THE FAST GENERALIZED GAUSS TRANSFORM [*]

MARINA SPIVAK, SHRAVAN K. VEERAPANENI, AND LESLIE GREENGARD

Abstract. The fast Gauss transform allows for the calculation of the sum of N Gaussians at M points in $\mathcal{O}(N+M)$ time. Here, we extend the algorithm to a wider class of kernels, motivated by quadrature issues that arise in using integral equation methods for solving the heat equation on moving domains. In particular, robust high-order product integration methods require convolution with $\mathcal{O}(q)$ distinct Gaussian-type kernels in order to obtain q^{th} order accuracy in time. The generalized Gauss transform permits the computation of each of these kernels and, thus, the construction of fast solvers with optimal computational complexity.

We also develop plane-wave representations of these Gaussian-type fields, permitting the "diagonal translation" version of the Gauss transform to be applied. When the sources and targets lie on a uniform grid, or a hierarchy of uniform grids, we show that the *curse of dimensionality* (the exponential growth of complexity constants with dimension) can be avoided. Under these conditions, our implementation has a lower operation count than the fast Fourier transform (FFT).

Key words. Gauss transform, fast algorithms, heat potentials, high-order accuracy, tensor-product grids

AMS subject classifications. 35K05, 31A10, 65N38, 65Y20

1. Introduction.

The fast Gauss transform (FGT) [9] computes discrete sums of the form

$$\sum_{k=1}^{N} f_k e^{-\frac{\|x_j-y_k\|^2}{\delta}} \quad \text{at} \quad \{x_j \,|\, j=1,...,M\}\,, \tag{1.1}$$

where $x_j, y_k \in \mathbb{R}^d$, using only $\mathcal{O}(N+M)$ operations. Since this is a common computational kernel, the algorithm has been applied in a variety of fields (e.g., [3, 6, 12, 15]). In recent years, some effort has been made to develop versions of the FGT when the ambient dimension d is large [17]. Here, we limit our attention to $d=2,3$, with a focus on issues that arise in solving the diffusion equation

$$\frac{\partial u}{\partial t}(x,t) = \triangle u(x,t) + f(x,t) \quad \text{in} \quad \Omega(t), \tag{1.2}$$

with prescribed Dirichlet, Neumann or Robin data on the boundary $\Gamma(t)$ of the (possibly) moving domain $\Omega(t)$. While there are many approaches to solving (1.2) numerically, we restrict our attention to integral equation based approaches. For illustration, let us consider the Neumann problem with zero initial data. Classical potential theory suggests seeking the solution in the form [11]

$$u(x,t) = \mathcal{V}[f](x,t) + \mathcal{S}[\sigma](x,t) \tag{1.3}$$

where the volume potential $\mathcal{V}$ and single-layer potential $\mathcal{S}$ are defined in terms of the free-space Green's function for the heat equation $G(x,t) = (4\pi t)^{-d/2} e^{-\frac{\|x\|}{4t}}$ as follows:

$$\mathcal{V}[f](x,t) = \int_0^t \int_{\Omega(t)} G(x-y,t-\tau) f(y,\tau)\, dy d\tau, \quad \mathcal{S}[\sigma](x,t) = \int_0^t \int_{\Gamma(t)} G(x-y,t-\tau)\sigma(y,\tau)\, d\Gamma(y) d\tau\,. \tag{1.4}$$

Enforcing the boundary condition results in a second-kind integral equation that needs to be solved for the surface density σ. Once we have solved for σ, we can use (1.3) to evaluate u at any desired target locations in space and time.

Several algorithms [1, 7, 8, 14] have been developed for computing layer and volume heat potentials efficiently. The essential idea (which we describe for layer potentials) is to split

$$\mathcal{S}[\sigma](x,t) = \int_0^{t-\delta} \int_{\Gamma(t)} G(x-y,t-\tau)\sigma(y,\tau)\, d\Gamma(y) d\tau \;+ \int_{t-\delta}^{t} \int_{\Gamma(t)} G(x-y,t-\tau)\sigma(y,\tau)\, d\Gamma(y) d\tau\,. \tag{1.5}$$

[*]Courant Institute of Mathematical Sciences, New York University, New York, NY 10012 (spivak@cs.nyu.edu, shravan@cims.nyu.edu, greengard@courant.nyu.edu). The work of M.S. and L.G. was supported by the Applied Mathematical Sciences Program of the U.S. Department of Energy under grant DEFG0288ER25053. The work of S.V. was supported by the National Science Foundation under grant OCI-0749162.

The first term (referred to as the *history* part) appears to dominate the computational cost, since it is dependent on the the density σ all the way back to $t = 0$. The fast algorithms cited above overcome this obstacle and yield optimal (or nearly optimal) schemes. The second term (referred to as the *local* part) is nonsmooth as $\tau \to t$. While a variety of discretization methods can achieve high-order convergence, it was shown in [13] that care must be taken in order to avoid the phenomenon of *geometrically-induced stiffness* - a loss of resolution for large times steps. More precisely, it was shown that the formal order of accuracy of many schemes was not evident until the time step Δt was of the order Δx^2, where Δx denotes the distance between neighboring spatial grid points. To avoid this difficulty, robust methods were proposed in [13, 16], based on full "product integration" in time. Following [13], a third-order accurate scheme can be obtained by substituting the Taylor expansion of $\sigma(y,t)$ in time

$$\sigma(y,t) = \sigma_0(y) + (t-\tau)\sigma_1(y) + \frac{1}{2}(t-\tau)^2\sigma_2(y) + \mathcal{O}((t-\tau)^3), \quad \tau \in (t-\delta, t) \tag{1.6}$$

into the local part[1]

$$\mathcal{S}_L[\sigma,\delta](x,t) = \int_\Gamma \int_{t-\delta}^{t} G(x-y, t-\tau)\sigma(y,\tau)\, d\Gamma(y) d\tau\,. \tag{1.7}$$

