacing continuous by discrete Fourier coefficients. A Fourier series calculation described in [8] gives

$$|\hat{g}(k) - \tilde{g}(k)| \leqslant 2F(e^{-\delta(2q-p)^2} + O(e^{-\delta(4q-p)^2}))$$

if $|k| \leqslant p$.

Finally, we can evaluate $\hat{f}(k)$ by unsmoothing:

$$\hat{f}(k) = e^{\delta k^2} \hat{g}(k), \qquad 0 \leqslant k \leqslant p.$$

This will multiply any errors (including roundoff errors) in the computation of g by a factor $e^{\delta k^2} \leqslant e^{\delta p^2}$. The whole algorithm will therefore be unstable unless $\delta p^2 \leqslant c$, where c is a constant depending only on the precision desired. Thus we tentatively set $\delta = c/p^2$.

Now we must determine the parameters c and q to achieve the desired accuracy and efficiency. The error in $\hat{f}(k)$ for $0 \leqslant k \leqslant p$ will be bounded by εF if the following three conditions are satisfied:

$$\sqrt{\pi/\delta}\, e^{-\pi^2/\delta} F \leqslant \frac{\varepsilon e^{-c}}{2q} F$$

$$\sqrt{\pi/\delta}\, e^{-R^2/4\delta} 3F \leqslant \frac{\varepsilon e^{-c}}{2q} F$$

$$2Fe^{-\delta(2q-p)^2} \leqslant \varepsilon e^{-c} F.$$

The first inequality comes from truncating the infinite sum of Gaussians after three terms, the second from allowing a_j to influence only points x within a range R, and the third inequality requires the quadrature error in evaluating $\hat{g}(k)$ by the trapezoidal rule to be small. The total work required by the algorithm is

$$O(LN) + O(q \log q) + O(p).$$

The first term comes from evaluating Gaussians, the second from applying the FFT to g and the third from evaluating $\hat{f}$.

First, we require $c \leqslant 2 \log 10$; thus the final processing of $\hat{f}$ can lose no more than two decimal digits. This is a compromise between speed and roundoff error. The quadrature error bound will hold if

$$e^{c(1-(2q/p-1)^2)} = \varepsilon$$

or $c = \log \varepsilon/(1-(2q/p-1)^2)$. The requirement $c \leqslant 2 \log 10$ gives a lower bound for q/p;

$$q/p \geqslant \max(2, \tfrac{1}{2} + \tfrac{1}{2}\sqrt{1 - \tfrac{1}{2}\log \varepsilon/\log 10}).$$

For $\varepsilon \geqslant 10^{-16}$, this reduces to $q/p \geqslant 2$. Thus c is determined given q/p. Now let $\varepsilon' = \varepsilon e^{-c}/10q$ (given q). We choose $R = Lh = L\pi/q$ so that

$$(1/\sqrt{\delta})\, e^{-R^2/4\delta} \leqslant \varepsilon'.$$

Thus

$$R = \sqrt{-4\delta \log(\sqrt{\delta\varepsilon'})}$$
$$= O(h \log p).$$

We choose q so that $R \leqslant \pi$; thus only three images need be kept, and the first requirement is satisfied as well. The total work estimate now looks like

$$W = O(N \log p + p \log p + p)$$
$$= O((N+p) \log p).$$

This completes our description of Rokhlin's non-equidistant fast Fourier transform.

In practice, the algorithm performs extremely well. We wrote a Fortran code implementing the algorithm and tested it with a set of N points a_j chosen from a uniform distribution on $[-\pi, \pi]$ and f_j chosen from a uniform distribution on $[0, 1]$. Table I shows the parameters, errors, and times (T_f) obtained with $\varepsilon = 10^{-7}$ and $N = p = 16$, 32, 64, 128, 256, 512, 1024, 2048. The times T_d given for direct evaluation are extrapolated from the time required for direct evaluation at 80 points for the larger runs, and the column headed T_{FFT} shows the time required to execute one standard FFT of size $2p$. Thus we see that it costs only five or six times as much to evaluate Fourier coefficients with arbitrary points as it does to evaluate them with equidistant points. The fast algorithm is much faster than direct evaluation, breaking even at only about 16 points and coefficients, and the error in the fast evaluation scheme is always considerably smaller than the error bound.

TABLE I

Results for the Non-equidistant FFT, with CPU Times on a Sun-4 Workstation

$N = p$	q	δ	L	T_f	T_d	T_{FFT}	Error/F
16	32	7.0−3	8	0.03	0.03	0.004	3.0−8
32	64	1.8−3	8	0.05	0.11	0.008	6.2−8
64	128	4.4−4	8	0.11	0.38	0.018	6.2−11
128	256	1.1−4	8	0.24	1.54	0.035	8.8−12
256	512	2.7−5	8	0.45	6.22	0.077	7.3−12
512	1024	6.8−6	9	0.95	25.0	0.16	5.9−14
1024	2048	1.7−6	9	1.93	100.3	0.34	2.9−14
2048	4096	4.3−7	9	4.01	401.4	0.67	3.0−14

4.2. *Evaluation of Fourier Series*

Now let us consider the inverse problem: Given $\hat{f}(k)$ for $k = -p, -p+1, ..., p-1, p$, and N arbitrary points $a_j \in [-\pi, \pi]$, evaluate the trigonometric polynomial

$$f(x) = \sum_{-p}^{p} e^{ikx} \hat{f}(k)$$

at the points a_j in $O((N+p) \log p)$ work, with error bounded by $\varepsilon\hat{F}$, where $\hat{F} = \sum |\hat{f}(k)|$.

Our approach is straightforward; we zero-pad the coefficients $\hat{f}_j$ to length $2q$ and perform a standard FFT of length $2q$ to obtain $2q$ values

$$f(jh) = \sum_{-p}^{p} e^{ijkh} \hat{f}(k)$$

on a fine grid on $[-\pi, \pi]$ with step size $h = \pi/q$. Then we interpolate between grid points to obtain the desired values $f(a_j)$ for $1 \leqslant j \leqslant N$, with interpolation error $\leqslant \varepsilon\hat{F}$. It turns out that we can guarantee such accuracy, for completely arbitrary Fourier coefficients $\hat{f}(k)$, by taking q fairly large compared to p and using fairly high-order interpolation. Thus the algorithm turns out to be considerably more expensive than a standard FFT when $\hat{f}$ are randomly chosen and high accuracy is desired. Of course, in most practical situations, $\hat{f}$ are approximations to the Fourier coefficients of a function and in that case much less work is required; an example will occur in Section 5.1.

Let $q = np$; we will choose n and the order of interpolation $2k-1$ to make the interpolation error $\leqslant \varepsilon\hat{F}$. In general, the error in equidistant polynomial interpolation of a function f at a point $x \in [0, 1]$, by polynomials of degree $2k-1$, is bounded by

$$|E_{2k-1}| \leqslant \frac{|f^{(2k)}|_\infty}{(2k)!}\, \omega_{2k-1}(x)$$

where $|g|_\infty$ denotes the max-norm of g and

$$\omega_n(x) = (x - 0 \cdot h)(x - 1 \cdot h) \cdots (x - n \cdot h).$$

We use interpolation only on the center interval $kh \leqslant x \leqslant (k+1)h$; then

$$|\omega_{2k-1}(x)| \leqslant \tfrac{1}{2} h^{2k} (k!)^2.$$

By Stirling's formula,

$$\frac{(k!)^2}{(2k)!} \approx \frac{\sqrt{\pi k}}{2^{2k}},$$

TABLE II

Error Bounds for Fourier Series Evaluation, Using a Mesh Ratio n and Polynomial Interpolation of Degree $2k-1$