Time integration can then be computed analytically. In two spatial dimensions, the result is

$$\begin{aligned}\mathcal{S}_L[\sigma,\delta](x,t) =& \frac{1}{4\pi}\int_\Gamma \left(\sigma_0(y) - \frac{r^2}{4}\sigma_1(y) + \frac{r^4\delta}{64}\sigma_2(y)\right) \mathrm{Ei}\left(\frac{r^2}{4\delta}\right) d\Gamma(y) + \\ & \frac{1}{4\pi}\int_\Gamma \left(\delta\sigma_1(y) + \frac{\delta^2}{4}\sigma_2(y) - \frac{r^2\delta}{16}\sigma_2(y)\right) e^{-\frac{r^2}{4\delta}}\, d\Gamma(y)\end{aligned} \tag{1.8}$$

where $r = \|x-y\|$ and $\mathrm{Ei}(x)$ is the exponential integral function, which is logarithmically singular. (The first order version of this rule is commonly used in the boundary element literature [2, 4].) The main focus of the present paper is the efficient computation of the convolutions in the second integral. For this particular third-order scheme, we need to compute convolutions with the kernels $e^{-\frac{r^2}{4\delta}}$ and $r^2 e^{-\frac{r^2}{4\delta}}$. For a scheme of order q with $q > 3$, we need to compute all the convolutions

$$G_n[f](x) = \int_\Gamma r^{2n} e^{-\frac{r^2}{4\delta}} f(y)\, d\Gamma(y), \quad 0 \le n \le q-2\,. \tag{1.9}$$

Here, we develop a $\mathcal{O}(q(N+M))$ scheme to compute (1.9) for N sources (discretization points on Γ) and M targets, using both the Hermite [9] and plane-wave [10] versions of the FGT[2]. We also make several improvements to the algorithm, particularly in the extension of the plane-wave scheme to tensor-product grids in dimensions greater than one.

The remainder of the paper is organized as follows. In Section 2, we derive plane-wave representations for the kernels in (1.9) and discuss their approximation properties. While the discretization of volume and surface potentials results in similar discrete sums, we can exploit the underlying structure of the volume discretization if it makes use of a hierarchy of regular grids. (Boundary discretizations, in general, lead to unstructured sets of "sources".) Both cases are discussed in detail in Section 3. Finally, in Section 4, we compare the performance of the new scheme to the older versions and present results on the computational cost of the algorithm for the sequence of Gaussian-type kernels in (1.9). Some notation used throughout the paper is summarized in Table 1.1.

[1]For the sake of simplicity, we assume the domain to be stationary. Moving domain problems result in similar kinds of convolutions. See [13, 16] for more details.

[2]It was noted in [9] that such fast transforms are possible, but the extension was not carried out. It is even possible to compute these convolutions using only the FGT, by expanding r^{2n} and computing several Gauss transforms separately. For example,

$$r^2 e^{-\frac{r^2}{4\delta}} f = (x\cdot x)\int e^{-\frac{r^2}{4\delta}} f - 2x \cdot \int e^{-\frac{r^2}{4\delta}} yf + \int e^{-\frac{r^2}{4\delta}} (y\cdot y) f.$$

However, this is inefficient because one would have to compute $\mathcal{O}(q^2)$ distinct Gauss transforms.

Symbol	Definition	Symbol	Definition
N	number of sources	δ	bandwidth of the Gaussian
M	number of targets	s	size of box in FGT grid
$\lvert B\rvert$	number of FGT boxes	h	order of Hermite expansion
N_B	number of sources per FGT box	C_n	Fourier transform of kernel in (1.9)
K	length of interaction list in each dimension	$2p$	number of plane wave coefficients in each dimension

Table 1.1: *Frequently used symbols.*

2. Mathematical Preliminaries.

In this section, we derive the plane wave expansions of the kernels in (1.9) and give estimates for their truncation errors. We begin by stating the well-known Fourier expansion of the d-dimensional Gaussian:

$$e^{-\|x\|^2} = \left(\frac{1}{2\sqrt{\pi}}\right)^d \int_{\mathbb{R}^d} e^{-\frac{z^2}{4}} e^{izx}\, dz \quad x \in \mathbb{R}^d. \tag{2.1}$$

It is straightforward to develop an analogous formula for the Gaussian-type kernels of interest in the one-dimensional case.

Theorem 2.1. *The Fourier expansion of the kernel $x^{2n}\, e^{-x^2}$ is given by*

$$x^{2n}\, e^{-x^2} = \frac{(-1)^n}{2^{2n+1}\sqrt{\pi}} \int_{-\infty}^{\infty} h_{2n}\left(\frac{z}{2}\right) e^{izx}\, dz \quad n \in \mathbb{N} \tag{2.2}$$

where $h_n(x)$ are Hermite functions defined in terms of the Hermite polynomials $H_n(x)$ by $h_n(x) = e^{-x^2} H_n(x)$.

Proof. The statement for $n = 0$ reduces to the Fourier transform formula (2.1). Let us assume now that the statement is true for n and proceed by induction.

$$\begin{aligned} x^{2(n+1)}\, e^{-x^2} &= x^2 \frac{(-1)^n}{2^{2n+1}\sqrt{\pi}} \int_{-\infty}^{\infty} h_{2n}\left(\frac{z}{2}\right) e^{izx}\, dz \\ &= x^2 \frac{(-1)^n}{2^{2n+1}\sqrt{\pi}} \int_{-\infty}^{\infty} -\frac{1}{2} h_{2n+1}\left(\frac{z}{2}\right) \frac{e^{izx}}{ix}\, dz \\ &\quad \text{(differentiating by parts and using the standard relation} \quad h_n' = -h_{n+1}) \\ &= x^2 \frac{(-1)^n}{2^{2n+1}\sqrt{\pi}} \int_{-\infty}^{\infty} \left(-\frac{1}{2}\right)^2 h_{2n+2}\left(\frac{z}{2}\right) \frac{e^{izx}}{-x^2}\, dz \\ &= \frac{(-1)^{n+1}}{2^{2(n+1)+1}\sqrt{\pi}} \int_{-\infty}^{\infty} h_{2(n+1)}\left(\frac{z}{2}\right) e^{izx}\, dz\,, \end{aligned}$$

completing the proof. □

In order to derive the multi-dimensional expansion of the kernel, we make use of the following multi-index notation (for $k \in \mathbb{R}^d$ and $x \in \mathbb{R}^d$) :

$$\begin{aligned} k! &= k_1! k_2! \ldots k_d! \\ |k| &= \max_i |k_i| \\ \|k\|_1 &= k_1 + k_2 + \ldots + k_d \\ h_k(x) &= h_{k_1}(x_1) h_{k_2}(x_2) \ldots h_{k_d}(x_d) \\ dx &= dx_1 dx_2 \ldots dx_d \end{aligned}$$

THEOREM 2.2. (Multi-dimensional expansion) *Let $x \in \mathbb{R}^d$, then the Fourier expansion of $\|x\|^{2n} e^{-\|x\|^2}$ is given by*

$$\|x\|^{2n} e^{-\|x\|^2} = \int_{\mathbb{R}^d} C_n(z) e^{iz\cdot x}\, dz \tag{2.3}$$

$$\tag{2.4}$$

$$\text{where} \quad C_n(z) = \frac{(-1)^n}{2^{2n+d}\,\pi^{\frac{d}{2}}} \sum_{\|k\|_1=n} \frac{n!}{k!} h_{2k}\left(\frac{z}{2}\right) \tag{2.5}$$

Proof.

$$\begin{aligned}
\|x\|^{2n} e^{-\|x\|^2} &= \left(x_1^2 + x_2^2 + \ldots + x_d^2\right)^n e^{-\|x\|^2} \\
&= \left(\sum_{\|k\|_1=n} \frac{n!}{k!} x_1^{2k_1} x_2^{2k_2} \ldots x_d^{2k_d}\right) e^{-\|x\|^2} \\
&= \sum_{\|k\|_1=n} \frac{n!}{k!} \left(x_1^{2k_1} e^{-x_1^2}\right)\left(x_2^{2k_2} e^{-x_2^2}\right)\ldots\left(x_d^{2k_d} e^{-x_d^2}\right) \\
&= \sum_{\|k\|_1=n} \frac{n!}{k!} \frac{(-1)^{k_1+\ldots+k_d}}{2^{2(k_1+\ldots k_d)+d}(\sqrt{\pi})^d} \int_{-\infty}^{\infty} h_{2k_1}\left(\frac{z_1}{2}\right) e^{iz_1x_1}\, dz_1 \ldots \int_{-\infty}^{\infty} h_{2k_d}\left(\frac{z_d}{2}\right) e^{iz_dx_d}\, dz_d \\
&\quad \text{(from Theorem 2.1)} \\
&= \frac{(-1)^n}{2^{2n+d}\,\pi^{\frac{d}{2}}} \int_{\mathbb{R}^d} \left(\sum_{\|k\|_1=n} \frac{n!}{k!} h_{2k}\left(\frac{z}{2}\right)\right) e^{iz\cdot x}\, dz
\end{aligned}$$

□

THEOREM 2.3. (Truncation error) *The domain of integration of the Fourier expansion in (2.3) can be truncated to the hypercube $[-L, L]^d$ with the following error estimate*

$$\|x\|^{2n} e^{-\|x\|^2} = \int_{[-L,L]^d} C_n(z) e^{iz\cdot x}\, dz + e_T(L) \tag{2.6}$$

$$\text{where} \quad e_T(L) \le \frac{(3.08)^d}{\sqrt{2\pi}} \left(\frac{d}{2}\right)^n \sqrt{(2n)!}\, \text{erfc}\left(\frac{\text{L}}{2\sqrt{2}}\right). \tag{2.7}$$

The proof is outlined in Appendix A. Finally, since the integrand in (2.6) is smooth and exponentially small at the endpoints, the trapezoidal rule converges rapidly [5], yielding the discrete approximation

$$\|x\|^{2n} e^{-\|x\|^2} \approx \left(\frac{L}{p}\right)^d \sum_{|k|\le p} C_n(z_k) e^{iz_k\cdot x} + e_D(p), \quad z_k = \frac{Lk}{p}. \tag{2.8}$$

Rather than writing out a detailed error estimate for $e_D(p)$, we simply list the necessary values of p (the number of quadrature nodes) for different kernels and precisions in Table 2.1. We refer to the discretized approximation (2.8) as the *plane-wave expansion* of the kernel.

3. The Fast Algorithm.

Assuming the layer potentials of (1.9) or the corresponding volume integrals have been discretized by a suitable spatial quadrature formula (a Nyström discretization), the algorithmic issue becomes one of computing discrete sums of the form:

$$G_n[f](x_j) = \sum_{k=1}^{N} \|x_j - y_k\|^{2n} e^{-\frac{\|x_j - y_k\|^2}{\delta}} f_k \quad \text{at} \quad \{x_j \,|\, j = 1, \ldots, N\}. \tag{3.1}$$

ϵ	$n=0$				$n=1$				$n=4$			
	p	h	L	K	p	h	L	K	p	h	L	K
10^{-3}	6	4	5	7	8	4	6	7	12	10	8	11
10^{-6}	10	8	7	9	12	10	8	11	16	12	10	11
10^{-9}	16	12	10	11	19	13	10	13	20	14	11	13
10^{-12}	20	14	11	13	24	16	12	13	24	16	12	13

Table 2.1: Parameter values for various precisions. As in Table (1.1), p is the number of plane-wave coefficients needed per dimension to achieve the stated precision and h is the length of the Hermite expansion needed. L dictates the truncation error defined in Theorem 2.3. K determines the size of the interaction list in each dimension, discussed below in Section 3. $n = 0$ corresponds to the simple Gaussian. $n = 1$ is needed in (1.9) for third order accuracy and $n = 4$ is needed for sixth order accuracy.