$2k-1$	$n=3$	$n=4$	$n=6$	$n=8$	$n=12$	$n=16$
1	0.27E+00	0.15E+00	0.69E−01	0.39E−01	0.17E−01	0.96E−02
3	0.10E+00	0.32E−01	0.63E−02	0.20E−02	0.39E−03	0.12E−03
5	0.33E−01	0.59E−02	0.52E−03	0.92E−04	0.80E−05	0.14E−05
7	0.10E−01	0.10E−02	0.40E−04	0.40E−05	0.16E−06	0.16E−07
9	0.31E−02	0.18E−03	0.31E−05	0.17E−06	0.30E−08	0.17E−09
11	0.94E−03	0.30E−04	0.23E−06	0.73E−08	0.56E−10	0.18E−11
13	0.28E−03	0.50E−05	0.17E−05	0.30E−09	0.10E−11	0.18E−13
15	0.81E−04	0.81E−06	0.12E−08	0.12E−10	0.19E−13	0.19E−15
17	0.24E−04	0.13E−06	0.90E−10	0.51E−12	0.34E−15	0.19E−17
19	0.68E−05	0.22E−07	0.65E−11	0.21E−13	0.62E−17	0.20E−19

and in general the best we can say about the $2k$th derivative of f is

$$|f^{(2k)}|_\infty \leqslant p^{2k}\hat{F},$$

so

$$|E_{2k-1}| \leqslant \sqrt{k}\,\hat{F}\left(\frac{hp}{2}\right)^{2k} = \sqrt{k}\,\hat{F}\left(\frac{\pi}{2n}\right)^{2k},$$

since $h=\pi/q=\pi/np$. Table II shows this error bound (with the factor $\hat{F}$ omitted) as a function of k and n. Single precision accuracy ($\varepsilon=10^{-7}$) requires 19th-degree interpolation with $n=4$, 11th with $n=8$, and 7th with $n=16$. Double precision accuracy ($\varepsilon=10^{-13}$) requires 19th-degree interpolation with $n=8$ and 13th-degree with $n=16$. In practice, we found interpolation of degree higher than about 20 to lead to substantial rounding errors. We evaluated the interpolating polynomial by Aitken's algorithm.

Given n, one can find k by requiring $(\pi/2n)^{2k}\leqslant\varepsilon$ or $2k\geqslant -\log\varepsilon/\log(2n/\pi)$. Table III shows numerical results obtained from testing the algorithm on randomly generated Fourier series coefficients and points of evaluation as in Section 4.1, with error tolerance $\varepsilon=10^{-5}$, $n=8$, and 7th-degree interpolation. The error bound is quite sharp, and even in this fairly difficult case, the algorithm breaks even at only about 32 points and coefficients. These choices of n and k are not optimal, of course; in practice the choice of n and k will be a trade-off between speed and memory, especially for multidimensional problems.

TABLE III

Times and Errors for Evaluating Randomly Generated Fourier Series with $\varepsilon=10^{-5}$, $n=8$, and 7th-Degree Interpolation

$N=p$	Fast	Direct	Error/$\hat{F}$
16	0.07	0.03	1.5−6
32	0.11	0.12	8.6−7
64	0.25	0.49	9.7−7
128	0.48	2.0	9.6−7
256	1.0	7.9	1.1−6
512	2.1	31.2	1.7−7
1024	4.2	126	2.5−7
2048	8.4	502	1.2−7

Finally, we observe that both the scheme presented in this section and Rokhlin's algorithm generalize immediately to higher-dimensional problems. They are not tensor products as are standard FFTs, but the generalization is straightforward nonetheless. In the numerical calculations of this paper, we use schemes which evaluate two-dimensional Fourier sine coefficients and Fourier sine series with non-equidistant points; these are also straightforward generalizations of the schemes presented in this section.

4.3. *The Fast Gauss Transform*

In this section, we very briefly summarize the fast Gauss transform presented in [12]. Consider the evaluation of the d-dimensional Gaussian sum

$$f(x)=\sum_{j=1}^{N} f_j e^{-|x-s_j|^2/\delta} \tag{22}$$

at M points $x=t_i\in B=[0,1]^d$. Here s_j are N given points in B, f_j are N given real or complex numbers, and $|x|^2=x_1^2+\cdots+x_d^2$. Clearly direct evaluation takes $O(NM)$ work. The fast Gauss transform requires $O(N+M+\delta^{-d/2})$ work to evaluate (22) to precision εF with $F=\sum|f_j|$; the constant in $O(N+M+\delta^{-d/2})$ depends only on ε. In practice, the algorithm achieves a tremendous speedup over direct evaluation when M and N are large and δ is not too small. When δ is very small, the fast Gauss transform reduces to a structured and truncated direct evaluation scheme which takes advantage of the short range of influence of each source s_j.

The basic approach combines separation of variables with a divide and conquer approach, as in the fast multipole method [5]. We divide the box B into $O(\delta^{-d/2})$ boxes of side $O(\sqrt{\delta})$ and sort the sources s_j and targets t_i into boxes by spatial location. The influence of all sources in a given box can be combined into a single Hermite expansion about the center of the box. Each Hermite expansion influences a fixed number of boxes within range $O(\sqrt{\delta})$, by adding to the Taylor expansion of f about the center of each target box. Finally, the Taylor expansion is evaluated at each target in the box. A decision analysis ensures that Hermite expansions are formed and Taylor expansions evaluated only when it is efficient to do so; otherwise, box–box interactions take place directly or semi-directly.

The analytical apparatus required for the algorithm can be summed up in the rapidly converging series expansion

$$e^{-|x+y+z|^2}=\sum_{\alpha\geqslant 0}\sum_{\beta\geqslant 0}\frac{x^\alpha}{\alpha!}\frac{y^\beta}{\beta!}h_{\alpha+\beta}(z).$$

TABLE IV

Table of Cost and Errors for the Two-Dimensional Fast Gauss Transform with $\delta = 0.01$ and $\varepsilon = 10^{-6}$, with Targets and Sources Spaced Uniformly on a Circle

Case	$N = M$	Fast	Direct	Error/F
1	100	0.500	0.460	0.479E−09
2	200	1.540	1.840	0.447E−06
3	400	2.060	7.400	0.499E−06
4	800	2.370	29.600	0.737E−06
5	1600	3.180	117.920	0.749E−06
6	3200	4.320	486.080	0.755E−06
7	6400	6.930	1953.280	0.199E−06
8	12800	11.080	7686.400	0.199E−06
9	25600	19.690	30397.440	0.199E−06
10	51200	36.700	123141.120	0.200E−06
11	102400	72.130	485406.720	0.200E−06

Here x, y, z lie in $\mathbf{R}^d$, while $\alpha = (\alpha_1, ..., \alpha_d)$ and $\beta = (\beta_1, ..., \beta_d)$ are multiindices with positive integer elements, $x^\alpha = x_1^{\alpha_1} \cdots x_d^{\alpha_d}$, and h_γ is the d-dimensional Hermite function, which decays like a Gaussian as $|z|$ increases. Thus the influence of sources s_j in a box B with center s_B on targets t_i in a box C with center t_C is given by

$$[illegible] \sum \frac{(t-t_C)^\alpha}{[illegible]} \sum h \; [illegible]$$

$$\times \sum_{s_j \in B} \frac{(s_j - s_B)^\beta}{\beta!}.$$

This is a Taylor series about t_C. Its coefficients are formed by taking moments of the s_j's about s_B, summing over j, and transforming with a matrix multiply. One accumulates the Taylor coefficients due to all boxes B influencing C before evaluating. Accuracy is obtained by adjusting the number of terms retained in the sums over α and β. These sums converge extremely fast, so not very many terms are necessary in order to achieve quite high precision.

Table IV presents numerical results for a two-dimensional fast Gauss transform, with $\varepsilon = 10^{-6}$ and 7^2 terms kept in the Hermite series. These results show that the fast transform is never much slower than direct evaluation (for $N \geqslant 100$) and achieves tremendous speedups when N is large. The time required for evaluating the sum of 100,000 Gaussians at 100,000 points is reduced from a week to a minute by the fast Gauss transform.