Without loss of generality, let us assume that all targets and sources lie within the unit cube $\Omega = [0, 1]^d$. (This is easily accomplished by shifting the origin and rescaling δ as needed.) The fast Gauss transform begins by partitioning Ω into uniform boxes whose linear dimension s is of the order $\mathcal{O}(\sqrt{\delta})$. For each box B, the interaction list is defined as the set of neighboring boxes in which the Gaussian or Gaussian-type kernel has a contribution greater than the desired precision ϵ. We will denote the interaction list for box B by $\mathcal{I}[B]$. Because of the choice of box dimension, and the exponential decay of the kernel, the size of the interaction list is of the order K^d, where the value of K is rather modest (see Table 2.1). We refer the reader to [9] for a more detailed discussion.

Lemma 3.1. *Let B denote a box in the FGT data structure, centered at c^B and containing N_B sources $\{y_j\}$ with strength f_j. Then the Gauss-type field due to those sources is given by*

$$G_n[f](x) = \sum_{|k|\le p} C_n(z_k) w_k e^{iz_k\cdot(x-c^B)/\sqrt{\delta}} \tag{3.2}$$

for any point x lying within the interaction region for B, where

$$w_k = \left(\frac{L}{p}\right)^3 \sum_{y_j\in B} f_j e^{iz_k\cdot(c^B-y_j)/\sqrt{\delta}}, \quad \textit{with} \quad z_k = \frac{kL}{p}\,. \tag{3.3}$$

Lemma 3.2. *Let the plane-wave expansion for box B centered at c^B be given by*

$$G_n[f](x) = \sum_{|k|\le p} C_n(z_k) w_k e^{iz_k\cdot(x-c^B)/\sqrt{\delta}} \tag{3.4}$$

and let D denote a box in B's interaction list centered at c^D. Then $G_n[f](x)$ can be expressed within box D as

$$G_n[f](x) = \sum_{|k|\le p} C_n(z_k) v_k e^{iz_k\cdot(x-c^D)/\sqrt{\delta}} \tag{3.5}$$

where

$$v_k = w_k\, e^{iz_k\cdot(c^D-c^B)/\sqrt{\delta}}, \tag{3.6}$$

These lemmas follow immediately from (2.8).

Remark 1. *A plane-wave expansion of the form (3.5), which contains information transmitted from other boxes, will sometimes be referred to as a "local" plane-wave expansion. This has no particular physical meaning, but makes the organization of the algorithm easier to follow.*

The "new version" FGT [10] proceeds in three steps.

1. **S2W:** For each box B, form the plane-wave expansion induced by the sources it contains according to Lemma 3.1, where p is determined by the desired precision.
2. **W2L:** For each box B, translate the computed plane-wave expansion to every box D in its interaction list using Lemma 3.2 and add it to the *local* expansion for D.
3. **L2T:** For each target x_j, determine the box in which it is located, and evaluate the local expansion at x_j.

We discuss each of these steps and some acceleration techniques in more detail in the remainder of this section.

3.1. Accelerating the computation of the plane-wave expansion. The naive cost of step 1 in the plane-wave based FGT is $O(N(2p)^d)$ work, since each source contributes to each of $(2p)^d$ plane wave coefficients. (A factor of two is easily saved using the fact that they appear in complex conjugate pairs from the quadrature (2.8).) There are two more significant ways, however, to accelerate this step.

3.1.1. Tensor-product grids. If the source locations belong to a tensor-product grid, separation of variables becomes a useful tool. For illustration, suppose that $d = 3$ and that the data is available in box B on an $n \times n \times n$ grid, so that $N_B = n^3$. Then we can form the wave expansions using $\mathcal{O}(pn^3 + p^2n^2 + p^3n)$ operations. Assuming $n > p$, the net cost of forming the plane-wave expansion for B becomes $\mathcal{O}(dpN_B)$. If the global data structure defining the source distribution consists of a tensor-product grid, then the net cost is reduced to $\mathcal{O}(dpN)$ operations.

To see this, suppose the center of box B is located at (x_1^b, x_2^b, x_3^b) and that the n^3 sources lie on the tensor product grid $\{(x_{j_1}, x_{j_2}, x_{j_3}) | j_1, j_2, j_3 = 1, \ldots, n\}$. Our goal is to compute the plane wave coefficients $\{w_{k_1k_2k_3} | k_1, k_2, k_3 = -p, \ldots, p\}$ from the source strengths $\{f_{j_1j_2j_3} | j_1, j_2, j_3 = 1, \ldots, n\}$. We can simply unroll the expansion formation one dimension at a time.

```
STAGE 1
for $j_2, j_3 = 1$ to $n$ do
    $w_{k_1}(j_2, j_3) = \sum_{j_1=1}^{n} f_{j_1j_2j_3} e^{iz_{k_1}(x_1^b - x_{j_1})/\sqrt{\delta}}$   for   $k_1 = -p, \ldots, p$
end for                                                                  $\mathcal{O}(pn^3)$

STAGE 2
for $j_3 = 1$ to $n$ do
    $w_{k_1k_2}(j_3) = \sum_{j_2=1}^{n} w_{k_1}(j_2, j_3) e^{iz_{k_2}(x_2^b - x_{j_2})/\sqrt{\delta}}$   for   $k_1, k_2 = -p, \ldots, p$
end for                                                                  $\mathcal{O}(p^2n^2)$

STAGE 3
$w_{k_1k_2k_3} = \left(\frac{L}{p}\right)^3 \sum_{j_3=1}^{n} w_{k_1k_2}(j_3) e^{iz_{k_3}(x_3^b - x_{j_3})/\sqrt{\delta}}$   for   $k_1, k_2, k_3 = -p, \ldots, p$          $\mathcal{O}(p^3n)$
```

In d dimensions, it is easy to see that the net cost is $\mathcal{O}(dpN)$, assuming again that the number of source points per box is of the same order as the number of expansion coefficients $(n > p)$. This is a standard technique, of course, used in many other contexts including the multidimensional fast Fourier transform.

Remark 2. *We have assumed above that the tensor product grid is uniform. The algorithm depends only on separation of variables and works equally well for nonuniform and adaptive tensor product grids. The only difference is that, within each FGT box, the three stages must be executed separately for each tensor product subgrid.*

3.1.2. Using Hermite expansions. In the general case, including layer potential discretizations, we cannot use the preceding acceleration. From Table 2.1, however, it is clear that the Hermite expansion, which requires $\mathcal{O}(h^dN)$ work [9], is generally less expensive than forming a plane-wave expansion. We can exploit this fact by first forming the Hermite expansion and then converting it to a plane wave expansion by using the following facts.

Theorem 3.3. *Let the Hermite expansion about the center of box B be given by*

$$G_0[f](x) \approx \sum_{|n|\le h} A_n h_n\left(\frac{x-c^B}{\sqrt{\delta}}\right). \tag{3.7}$$

where

$$A_n = \frac{1}{n!}\sum_{j=1}^{N_B} f_j \left(\frac{y_j - c^B}{\sqrt{\delta}}\right)^n \tag{3.8}$$

Then

$$G_0[f](x) \approx \sum_{|k|\le p} w_k e^{-\frac{|z_k|^2}{4}} e^{iz_k\cdot(c^B - x)/\sqrt{\delta}}, \tag{3.9}$$

where

$$w_k = \left(\frac{L}{p}\right)^3 \sum_{|n|\le h} A_n (-i)^{|n|} z_{k_1}^{n_1} z_{k_2}^{n_2} \dots z_{k_d}^{n_d}. \tag{3.10}$$

This result is easily obtained by substituting the Fourier expansion (2.2) in the definition of the Hermite function $h_n(x) = (-1)^n \frac{d^n}{dx^n} e^{-x^2}$.

The naive cost of conversion from Hermite to plane-wave form using (3.10) is $\mathcal{O}\left(\frac{1}{2}(2p)^d h^d\right)$ for every FGT box. We can reduce this cost further by evaluating separately in each dimension:

1. $w_{k_1}(n_2, n_3) = \sum_{n_1=0}^{h} A_{n_1 n_2 n_3}(-iz_{k_1})^{n_1}$ `for` $n_2, n_3 = 0, \dots, h$ `and` $k_1 = 0, \dots, p$ $\quad \mathcal{O}(h^3 p)$
2. $w_{k_1 k_2}(n_3) = \sum_{n_2=0}^{h} w_{k_1}(n_2, n_3)(-iz_{k_2})^{n_2}$ `for` $n_3 = 0, \dots, h$ `and` $k_1, k_2 = 0, \dots, p$ $\quad \mathcal{O}(h^2 p^2)$
3. $w_{k_1 k_2 k_3} = \left(\frac{L}{p}\right)^3 \sum_{n_1=0}^{h} w_{k_1 k_2}(n_3)(-iz_{k_3})^{n_3}$ `for` $k_1, k_2, k_3 = -p, \dots, p$ $\quad \mathcal{O}(hp^3)$

Since $p > h$, the overall computational cost of forming the plane-wave coefficients at all the FGT boxes is $\mathcal{O}(h^d N + dp^d h|B|)$. Compared to a direct scheme that evaluates (3.3), this algorithm leads to significant computational savings when the number of sources per box is large. In practice, we found that it compares favorably in almost all cases of interest (in 3D).

3.1.3. Computing the action of multiple kernels. An advantage of the plane wave representation is that the plane wave expansion defined in (3.3) is *independent* of the kernel. The translation operator (3.6) is also independent of the kernel. Because of the nature of the Fourier transform, the particular choice of n that defines the kernel appears only in the local expansion (3.5). Therefore, if multiple Gaussian-type transforms are being carried out on the same function f, all of them can be computed with just one S2W step and one W2L step.

In the case of our quadrature formula for heat potentials, the second term of equation (1.8) can be written as

$$G_0\left[\delta\sigma_1(y) + \frac{\delta^2}{2}\sigma_2(y)\right] + G_1\left[-\frac{\delta}{8}\sigma_2(y)\right] \tag{3.11}$$

which involves multiple kernels acting on distinct source distributions. In such cases, instead of computing the action of each kernel separately, the effect of all sources can be represented using a single plane wave expansion.

Lemma 3.4. *Let kernels $r^{2n}e^{-\frac{r^2}{4\delta}}, 0 \le n \le q$ act on sources y_j with strengths f_j^n, lying in a box B with center c^B. Then the Gauss-type field due to those sources is given by*

$$\sum_{0\le n\le q} G_n[f^n](x) = \sum_{|k|\le p}\left(\sum_{0\le n\le q} C_n(z_k) w_k^n\right) e^{iz_k\cdot(x-c^B)/\sqrt{\delta}} \tag{3.12}$$

for any point x within the interaction region of B, where

$$w_k^n = \sum_{y_j \in B} f_j^n e^{iz_k\cdot(c^B - y_j)/\sqrt{\delta}} \quad \textit{and} \quad z_k = \frac{Lk}{p}\,. \tag{3.13}$$

The result is easily obtained using Lemma 3.1.

If the source expansions w_k^n are computed using the method in 3.1.2, the overall computational cost of forming the expansions is $\mathcal{O}(q(h^d N_B + dp^d h|B|))$. It should be noted, however, that if the discretization parameters (L, p, K) differ greatly for the various kernels, the cost will be dominated by the worst case parameter set.

3.2. Accelerating the plane-wave translation step. Once the plane-wave expansions are formed for each of the FGT boxes, step 2 involves translating the information to its interaction list to obtain the *local* expansions. A direct scheme forms the local expansions by simply visiting all the boxes in $\mathcal{I}[B]$ for each box B and translating the wave expansions according to Lemma 3.2. Since the size of the interaction list is K^d, this algorithm requires $\mathcal{O}(K^d p^d |B|)$ work to form all local expansions, where $|B|$ is the total number of boxes. In the "new version" FGT [10], it was observed that a *sweeping algorithm* could be used to accelerate the translation step. The idea is straightforward and illustrated in Fig. 3.1.