5. RAPID EVALUATION OF LAYER POTENTIALS

In this section, we present our new algorithms for evaluating the discretized single layer potential

$$S\mu(x) = \int_\Gamma K(x, x')\, \mu(x')\, dx'$$

with optimal efficiency. From Section 3, we know that it is natural to consider separately the case when the evaluation point x is restricted to lie on Γ and the case when x lies anywhere in B, either on or off Γ. The applications make it natural also to consider two even more specific cases: First, in Section 5.1, we describe how to evaluate $S\mu(x)$ at the N points x_j on Γ where the values of μ were originally given. This is the essential part of solving for μ on Γ by an iterative method. We carry this out by using product integration in time and the fast Gauss transform to evaluate the local part, and non-equidistant FFT methods to evaluate the Fourier part. Optimal efficiency then dictates a certain balance between δ, p, and the mesh size h.

Second, in Section 5.2, we suppose μ given on each element and evaluate $S\mu(x)$ on an equispaced grid in B, in other words at the points $x = (i_1 H, ..., i_d H)$ with $0 \leqslant i_1, ..., i_d \leqslant M$ and $H = 1/M$. We assume the grid is coarser than the mesh on Γ, that is $H \geqslant h$, as this eliminates a tiresome consideration of cases and is the case in almost all applications. In this case, we evaluate the Fourier coefficients using the non-equidistant FFT and evaluate the Fourier part on the grid with a standard FFT. (If the grid were irregular, we could still construct an optimal algorithm by using the Fourier series evaluation scheme of Section 4.2). The local part is done by product integration with [illegible] in Section 5.3. Optimal efficiency now dictates a different relationship between p, δ, h, and H.

5.1. *Evaluation of $S\mu$ on Γ*

First split $S\mu = S_F\mu + S_L\mu$ with δ to be determined, and truncate $S_F\mu$ after terms with $k_i = p$. Then we need to evaluate p^d coefficients

$$\hat{\mu}(k) = \int_\Gamma \sin \pi k_1 x'_1 \cdots \sin \pi k_d x'_d \mu(x')\, dx',$$

in $O(N \log N)$ work, with accuracy $O(h^m) + \varepsilon$. If we use product q-point Gauss–Legendre quadrature on each element Γ_j, we obtain an expression of the form

$$\hat{\mu}(k) = \sum_{j=1}^{N} \sum_i \sin \pi k_1 x^i_{j1} \cdots \sin \pi k_d x^i_{jd} \mu^j_i w_{ij}.$$

This can be evaluated with the non-equidistant FFT (extended to do the d-dimensional sine transform) in $O((q^d N + p^d) \log p)$ work and with accuracy εF, where

$$F = \sum_{j=1}^{N} \sum_i |\mu^j_i|\, w_{ij} \approx |\mu|_1.$$

This suggests that we take $p = O(N^{1/d})$ and the Fourier series truncation error requirement (that $p^2\delta$ be bounded away from zero) suggests then that $\delta = O(N^{-2/d})$. In d dimensions, having N elements on Γ with maximum size h means roughly that $h = O(N^{1/(1-d)})$, because Γ is $(d-1)$-dimensional. Hence δ and h are related by $\delta = O(h^{2-2/d})$, and $p = O(h^{1/d-1})$. The error estimate for Gauss–Legendre quadrature presented in Section 3.1 is then dominated by

$$\begin{aligned}|E| &\leqslant C\delta^{1-d/2}\left(\frac{d\pi e}{8q}h^{1/d}\right)^{2q}|\mu|_\infty \\ &= O(h^{3-d-2/d+2q/d}).\end{aligned}$$

Thus we obtain order m accuracy uniformly in $1 \leqslant k_i \leqslant p$ if we take $q \geqslant d(m-3+d+2/d)/2$. For $d=2$, we need $q=m$, while in three dimensions we need $q = 1 + 3m/2$. The constant in the error estimate is quite small; for example, dropping factors of $C|\mu|_\infty$, it is 1.30 when $m=d=2$, 6.6×10^{-3} when $d=2$ and $m=4$, and generally speaking,

$$|E| \leqslant C|\mu|_\infty \left(\frac{\pi e}{4m}\right)^{md} h^m.$$

We can now evaluate the truncated Fourier series at the N points x_j, to obtain

$$\begin{aligned}S_F\mu(x_j) = 2^d \sum_{k_i=1}^{p} \frac{e^{-\pi^2|k|^2\delta}}{\pi^2|k|^2}\hat{\mu}(k) \\ \times \sin \pi k_1 x_1 \cdots \sin \pi k_d x_d + E,\end{aligned}$$

where the error E is of order

$$E = O(h^m) + O(\varepsilon) + O\left(\frac{e^{-\pi^2p^2\delta}}{p^3\delta^{(d+1)/2}}\right).$$

We do this by the Fourier series evaluation scheme of Section 4.2, extended to d-dimensional sine series, in other words, by evaluating $S_F\mu$ on a fine mesh with mesh size $H = 1/np$ and interpolating to each x_j with tensor product polynomial interpolation of degree $2K-1$. The error in this procedure is bounded by

$$|E| \leqslant C(H/2)^{2K}|f^{(2K)}|_\infty,$$

where

$$f(x) = \sum_{k_i=1}^{p} \frac{e^{-\pi^2|k|^2\delta}}{\pi^2|k|^2}\hat{\mu}(k)\sin \pi k_1 x_1 \cdots \sin \pi k_d x_d$$

is a smooth function. Comparison of the derivative of the sum with an integral shows that

$$|f^{(2K)}|_\infty \leqslant C_d|\mu|_1\delta^{-d/2-K+1},$$

where C_d is a constant of order unity, depending only on the dimension d. Thus

$$|E| \leqslant C|\mu|_1\delta^{1-d/2}\left(\frac{1}{4n^2p^2\delta}\right)^K.$$

In two dimensions, this bound can easily be made less than ε with choices of n and K which are independent of N; in three dimensions, n and K may have to increase logarithmically with N. Typically $n^2 \approx 10$ and $p^2\delta \geqslant 1$, so this error bound is $\leqslant 10^{-6}$ when $K=4$ (7th degree interpolation) if $C|\mu|_1 \leqslant 1$.

Thus if we choose p and δ with

$$\frac{e^{-\pi^2p^2\delta}}{40p^3\delta^{(d+1)/2}} \leqslant \varepsilon,$$

$p = O(N^{1/d})$, and $\delta = O(N^{-2/d})$, we can evaluate $S_F\mu$ on Γ in $O(N\log N)$ work with accuracy $O(h^m) + \varepsilon|\mu|_1$; the constant in $O(N\log N)$ depends only on $\log\varepsilon$.

Next we turn to the evaluation of the local part

$$S_L\mu(x) = \int_0^\delta\int_\Gamma K(x, x', t)\,\mu(x')\,dx'.$$

We use the product integration in time approach developed in Section 3.2 to reduce the problem to n Gauss transforms:

$$\begin{aligned}S_L\mu(x) &= \sqrt{\delta/\pi}\,w_0\mu(x) \\ &\quad + \sqrt{\delta/\pi}\sum_{j=1}^{n}\frac{w_j}{(4\pi\tau_j)^{(d-1)/2}}G_{4\tau_j}\mu(x) \\ &\quad + O(\delta^{n+3/2}).\end{aligned}$$

Since $\delta = O(h^{2-2/d})$, the error term looks like $O(h^{(2-2/d)(n+3/2)}) = O(h^m)$ if $n \geqslant (m-3+3/d)/(2-2/d)$. When $d=2$, we need to have $n \geqslant 1$ or $n \geqslant 3$ for second- or fourth-order accuracy, respectively, while in three dimensions, we need only $n \geqslant 0$ or $n \geqslant 2$, respectively. Thus in three dimensions, the simple approximation

$$S_L\mu(x) \approx \sqrt{\delta/\pi}\,\mu(x)$$

is already correct to second-order accuracy—and certainly very inexpensive to evaluate. In the other cases, we need to evaluate Gauss transforms to accuracy $\varepsilon' = \delta^{1-d/2}\varepsilon$ plus $O(h^m)$. The quadrature error in product q-point Gauss–Legendre quadrature for each Gauss transform is, from Section 3.2,

$$E = C\delta^{1-d/2}\left(\frac{e}{32q}\frac{nh^2}{\delta}\right)^q|\mu|_\infty.$$

We now know that $\delta = O(h^{2-2/d})$, so this estimate is equivalent to

$$E = O(h^{2q/d+3-d-2/d}).$$

This is $O(h^m)$ if $q \geqslant (md - 3d + d^2 + 2)/2$. If $d = 2$ we need $q \geqslant m$ while if $d = 3$ we need $q \geqslant 3m/2 + 1$.