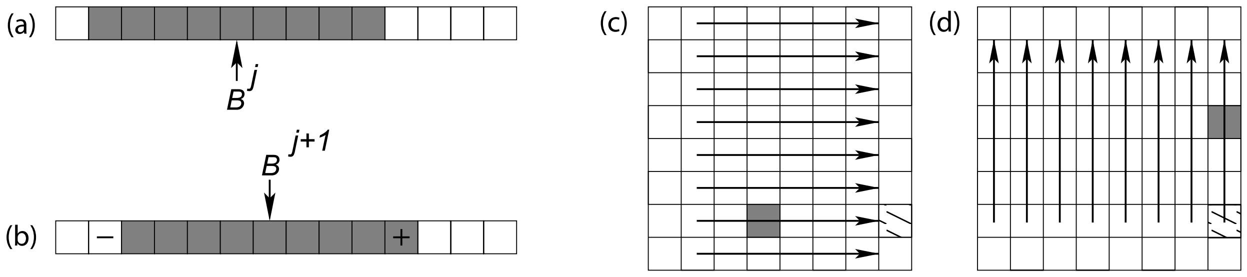

Fig. 3.1: *(a),(b): Accelerated translations in one dimension. Initialization: Compute the local expansion for the leftmost box (B^1) by directly translating the plane-wave expansion from each box in $\mathcal{I}[B^1]$. This requires $\mathcal{O}(\frac{K}{2}p)$ work since only the $\frac{K}{2}$ boxes to the right lie within the FGT grid. Sweeping: For $j = 2, ..., |B|$, the local expansion for box B^j can be shifted to the center of B^{j+1} at the cost of one translation. This is almost the local expansion for B^j. By inspection of the full interaction lists $\mathcal{I}[B^{j-1}]$ and $\mathcal{I}[B^j]$, indicated in gray, it is clear that one must simply subtract the unneeded contribution from the leftmost member of $\mathcal{I}[B^{j-1}]$ (marked by a $-$) and add the needed contribution from the rightmost member of $\mathcal{I}[B^j]$ (marked by a $+$). Boxes that fall outside the FGT grid can obviously be ignored. The overall cost of translating the wave expansions is reduced from $\mathcal{O}(Kp|B|)$ to $\mathcal{O}(3p|B|)$. It should be noted that this procedure depends strongly on the diagonal form of the plane-wave basis. Sweeping based on Hermite expansions, for example, would rapidly lose accuracy. (c), (d): In higher dimensions, one simply repeats the above procedure in each coordinate direction. To understand how information is propagated, consider the gray "source" box in (c) and the gray "target" box in (d). Information from the source box is transmitted to the dashed box in the first sweep. In the second sweep, it is transmitted to the target.*

Remark 3. *Since the contributions from distant boxes are negligible, it may not be immediately obvious why one needs to subtract the influence from the box marked with a "$-$" in the procedure outlined in Fig. 3.1. The answer has to do with the plane-wave representation itself. We are representing the smooth field induced by Gaussian or Gaussian-like sources in terms of* oscillatory *complex exponentials. The quadrature formula (2.8) is subject to aliasing errors if the translation distance exceeds the range covered by the interaction list, resulting in $O(1)$ errors.*

The multidimensional version consists of a sequence of d one-dimensional sweeps, for which we introduce some notation. We let the FGT boxes be numbered B^j where $j = (j_1, j_2, \ldots, j_d)$ with $\{1 \le j_k \le |B|^{1/d}\}_{k=1}^d$, and we let $n = \lfloor \frac{K}{2} \rfloor$. For a box B^j, the interaction list then consists of the d-dimensional cube centered at B^j and extending from $-n$ to n in each dimension. An informal description of the sweeping algorithm follows:

```
MULTI-DIMENSIONAL SWEEPING ALGORITHM
1) Sweep along first dimension
for $j_2, \dots, j_d = 1$ to $|B|^{1/d}$ do
  for $j_1 = 2$ to $n$ do
    Translate expansion from $B^{(j_1, j_2, \dots, j_d)}$ to $B^{(1, j_2, \dots, j_d)}$ according to Lemma 3.2.
  end for
  for $j_1 = 1$ to $|B|^{1/d} - 1$ do
    Translate local expansion from $B^{(j_1, j_2, \dots, j_3)}$ to $B^{(j_1+1, j_2, \dots, j_d)}$, subtract contribution from $B^{(j_1+1-n, j_2, \dots, j_d)}$,
    and add contribution from $B^{(j_1+1+n, j_2, \dots, j_d)}$, all according to Lemma 3.2. (Ignore contributions if the range
    of the index falls outside the FGT grid.)
  end for
end for
2) Repeat analogous loop for each dimension $2, \dots, d$.
```

It is straightforward to see why the multi-dimensional algorithm gathers the necessary local expansions for all boxes. The first sweep collects information along one dimension; each box accumulates expansions from K members of its interaction list. The second sweep propagates that information along the second dimension, with each box accumulating expansions from K^2 members of its interaction list. This continues until the dth sweep, at which point all K^d expansions have been translated. In terms of operation count, the net cost is reduced from $\mathcal{O}(K^d p^d |B|)$ to $\mathcal{O}(3dp^d|B|)$.

3.3. Accelerating the evaluation of local expansions. Once the local coefficients are formed at all the FGT boxes, the convolution (1.9) can be computed at the target locations by a direct evaluation of the local expansion:

$$G_n[f](x_j) = \sum_{|k| \le p} C_n(z_k) v_k e^{i z_k \cdot (x_j - c^B)/\sqrt{\delta}} \quad \forall \quad x_j \in B. \tag{3.14}$$

The computational cost of this operation is $\mathcal{O}(p^d N)$. As in the S2W step (Step 1), we can accelerate this calculation in two ways: (i) if the targets belong to a tensor-product grid, then the cost can be reduced to $\mathcal{O}(dpN)$ using separation of variables; (ii) in the general case, we can convert the local plane-wave expansion into a Taylor expansion and evaluate the Taylor series at the targets, thereby, reducing the cost to $\mathcal{O}(h^d N + dp^d h|B|)$

LEMMA 3.5. *Let the local plane-wave expansion for box D centered at c^D be given by*

$$G_n[f](x) \approx \sum_{|k| \le p} C_n(z_k) v_k e^{i z_k \cdot (x - c^D)/\sqrt{\delta}} \quad \text{where} \quad z_k = \frac{Lk}{p}. \tag{3.15}$$

Then the Taylor expansion of $G_n[f](x)$ about c^D is

$$G_n[f](x) \approx \sum_{|m| \le h} T_m (x - c^D)^m \tag{3.16}$$

where

$$T_m = \frac{1}{m!} \sum_{|k| \le p} C_n(z_k) w_k \left(i \frac{z_{k_1}}{\sqrt{\delta}} \right)^{m_1} \left(i \frac{z_{k_2}}{\sqrt{\delta}} \right)^{m_2} \dots \left(i \frac{z_{k_d}}{\sqrt{\delta}} \right)^{m_d} \tag{3.17}$$

and h is the order of Taylor approximation.