The fast Gauss transform now evaluates $S_L\mu$ on Γ in $O(N \log N + \delta^{-d/2}) = O(N \log N)$ work, to accuracy ε'. Thus we can evaluate $S_\mu = S_F\mu + S_L\mu$ efficiently and accurately.

5.2. *Evaluation of $S\mu$ off Γ*

We now consider the evaluation of $S\mu$, discretized with N elements of size $\leqslant h$, on a mesh in the box B containing Ω. For concreteness, we consider an equally spaced mesh $x = (i/M, j/M, ...)$ on B; the more general case is an easy extension using the Fourier series evaluation scheme discussed in Section 4.2. We assume for simplicity that the boundary discretization is no coarser than the mesh, so that $h \leqslant H = 1/M$. The other case rarely occurs in practice and is easy to deal with when it does occur.

We allow ourselves $O((N + M^d) \log M)$ work to evaluate $S\mu$ on the grid; clearly this is optimal, up to a logarithm.

First we truncate $S_F\mu$ after terms with $k_i = p$. The resulting series will cost $O(M^d \log M)$ to evaluate on the mesh, so we might as well take $p = M$ and $\delta = O(M^{-2})$, with p and δ chosen to make the Fourier series truncation error $\leqslant \varepsilon\, |\mu|_1$. Now we need to evaluate M^d Fourier coefficients $\hat{\mu}(k)$ with $1 \leqslant k_i \leqslant M$. The fast algorithm error is bounded by (Section 3.1)

$$|E| \leqslant C\delta^{1-d/2} \left(\frac{d\pi e}{8q} Mh\right)^{2q} |\mu|_\infty$$

for $1 \leqslant k_i \leqslant M$. Since $Mh \leqslant 1$ by assumption, we have

$$|E| \leqslant C\delta^{1-d/2} \left(\frac{d\pi e}{8q}\right)^{2q} |\mu|_\infty$$

which can easily be made $\leqslant \varepsilon$ by choice of q. For example, $\varepsilon = 10^{-7}$ and $d = 2$ requires $q \geqslant 7$, while $\varepsilon = 10^{-12}$ and $d = 3$ still requires only $q \geqslant 11$ if $C\,|\mu|_\infty$ is of order unity. In practice, even a much smaller value of q suffices, because the Fourier series is dominated by lower terms for which $|k|\, h \ll 1$ and often we even have $Mh \ll 1$. Once we have the Fourier coefficients $\hat{\mu}(k)$, we multiply them by the appropriate Gaussian factors and evaluate $S_F\mu$ on the grid with a standard FFT. If the grid were non-equispaced, we would apply the Fourier series evaluation scheme of Section 4.2. We have now evaluated $S_F\mu$ on the grid at a cost of $O((N + M^d) \log M)$ work and with an accuracy of $O(h^m) + \varepsilon$.

Now we turn to the evaluation of the local part $S_L\mu(x)$ on the grid. This can actually be done in $O(N \log M)$ work, as it turns out. The key observation is that $\delta = O(M^{-2})$ and S_L decays to less than ε outside a tubular neighborhood of Γ having radius $O(\sqrt{\delta}) = O(1/M)$. Thus each of the N elements of Γ affects a number of grid points which depends only on $\log(\varepsilon/M)$, leading to $O(N \log M)$ work as N and M increase.

To be precise, suppose that we can ignore images so that $K_L(x - x') = (1/4\pi^{d-2})\, |x - x'|^{2-d}\, \Gamma(d/2 - 1,\ |x - x'|^2/4\delta)$. Then a point x at distance $D = \operatorname{dist}(x, \Gamma)$ from Γ has $S_L\mu(x)$ bounded by

$$\begin{aligned} S_L\mu(x) &\leqslant \frac{1}{4\pi^{d/2}} \int_\Gamma |x - x'|^{2-d} \\ &\quad \times \Gamma(d/2 - 1,\ |x - x'|^2/4\delta)\, \mu(x')\, dx' \\ &\leqslant \frac{1}{4\pi^{d/2}} |\mu|_1\, \Gamma(d/2 - 1,\ D^2/4\delta)\, D^{2-d} \\ &\leqslant C\left(\frac{D^2}{4\delta}\right)^{d/2-2} e^{-D^2/4\delta} D^{2-d} \end{aligned}$$

from Section 2. Suppose for conceteness that $\delta = M^{-2}$, so the Fourier series truncation error is about $\varepsilon = 10^{-5}$. Then this error bound is below $\varepsilon\, |\mu|_1$ when $D \geqslant K/M$, where $K \geqslant 6$ in two dimensions. Thus, to single precision accuracy, $S_L\mu$ is zero when x lies more than six grid spacings from Γ. Hence each point on Γ can affect at most 13^d grid points. The general case is similar.

Including images presents no additional difficulty, because they are subject to the same estimate. Only 3^d images need be included, and even these matter only if Γ comes very close to ∂B.

Thus we need evaluate $S_L\mu(x)$ only at $O(N \log M)$ grid points near Γ. When we do evaluate it, we use the scheme described in Section 3.3, evaluate the singular part of

$$|x - x'|^{2-d}\, \Gamma(d/2 - 1,\ |x - x'|^2/4\delta)$$

exactly over each element, and apply Gaussian quadrature to the remainder. The singular part is a logarithm if $d = 2$ and $|x - x'|^{-1}$ if $d = 3$; it can be integrated exactly over a linear element to obtain second-order accuracy and quite likely over a cubic element for fourth-order accuracy. See [7] for some discussion of the difficulties involved. (Actually, if $\delta = O(H^2)$ as is often the case, then one need only use quadratic elements in the evaluation of the local part to obtain fourth-order accuracy and zero-order elements to obtain second-order accuracy, because $S_L\mu$ is itself of size $O(h)$.)

The smooth part can be integrated by Gaussian quadrature with an error bound that looks like (Section 3.3)

$$E \leqslant C\left(\frac{h}{\sqrt{\delta}}\right)^{2q} |\mu|_{\infty}.$$

At the very worst, we will have $h \leqslant 1/M$ and $\sqrt{\delta} = 1/M$, whereupon

$$E \leqslant C 16^{-q} |\mu|_{\infty}.$$

Thus $q \geqslant 6$ gives single precision and $q \geqslant 10$ gives double precision accuracy assuming $C\,|\mu|_{\infty}$ is of order unity.

We have now evaluated $S_L\mu$ off Γ to accuracy $O(h^m) + \varepsilon$; hence we can add together $S_F\mu$ and $S_L\mu$ to obtain the full single layer potential $S\mu$ off Γ, evaluated in $O((N + M^d)\log M)$ operations, with the same order of accuracy.

In many calculations [1, 6], one needs to evaluate a discrete sum of point sources

$$S(i) = \sum_{j=1}^{N}{}' \mu_j K(x_i, x_j), \qquad 1 \leqslant i \leqslant N, \tag{23}$$

where the prime on the sum indicates that the $j = i$ term is to be omitted. Here x_i are distinct points given in $B = [0, 1]^d$ and K is a Green function for $-\Delta$ with boundary conditions imposed on ∂B. Direct evaluation of this sum costs $O(N^2)$ operations since one must sum up $N - 1$ sources for each target i. We present an algorithm which evaluates (23) much more efficiently.

First, we describe a method which is optimal only when the points x_i are distributed over B in a fairly uniform way. When the x_i's are uniformly distributed on a lower-dimensional set as $N \to \infty$, the algorithm is no longer optimal, but is still very fast; in numerical examples, it achieves tremendous speedups over direct calculation. The work estimate of the algorithm depends on the Hausdorff dimension D of the set where the x_i's concentrate as $N \to \infty$.

We also sketch an $O(N \log N)$ algorithm, which we have not implemented. It requires more complicated analysis and some new special function theory, which will be discussed elsewhere.

Method 1

Suppose for concreteness that K is the Dirichlet Green function for B, as described in (2). Then we have the Ewald splitting

$$S(i) = S_F(i) - \mu_i K_F(x_i, x_i) + S_L(i).$$

Here

$$S_F(i) = 2^d \sum_{k_i=1}^{p} \frac{e^{-\pi^2 |k|^2 \delta}}{\pi^2 |k|^2} \hat{\mu}(k) \times \sin \pi k_1 x_{i1} \cdots \sin \pi k_d x_{id} + E_F,$$

where

$$\hat{\mu}(k) = \sum_{j=1}^{N} \mu_j \sin \pi k_1 x_{j1} \cdots \sin \pi k_d x_{jd}$$

and E_F satisfies the Fourier series truncation error bound

$$|E_F| \leqslant \frac{e^{-\pi^2 p^2 \delta}}{50 p^3 \delta^{(d+1)/2}} M,$$

where $M = \sum |\mu_j|$. The second term must be subtracted because our original sum excluded the term with $j = i$. The local part is given by

$$S_L(i) = \frac{1}{4\pi^{d/2}} \sum_{j=1}^{N}{}' \sum_{\text{images}} \pm |x_i - \tilde{x}_j|^{2-d} \times \Gamma(d/2 - 1, |x_i - \tilde{x}_j|^2 / 4\delta) + E_L,$$

where we keep only 3^d images $\tilde{x}_j$ of each point x_j, those lying in the image boxes adjacent to B, and the error E_L in discarding the rest of the images is bounded by

$$|E_L| \leqslant C_d \delta^{2-d/2} e^{-1/4\delta} M,$$

with a constant C_d of order unity. We consider each of the three terms forming $S(i)$ in turn.

First, it is clear that we can evaluate the Fourier part $S_F(i)$ with the methods of Sections 4.1 and 4.2. This will require $O((p^d + N)\log p)$ work to produce accuracy $\varepsilon M/3$, say, if p and δ are chosen to make the Fourier series truncated after p^d terms accurate to $\varepsilon M/3$. This requires that $p^2\delta$ be bounded away from zero.

Next consider the subtracted term

$$\mu_i K_F(x_i, x_i) = \mu_i 2^d \sum_{k_i=1}^{p} \frac{e^{-\pi^2 |k|^2 \delta}}{\pi^2 |k|^2} \times \sin^2 \pi k_1 x_{i1} \cdots \sin^2 \pi k_d x_{id} + E_S,$$

where E_S satisfies the same estimate as E_F. At first glance, this term seems trivial, because each point x_i interacts only with itself. Unfortunately, $K_F(x_i, x_i)$ depends on δ and we have to sum up p^d values to evaluate $K_F(x, x)$ at each value of x. Thus it looks as though this term would cost $O(p^d N)$ which would be far too much.

Fortunately, it turns out that $K_F(x, x)$ can be evaluated by a fast method very similar to the Fourier series evaluation method of Section 4.2. We evaluate $K_F(x, x)$ on a fine mesh in B and interpolate to each desired value of x. The fine mesh evaluation must be done efficiently (though it could of course be done once and for all for each δ and stored permanently) and for this we need to observe that

$$4 \sin^2 x \sin^2 y = 1 - \cos 2x - \cos 2y + \cos 2x \cos 2y$$

or the analogue in higher dimensions. Thus $K_F(x, x)$ can be evaluated on a regular grid by zero-padding and fast cosine transforms.

Finally, consider the local part. Here, the essential feature is rapid decay. Choose $R = R(\delta)$ so that

$$C_d \frac{\delta^{2-d/2}}{R^2} e^{-R^2/4\delta} \leqslant \varepsilon/3;$$

then $R = O(\sqrt{\delta})$ up to a logarithm. Then the local term is less than $\varepsilon M/3$ whenever $|x - x'| \geqslant R$. Hence each point x_j only influences points x_i with $|x_i - x_j| \leqslant R$. The assumption that the x_j's are uniformly distributed on a set of Hausdorff dimension D as $N \to \infty$ means then that each x_j can influence only $O(R^D N)$ points as $N \to \infty$, so we can limit the sum to such points.

In practice, this needs a little further work, because we want to exclude the influence of distant points x_j *without* computing the distance to each point. (The latter would cost $O(N^2)$ work which would be too much.) This can be done by the standard technique of organizing the points into boxes of size $\geqslant R$ and allowing points in one box to interact only with points in the nearest neighbor boxes. This technique also allows easy inclusion of images, by using an extra layer of fictitious boxes outside B.

Hence the total work required to evaluate $S_L(i)$ for $1 \leqslant i \leqslant N$ is

$$O(R^D N^2) = O(\delta^{D/2} N^2).$$

Now we can minimize the total work,

$$W = O(p^d \log N + N \log N + \delta^{D/2} N^2),$$

subject to the constraint that $p^2\delta$ is bounded away from zero. The result is

$$W = O(N^\alpha \log N),$$

where $\alpha = 2/(1 + D/d)$. Clearly $\alpha = 2$ when $D = 0$ (points converge to a finite set of points as $N \to \infty$), while $\alpha = 1$ when $D = d$. Thus the algorithm is optimal if the points x_i cover B in a reasonably uniform way when $N \to \infty$. An interesting intermediate case is when the x_i's are uniformly distributed over a hypersurface, so $D = d - 1$. Then we find $\alpha = 2/(2 - 1/d)$. In two dimensions this gives us an $O(N^{4/3} \log N)$ algorithm while in three dimensions we get an $O(N^{6/5} \log N)$ algorithm. Thus the algorithm differs little from an $O(N \log N)$ algorithm in this case; the ratio $N^{1/3}$ is less than 100 for $N \leqslant 10^6$, while $N^{1/5}$ is bounded by 10 for $N \leqslant 10^5$. In practice, these methods achieve large speedups over direct evaluation.

Note that we were able to do better than this when the sources were distributed on a hypersurface, in the continuous case when the discrete problem corresponded to quadrature of a layer potential. This is because in the continuous case, we were able to classify certain parts of the error as quadrature error; we do not have this option when evaluating discrete sums.

Method 2

As we have seen, the difficulty in making an $O(N \log N)$ algorithm is due to the local part $S_L(i)$. One needs a multipole-type expansion which separates the variables yet—unlike multipoles—preserves localization. Such an expansion can be constructed by integrating the Hermite expansion which was used to construct the fast Gauss transform. Begin with the expansion [12]

$$e^{|x-y|^2/\delta} = \sum_{\alpha \geqslant 0} \frac{1}{\alpha!} \left(\frac{x^\alpha}{\sqrt{\delta}}\right) h_\alpha \left(\frac{y}{\sqrt{\delta}}\right). \tag{24}$$

Ignoring images temporarily, we have

$$K_L(x - y) = \int_0^\delta \frac{e^{-|x-y|^2/4t}}{(4\pi t)^{d/2}} \, dt.$$

Combining these two expressions gives an expansion

$$\begin{aligned} K_L(x - y) &= \sum_{\alpha \geqslant 0} \frac{x^\alpha}{\alpha!} \int_0^\delta (4\pi t)^{-d/2} \\ &\quad \times (4t)^{-|\alpha|/2} h_\alpha \left(\frac{y}{\sqrt{4t}}\right) dt \\ &= \sum_{\alpha \geqslant 0} \frac{x^\alpha}{\alpha!} g_\alpha(y). \end{aligned}$$

This expansion can be used in the same way as (24) was used in the derivation of the fast Gauss transform, if allowances are made for the singularity of the functions g_α. The technique required to make these allowances is precisely the same as in the fast multipole method [5], but with the substantial simplification of localization; $K_L(x - y)$ decays rapidly as $|x - y|$ increases. Images are included (if necessary) in the obvious way.

This scheme is theoretically elegant—and practical compared to direct evaluation—but it seems unlikely to be competitive with Method 1 except in situations unlikely to occur. Thus we did not implement or test it in this paper. It will be discussed in a future publication if it turns out to be practical.