This result is obtained by substituting the Taylor expansion about c^D

$$e^{\frac{i z_k (x - c^D)}{\sqrt{\delta}}} = \sum_{|m| \le h} \frac{1}{m!} \left(i \frac{z_k}{\sqrt{\delta}} \right)^m (x - c^D)^m \tag{3.18}$$

into 3.15. As in the conversion from Hermite expansions to the plane wave basis (Section 3.1.2), the cost per box of converting a local plane wave expansion to a Taylor expansion can be reduced from $\mathcal{O}(p^d h^d)$ to $\mathcal{O}(dp^d h)$ by separation of variables.

3.4. Summary of Computational Costs. We summarize the computational costs of the various schemes discussed here in Table 3.1.

Method	Steps 1/3 (Formation/Evaluation)	Step 2 (Translation)
(i) Hermite	$h^d N$	$dh^{d+1}K^d\|B\|$
(ii) PW	$\frac{1}{2}(2p)^d N$	$3dp^d\|B\|$
(iii) H–PW	$h^d N + dp^d h\|B\|$	$3dp^d\|B\|$
Tensor-product grids		
(iv) Hermite	dhN	$dh^{d+1}K^d\|B\|$
(v) PW	dpN	$3dp^d\|B\|$
(vi) H–PW	$dhN + dp^d h\|B\|$	$3dp^d\|B\|$

Table 3.1: Operation count for variants of FGT. Method (i) is the original FGT [9] and (ii) is the "new-version" FGT [10]. Methods (iii) to (vi) incorporate accelerations introduced in this section. Methods (i) and (iv) translate Hermite expansions while the rest translate plane wave expansions. H–PW indicates that a plane wave expansion is computed by first forming the Hermite expansion and then using Theorem 3.3.

4. Results. In this section, we present some timing results for a Fortran implementation of the various FGT algorithms in three dimensions, performed on a laptop with an Intel® 1.3GHz processor and 3.9GB RAM.

In all the test cases, we assume that the sources and targets lie in the unit cube. We set the FGT box size to be $\sqrt{\delta}$, so that $|B| = \left(\frac{1}{\sqrt{\delta}}\right)^3$. The parameters p, h, L and K are dictated by the user specified accuracy ϵ according to Table 2.1. Thus, there are only three independent parameters in the algorithm, namely, N, ϵ and δ. The Gaussian width, determined by δ sets the number of boxes in the FGT data structure, as indicated above. Since we are primarily motivated by fast solvers for the heat equation, we let $\delta = \Delta t = \mathcal{O}(\Delta x) = \mathcal{O}(N^{1/3})$ for a regular three-dimensional grid with N points. This corresponds to a large time step, for which finite difference and finite element discretizations would require implicit marching schemes.

4.1. Gauss transform. Let us begin by analyzing the computational cost of the standard Gauss transform for three test cases with different source distributions. In the first case, we consider the calculation of

$$\phi(x) = \int_\Omega e^{-\frac{|x-y|^2}{\delta}} f(y)\, dy, \quad \Omega = [0,1]^3, \tag{4.1}$$

discretized using a Nyström method based on tensor product Gaussian quadrature. In Table 4.1, we list the CPU times for the current scheme (method *vi* of Table 3.1) and an accelerated version of the original FGT (method *iv*). In the second case, we consider a random uniform distribution of sources, with results presented in Table 4.2. From this data, it is clear that forming expansions is much faster on tensor-product grids and that translating plane wave expansions is much faster than translating Hermite expansions, consistent with the complexity analysis in the previous section. The extent of the acceleration depends, of course, on the problem size, number of boxes, desired precision, etc. When the number of boxes is small (in the extreme case, just one), the original FGT is faster because its expansion cost is lower. As soon as translation costs dominate, the plane wave-based methods outperform the original FGT.

In our third test, we consider a typical computational task encountered in solving the inhomogeneous heat equation via potential theory in a complicate domain. At every time-step, we need to evaluate the function

$$u_L(x,t) = \mathcal{S}_L[\sigma,\delta](x,t) + \mathcal{V}_L[f,\delta](x,t)\,, \tag{4.2}$$

the sum of a volume and a layer potential:

$$u_L(x,t) = \int_\Omega e^{-\frac{|x-y|^2}{\delta}} f(y)\,d\Omega(y) \;+\; \int_\Gamma e^{-\frac{|x-y|^2}{\delta}} \sigma(y)\,d\Gamma(y). \tag{4.3}$$

The domain Ω is assume to be the unit cube while the surface Γ is shown in Figure 4.1. CPU times for various volume grid and boundary discretizations are listed in Table 4.3.

Finally, in Figure 4.2, we compare the performance of the tensor-product FGT with the FFT as a function of problem size for a variety of precision values.