7. GENERALIZATIONS

We have presented fast algorithms which evaluate a discretized version of the single layer potential with an arbitrary order of accuracy, in an essentially optimal amount of computational effort. Different approaches are used to evaluate $S\mu$ on and off Γ, corresponding to the common situation where one solves an integral equation on Γ by iteration, then evaluates the potential of the resulting density μ on a grid in the domain Ω.

Our algorithm can immediately be generalized to solve many other problems which arise in practice. We list some examples.

1. The modifications needed to evaluate double rather than single layer potentials are straightforward. This is important in practice because one usually solves a standard Dirichlet or Neumann problem on Ω by converting it to an integral equation on Γ, in which the integral operator is the nonsingular part of either the double layer or the normal derivative of the single layer. Thus one often needs to apply such an operator efficiently on Γ.

2. One can easily modify the analysis to handle potentials formed with other Green functions for $-\Delta$ on a cube; for example, the periodic Green function is dealt with by replacing sine series by exponential Fourier series. This is useful in periodic computations with interfaces.

3. Volume potentials Vf can be evaluated, say on a regular $M \times M$ grid in B, using the values of f on the grid points lying inside Ω. The work estimate is $O(M^d \log M)$ on a M^d-point grid. The only new piece of work that must be done is to do product integration over d-dimensional elements on Ω rather than $(d-1)$-dimensional ones on Γ. Much benefit is derived from the fact that the local part $V_L f$ is $O(h^2)$ to begin with; thus quadratic approximation of f gives fourth-order accuracy, and second-order accuracy can be achieved by dropping the local part altogether.

solvers for boundary value problems for any elliptic equation or system which admits a potential theory. An important example in applications is the stationary Stokes equations, for which the fundamental solution is known and Ewald summation has been described in [15]. (Boundary element methods for this problem have been constructed, e.g., in [16].) The analysis goes through in the same way, and the result is a fast solver for the Stokes equations in a bounded domain or for Stokes flow with interfaces.

5. Precisely the same generalizations can be made for the discrete sum algorithm of Section 6.

8. NUMERICAL RESULTS

We programmed two-dimensional versions of three of the algorithms presented in this paper and tested them on examples. The computations were done in double-precision arithmetic in optimized Fortran on a SPARC station 1 or a Sun-4 workstation.

The results are quite satisfactory; all three algorithms are much faster than direct evaluation schemes for large-scale computations, while the overhead is small enough that it is feasible to use them for very small calculations as well. They break even at a very small number of points and achieve dramatic speedups for large jobs. The $O(N \log N)$ and $O((M^2 + N) \log N)$ time estimates for the evaluation of the single layer potential on and off Γ are verified by the numerical results. The discrete sum algorithm exhibits linear time requirements when the points are uniformly distributed and $O(N^{4/3} \log N)$ when the points lie on a curve, as predicted. In both cases, a considerable speedup is obtained, even when N is as small as 160.

The accuracy of all three calculations is excellent. The layer potential calculations were clearly second-order accurate or better, while the discrete sum evaluation scheme achieved the error tolerance desired and was substantially more accurate than direct evaluation when the number of points was large.

8.1. *Layer Potentials Evaluated on Γ*

We tested the algorithm of Section 5.1 on two examples, the first for accuracy and speed, the second only for speed, and compared it with a direct evaluation scheme. In the direct method, the same discretized single layer potential is evaluated on Γ by direct summation. Thus the direct calculation already uses separation of variables, product integration in time, and Gaussian integration over each element. We programmed it also to evaluate Gaussians only when they were above the cutoff ε.

In this type of experiment, a standard time-saving procedure is to use the direct calculation to evaluate the potential only at 100 of the N points on Γ. The resulting CPU time is then multiplied by $N/100$ to obtain an estimate of the time

entire calculation. In our present situation, the direct calculation still has to evaluate *all* the Fourier coefficients even if only 100 values of $S\mu$ are desired. Thus the standard procedure would punish the direct calculation unfairly. Hence we compared our results with the full direct calculation, as long as the time required did not try our patience, and estimated the time required for larger direct calculations by extrapolation. In other words, we multiplied T_d by 4 whenever we doubled N. This procedure tends to produce conservative estimates.

In our first numerical example, we took Γ to be an off-center circle, with radius 0.13 and center at (0.4, 0.7), parametrized by $0 \leqslant \theta \leqslant 2\pi$, and we took the density $\mu(\theta) = 10k \cos(k\theta)$, with $k = 3$. We tested the algorithm with various values of k between 1 and 10; the results we report were obtained with $k = 3$, but they would be little different in form for other cases. The main difference is that the asymptotic second-order accuracy takes longer and longer to be reached as k is increased, because it takes more and more points to resolve the rapid variations in μ. The potential $S\mu$ is of order unity, and is plotted in Fig. 1. It can be evaluated essentially exactly by numerical integration, and the accuracy of both fast and direct evaluation schemes compared to an exact solution.

The numerical parameters N, p, and δ are reported in Table V. We set the tolerance ε to 10^{-4} initially and reduced it by a factor of 4 at each step. This is because it would have been pointless to demand an accuracy of evaluation much less than quadrature error could reasonable be expected to be. The parameters δ and p were chosen so that $e^{-\pi^2 p^2 \delta}/50p^3\delta^{3/2} \leqslant \varepsilon$ initially (when $N = 20$) and then refined by factors of 1.5 and 0.5, respectively, as N was doubled. We used $n = 1$ level of product integration in time, $q = 2$ points for Gaussian quadrature on each element to evaluate the Fourier coefficients, and $q = 2$ points per element to evaluate the Gauss transform. The Fourier series evaluation scheme used a fine mesh of size $3p$ and fifth-order interpolation. The fast transform parameters were tested by refining them to see that they made no significant difference in the error in the numerical solution.

The numerical results are shown in Table V. The timing runs were made on a SPARC station 1 with the FORTRAN optimizer; this is about a one-megaflop machine. The columns headed T_f and T_d give respectively the time required for the fast algorithm to evaluate $S\mu$ and the time required to evaluate $S\mu$ directly at all N points. It is clear that the fast algorithm is much faster than direct evaluation for large jobs and breaks even for a surprisingly small number (between 20 and 40) of points on the curve. With 10,000 points on the circle, the fast algorithm is over 300 times faster than the direct calculation, taking 3 min rather than 17 h.

The columns headed E_f and E_d report the maximum error measured in 200 randomly chosen values of $S\mu$ on Γ, for the fast and direct computations, respectively. Asterisks indicate cases for which we did not obtain the error in the direct calculation because we did not run it. Once the oscillations in the solution are well-resolved, the error in the method is clearly at least second order. There is no appreciable difference between the errors of the fast and direct calculations, because the quadrature error dominates the error due to the fast evaluation schemes. One feature of the error which is not apparent from the tables is that the relative error is small, as well as the absolute error. In places where the potential is small, the error is also small; the potential has roughly the same number of correct significant digits in all places.

Next, we ran a computation with a more complicated boundary just to see how much faster the fast algorithm would be in a more realistic situation. In this case, we took Γ to be the union of four circles, modelled on a typical problem in crystal growth [27, 28], and μ to be $20 \cos 2\theta$ on each circle. More applications to crystal growth will be reported in future publications. The single layer potential is shown in Fig. 2. The timings given in Table VI were obtained with the same parameters that were used for the previous example, but this time we did not measure the error. Study of the results at selected points suggests, however, that the error is quite similar to the previous case, so that we achieve 1% accuracy with $N = 640$ points distributed over the four circles. The fast algorithm is then about nine times faster than the direct evaluation; it is never slower, and soon becomes much faster as the mesh is refined. With 20,000 points, it is 240 times faster than direct evaluation.

8.2. *Layer Potentials Off Γ*

In the next pair of examples, we tested the evaluation of $S\mu$ on a regular grid of size $M \times M$, for the potential of a

TABLE V

Results of Evaluating (on the Circle) the Single-Layer Potential of $30 \cos 3\theta$ on a Circle

Case	N	p	δ	E_f	E_d	T_f	T_d
1	20	9	0.01024	0.478	0.478	0.29	0.21
2	40	13	0.00512	0.202	0.202	0.57	0.80
3	80	19	0.00256	0.572E − 1	0.573E − 1	1.09	3.35
4	160	28	0.00128	0.119E − 1	0.119E − 1	2.40	14.11
5	320	42	0.00064	0.218E − 2	0.218E − 2	4.51	61.61
6	640	63	0.00032	0.379E − 3	0.379E − 3	10.15	249.38
7	1280	94	0.00016	0.645E − 4	**	20.42	985.76
8	2560	141	0.00008	0.105E − 4	**	41.56	3943.04
9	5120	211	0.00004	0.166E − 5	**	93.02	15772.16
10	10240	316	0.00002	0.264E − 6	**	192.75	63088.64

TABLE VI

Time Required to Evaluate (on the Circles) the Single Layer Potential of $20 \cos 2\theta$ on four circles

Case	N	p	δ	T_f	T_d
1	80	9	0.01024	0.79	1.05
2	160	13	0.00512	1.43	4.18
3	320	19	0.00256	3.78	16.15
4	640	28	0.00128	7.43	64.60
5	1280	42	0.00064	15.88	258.40
6	2560	63	0.00032	29.32	1033.60
7	5120	94	0.00016	65.63	4134.40
8	10240	141	0.00008	127.29	16537.60
9	20480	211	0.00004	277.85	66150.40

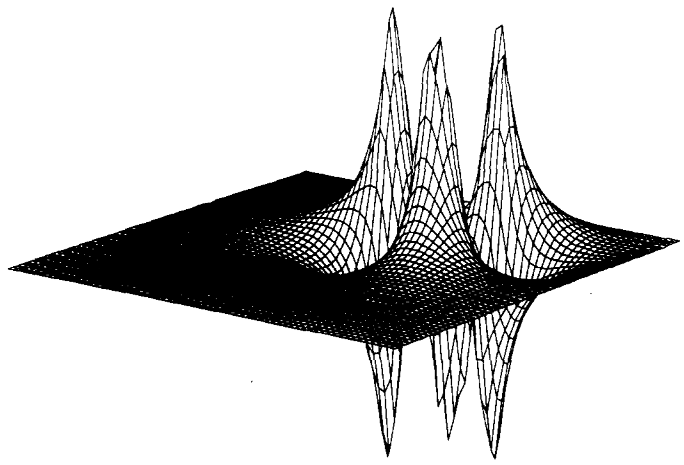

FIG. 1. Single layer potential of 30 cos 3θ on an off-center circle.

circle or four circles. We did not include images for the local calculation, so δ was chosen so that images were negligible. This put a lower bound on $p = M$ via the Fourier series truncation error estimate. The direct calculation now consists of the same local part as the fast algorithm, cut off at the same distance, together with direct evaluation of the Fourier coefficients of the local part. The Fourier series is then evaluated on the grid with an FFT, as direct evaluation would be unfair. Thus the only difference between the fast and direct calculations is that the fast algorithm evaluates the Fourier coefficients much more efficiently. The fast algorithm is $O(M^2 + N)$ work, whereas the direct calculation requires $O(M^2 N)$ work. The growth of the time required by the fast algorithm is actually closer to linear than to quadratic; this is because most of the effort is spent on evaluating the local part, which is an $O(N)$ calculation. Again, the error was relatively small as well as absolutely small; the potential has roughly the same number of significant digits in different places, despite the fairly rapid spatial variation evident in Fig. 1. In this calculation, we used $q = 5$ points for Gaussian quadrature of both the local part and the Fourier coefficients. The same sequence of tolerances ε was used as in the previous example.

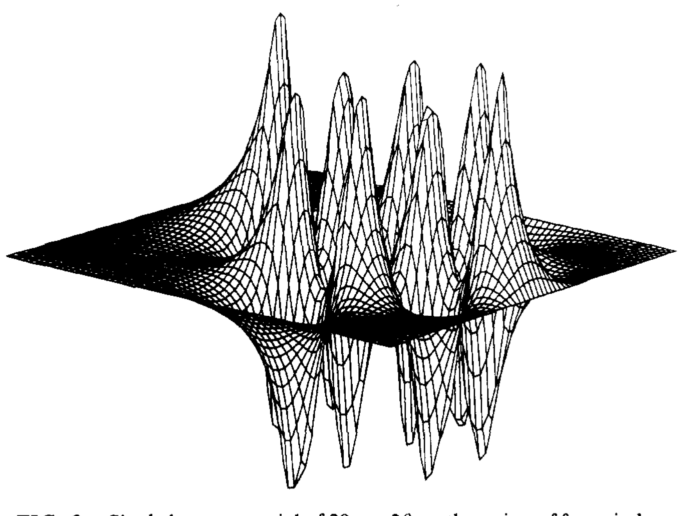

FIG. 2. Single layer potential of 20 cos 2θ on the union of four circles.

TABLE VII

Results of Evaluating (on a $M \times M$ Grid) the Single Layer Potential of 30 cos 3θ on a Circle

Case	N	M	δ	E_f	E_d	T_f	T_d
1	20	7	0.02048	0.111E−01	0.111E−01	3.89	3.67
2	40	13	0.00512	0.785E−02	0.785E−02	10.29	10.58
3	80	25	0.00128	0.898E−03	0.898E−03	21.65	28.71
4	160	49	0.00032	0.285E−03	0.285E−03	45.15	113.22
5	320	97	0.00008	0.658E−04	0.658E−04	97.71	688.92
6	640	193	0.00002	0.144E−04	0.144E−04	229.83	5173.21

Table 8 shows results for evaluating the four-circle potential on the grid. Again, the fast algorithm is much faster than the direct calculation. A speedup of 70 is obtained with 2560 points distributed over the four circles. Asterisks denote direct timings estimated by extrapolation from previous values.

8.3. *Discrete Sums*

Finally, we present two numerical examples for the discrete sum algorithm. In the first, we generated N random points uniformly on B, and observe the predicted linear growth of work with N; in the second, we distribute points uniformly on a circle and observe the expected $O(N^{4/3} \log N)$ work requirement. Both cases exhibit a considerable speedup, compared to a direct calculation with the same accuracy. The breakeven point is quite low as well, indicating that the fast algorithm has very little overhead.

We compared our scheme with a direct evaluation scheme which computes

$$S(i) = \sum_{j=1}^{N}{}' K(x_i, x_j)\, \mu_j$$

to accuracy εM with $M = \sum |\mu_j|$. To carry out the direct evaluation, we split $K = K_F + K_L$ by Ewald summation and

TABLE VIII

Time Required to Evaluate (on the Grid) the Single Layer Potential of 20 cos 2θ on the Union of Four Circles

Case	N	M	δ	T_f	T_d
1	80	7	0.02048	8.54	7.84
2	160	13	0.00512	19.12	20.69
3	320	25	0.00128	40.08	69.29
4	640	49	0.00032	87.07	554.32
5	1280	97	0.00008	205.24	4434.56*
6	2560	193	0.00002	509.30	35476.48*

TABLE IX

Results for the $O(N \log N)$ Discrete Sum Algorithm with Uniformly Distributed Points

Case	N	p	δ	T_f	T_d	E_f	E_d
1	10	5	0.020480	0.73	0.82	0.602E−06	0.470E−07
2	20	8	0.010240	1.44	3.50	0.263E−06	0.696E−07
3	40	12	0.005120	2.87	14.32	0.125E−06	0.440E−07
4	80	17	0.002560	11.12	56.88	0.606E−07	0.468E−07
5	160	25	0.001280	21.06	228.16	0.233E−07	0.419E−07
6	320	36	0.000640	35.39	908.16	0.688E−08	0.189E−07
7	640	51	0.000320	74.88	3701.12	0.568E−08	0.279E−07
8	1280	73	0.000160	142.81	14896.64	0.133E−08	0.229E−07
9	2560	104	0.000080	285.96	59898.88	0.119E−08	0.236E−07
10	5120	148	0.000040	545.12	236687.36	0.435E−09	0.233E−07
11	10240	210	0.000020	1123.72	944834.56	0.107E−09	0.236E−07
12	20480	297	0.000010	2113.84	3753779.20	0.763E−10	0.226E−07
13	40960	421	0.000005	4111.72	15002624.00	0.396E−10	0.227E−07

evaluate each piece to accuracy $\varepsilon/2$. The splitting parameter δ' for the direct evaluation is chosen as large as possible subject to the restriction that only nine images be kept; thus we required

$$\frac{2\delta'}{\pi} e^{-1/4\delta'} \leqslant \varepsilon/2.$$

The number of terms p^2 kept in K_F was then set by the Fourier series truncation error estimate

$$\frac{e^{-\pi^2 p^2 \delta'}}{50 p^3 (\delta')^{3/2}} \leqslant \varepsilon/2.$$

Typically, we took $\varepsilon = 10^{-6}$, $\delta' = 0.02$, and $p = 7$. We used the direct summation method to evaluate the potential at 10 points and extrapolated the time estimates.

First we used random x_i uniformly distributed over B. We took $\varepsilon = 10^{-6}$, and used a mesh of size $4p$ with seventh-order interpolation in the Fourier series evaluation scheme. We took $N = 10, 20, 40, \ldots$. The results are shown in Table IX, where T_d is the CPU time required for the direct calculation, T_f is the time required by the fast algorithm, and E_d and E_f are the corresponding errors. The error E_f is asymptotically smaller than the error E_d in direct evaluation, and the speedup T_d/T_f is astronomical. With 40,960 points, the direct calculation would take almost six months to do what the fast algorithm does in a little over one hour. The fast algorithm incurs so little overhead that it is faster than direct evaluation even with only 10 points.

Our second test case used x_i uniformly distributed on a circle of radius 0.39 and center (0.4, 0.4), so it almost touches the edge of the box and images must be included. Now our refinement strategy was to increase p by $2^{2/3}$ and reduce δ by $2^{4/3}$ when N was doubled. The work is supposed to be $O(N^{4/3} \log N)$ in this case, and the numerical results bear out this expectation. A speedup of 360 is achieved when $N = 10{,}000$.

TABLE X

Results for the Discrete Sum Algorithm with Points Distributed on a Circle; the Algorithm is then $O(N^{4/3} \log N)$.

Case	N	p	δ	T_f	T_d	E_f	E_d
1	20	6	0.020480	1.63	2.70	0.635E−07	0.511E−07
2	40	10	0.008127	2.93	10.84	0.469E−07	0.156E−07
3	80	16	0.003225	9.28	44.08	0.192E−07	0.153E−07
4	160	26	0.001280	19.15	177.76	0.378E−08	0.111E−07
5	320	42	0.000508	34.65	718.40	0.523E−08	0.555E−08
6	640	67	0.000202	78.27	2947.20	0.411E−08	0.720E−08
7	1280	107	0.000080	183.00	11773.44	0.259E−08	0.760E−08
8	2560	170	0.000032	523.16	46376.96	0.138E−08	0.999E−08
9	5120	270	0.000013	970.04	176517.12	0.925E−09	0.816E−08
10	10240	429	0.000005	1951.72	708864.00	0.271E−08	0.846E−08

9. CONCLUSIONS

We have presented three new families of fast algorithms for classical potential theory and demonstrated their practicality with numerical results. Our algorithms are based on a new approach which combines Ewald summation with fast transforms for simple special functions. The key is Ewald summation, which separates the Green function into low-frequency global information and high-frequency local information. The low-frequency information is carried by a rapidly converging Fourier series and the high-frequency information is localized near the singularity. Adjusting the splitting parameter adjusts the decay in Fourier space versus the decay in real space and leads to new fast algorithms for various problems.

Numerical results show that our new approach is accurate and efficient. The breakeven point is amazingly low, and the new algorithms achieve considerable speedups for large problems. When one wants to evaluate a layer potential to one percent accuracy on a reasonably complicated domain, the fast algorithm is nine times faster than even a sophisticated direct calculation. When higher accuracy is desired or the boundary is more complicated, requiring a finer discretization, the fast algorithm can be several hundred times faster. Ewald summation is responsible for the accuracy of the algorithm which evaluates the potential on Γ, because it splits the potential into pieces, each structured so that simple Gauss–Legendre quadrature can be used for each element. The efficiency is then due to the use of the fast Gauss transform and fast manipulation methods for Fourier series. When evaluating the potential off Γ, accuracy suggests the use of product integration for the local part. Efficiency can still be achieved because the local part decays rapidly when one moves away from Γ.

Discrete sums can be evaluated rapidly as well, suggesting that this algorithm will be very useful in large-scale computations with periodic vortex methods or other particle methods. Numerical results for discrete sums show that speedups of 4000 can be achieved in calculations with 40,000 random uniformly distributed particles, and the algorithm breaks even at 10 particles.

ACKNOWLEDGMENTS

I thank V. Rokhlin, for suggesting the idea of his unpublished non-equidistant FFT algorithm to me, and one of the referees, for his constructive criticism.

REFERENCES

1. D. J. Adams and G. S. Dubey, *J. Comput. Phys.* **72**, 156 (1987).
2. C. Anderson, *J. Comput. Phys.* **62**, 111 (1986).
3. G. Beylkin, R. R. Coifman, and V. Rokhlin, preprint, 1989.
4. A. Brandt and A. A. Lubrecht, *J. Comput. Phys.*, to appear.
5. J. Carrier, L. Greengard, and V. Rokhlin, *SIAM J. Sci. Statist. Comput.* **9**, 669 (1988).
6. A. J. Chorin, *Commun. Math. Phys.* **83**, 517 (1982).
7. M. Costabel, *SIAM J. Math. Anal. Appl.* **19**, 613 (1988).
8. P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration* (Ginn, Blaisdell, Boston, 1967).
9. H. Dym and H. P. McKean, *Fourier Series and Integrals* (Academic Press, New York/London, 1972).
10. A. Erdelyi (Ed.), *Higher Transcendental Functions,* Vol. **II** (McGraw-Hill, New York, 1953).
11. P. P. Ewald, *Ann. Phys. (Leipzig)* **64**, 253 (1921).
12. L. Greengard and J. Strain, *SIAM J. Sci. Statist. Comput.* **12**, 79 (1991).
13. L. Greengard and J. Strain, *Commun. Pure Appl. Math.* **XLIII**, 949 (1990).
14. J. A. Gregory, in *Mathematics of Finite Elements and Applications IV, MAFELAP 1981*, edited by J. R. Whiteman (Academic Press, New York, 1982), p. 498.
15. H. Hasimoto, *J. Fluid Mech.* **5**, 317 (1959).
16. F. K. Hebeker, in *Mathematics of Finite Elements and Applications V, MAFELAP 1984*, edited by J. R. Whiteman (Academic Press, New York, 1985), p. 257.
17. E. Hille, *Ann. of Math.* **27**, 427 (1926).
18. F. de Hoog and R. Weiss, *Math. Comput.* **27**, 295 (1973).
19. C. G. L. Johnson and L. R. Scott, *SIAM J. Numer. Anal.* **26**, 1356 (1989).
20. M. Kac, *Integration in Function Space and Some of Its Applications* (Academia Nazionale dei Lincei Scuola Normae Superiore Lezioni Fermiane, Pisa, 1980).
21. O. Kellogg, *Foundations of Potential Theory* (Dover, New York, 1937).
22. J. C. Nedelec, in *Mathematical Analysis and Numerical Methods for Science and Technology*, by R. Dautray and J.-L. Lions, transl. by I. N. Sneddon, Springer-Verlag, New York/Berlin, 1988, Chaps. 11 and 13.
23. L. Reichel, *J. Comput. Math.* **14**, 125 (1986).
24. V. Rokhlin, *J. Comput. Phys.* **60**, 187 (1985).
25. V. Rokhlin, unpublished manuscript.
26. J. E. Romate, *SIAM J. Numer. Anal.* **27**, 529 (1990).
27. J. Sethian and J. Strain, *J. Comput. Phys.*, in press.
28. J. Strain, *J. Comput. Phys.* **85**, 342 (1989).
29. W. L. Wendland, in *Mathematics of Finite Elements and Applications V, MAFELAP 1984*, edited by J. R. Whiteman (Academic Press, New York, 1985), p. 193.