δ	$N = M$	FGT (version *vi*)			Original FGT (version *iv*)			Direct
		Total	Steps 1,3 (S2W,L2T)	Step 2 (W2L)	Total	Steps 1,3 (S2H,L2T)	Step 2 (H2L)	
0.01	1000000	2.6	1.7	0.9	107	0.3	106	1.7e5
0.007	3375000	5.4	3.5	1.9	229	0.6	228	2.1e6
0.005	8000000	11	8	3	517	1.6	515	3.3e7
0.004	15625000	16	10	6	654	2.8	651	5.2e7
0.003	27000000	27	19	8	1207	5	1199	1.2e8
0.0025	64000000	39	30	9	1445	10	1432	7.7e8

Table 4.1: *CPU times for the accelerated plane-wave FGT and an accelerated version of the original FGT, with precision set to $\epsilon = 10^{-7}$. The sources and targets are the tensor product Gaussian quadrature nodes scaled to $[0,1]^3$. For both schemes, note that the expansion costs increase linearly with N. The translation costs increase more slowly, since $|B| \propto \sqrt{N}$ ($\delta \propto N^{-1/3}$).*

δ	$N = M$	FGT (version *iii*)			Original FGT (version *i*)			Direct
		Total	Steps 1,3 (S2W,L2T)	Step 2 (W2L)	Total	Steps 1,3 (S2H,L2T)	Step 2 (H2L)	
0.01	1000000	15	13.6	1.2	124	12	112	1.7e5
0.007	3375000	49	45	1.7	275	44	275	2.5e6
0.005	8000000	113	108	4	640	106	534	3.0e7
0.004	15625000	232	224	6	882	207	675	4.8e7
0.003	27000000	395	383	8	1609	345	1247	1.3e8

Table 4.2: *CPU times for new and original fast Gauss transforms with sources and targets randomly distributed.*

4.2. Generalized Gauss transform. As a final example, we consider the computation of volume heat potentials with high order accuracy in time. In three dimensions, this requires the evaluation of

$$\phi(x) = \frac{1}{(\sqrt{4\pi})^3}\int_\Omega e^{-\frac{r^2}{4\delta}}\left[2\,f_1(y)\,\sqrt{\delta} + \frac{1}{3}\,f_2(y)\,\delta^{3/2}\left(1 - \frac{1}{2}\frac{r^2}{\delta}\right)\right]\,dy \tag{4.4}$$

for third order accuracy and

$$\begin{aligned}\psi(x) = \phi(x) + \frac{1}{(\sqrt{4\pi})^3}\int_\Omega e^{-\frac{r^2}{4\delta}}\left[\frac{1}{15}\,f_3(y)\,\delta^{5/2}\left(\frac{1}{12}\frac{r^4}{\delta^2} - \frac{1}{6}\frac{r^2}{\delta} + 1\right) + \frac{1}{84}\,f_4(y)\,\delta^{7/2}\right.\\ \left.\left(-\frac{1}{120}\frac{r^6}{\delta^3} + \frac{1}{60}\frac{r^4}{\delta^2} - \frac{1}{10}\frac{r^2}{\delta} + 1\right) + \frac{1}{540}\,f_5(y)\,\delta^{9/2}\left(\frac{1}{1680}\frac{r^8}{\delta^4} - \frac{1}{840}\frac{r^6}{\delta^3} + \frac{1}{140}\frac{r^4}{\delta^2} - \frac{1}{14}\frac{r^2}{\delta} + 1\right)\right]\,dy\end{aligned} \tag{4.5}$$

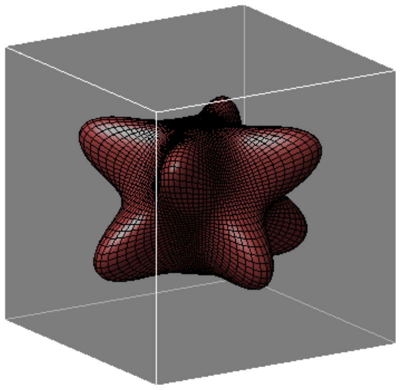

Fig. 4.1: *Test domain for Table 4.3. In our third test, we set the sources and targets to be the tensor product Gaussian quadrature nodes in the cube* $[0,1]^3$ *(discretizing the volume potential), augmented by a set of discretization nodes on the surface* Γ.

δ	N = M		New FGT (versions iii, vi)			Original FGT (version i)			Direct
	volume	surface	Total	S2W+L2T	W2L	Total	S2H+L2T	H2L	
0.01	1000000	80100	3.9	3	0.9	108	1	107	1.7e5
0.007	3375000	180600	8.4	6.5	1.9	225	3	222	2.1e6
0.005	8000000	320800	17	13	3.5	520	5	514	3.3e7
0.004	15625000	501000	25	20	5	660	9	650	5.0e7
0.003	27000000	721200	41	33	8	1215	17	1198	1.2e8

Table 4.3: *CPU times for the new and original fast Gauss transforms, for precision* $\epsilon = 10^{-7}$. *In the new scheme, version iii is used for the irregular boundary nodes, while version vi is used for the regular volume nodes. The source distributions are described in Figure 4.1.*

for sixth order accuracy, where

$$f(y,t) = f_1(y) + (t-\tau)f_2(y) + \frac{1}{2}(t-\tau)^2 f_3(y) + \cdots + \frac{1}{5!}(t-\tau)^5 f_5(y) + \mathcal{O}((t-\tau)^6)\,. \tag{4.6}$$

The sources and targets are assumed to be tensor product Gaussian nodes in the unit cube, and we report CPU times in Table 4.4. Note that in each case, the distinct sources and source types can be combined into a single plane wave expansion using Lemma 3.4. For the sake of simplicity, we have set the parameters p, L, K so that the Gaussian is approximated with an error $\epsilon = 10^{-7}$ ($n = 0$ in Table 2.1). We use these parameters for all kernels, leading to some loss in accuracy. In particular, we obtain six-digit accuracy for the third order kernels and five-digit accuracy for the sixth order kernels. As indicated in Table 2.1, higher values of p, L, K can be chosen to guarantee any desired ϵ.

5. Conclusions. We have presented several methods for accelerating the fast Gauss transform and a framework for its generalization to convolution with Gaussian-type kernels. The translation costs are dramatically reduced using plane-wave representations (as in [10]), and the expansion formation costs are dramatically reduced for (locally) tensor product data structures. These techniques are of particular importance in evaluating volume potentials on hierarchies of adaptive grids. For pure layer potentials, where the sources and targets lie on a lower dimensional manifold, the gain is more modest.

Finally, the algorithms described in this paper are applicable even when the kernel and/or its Fourier

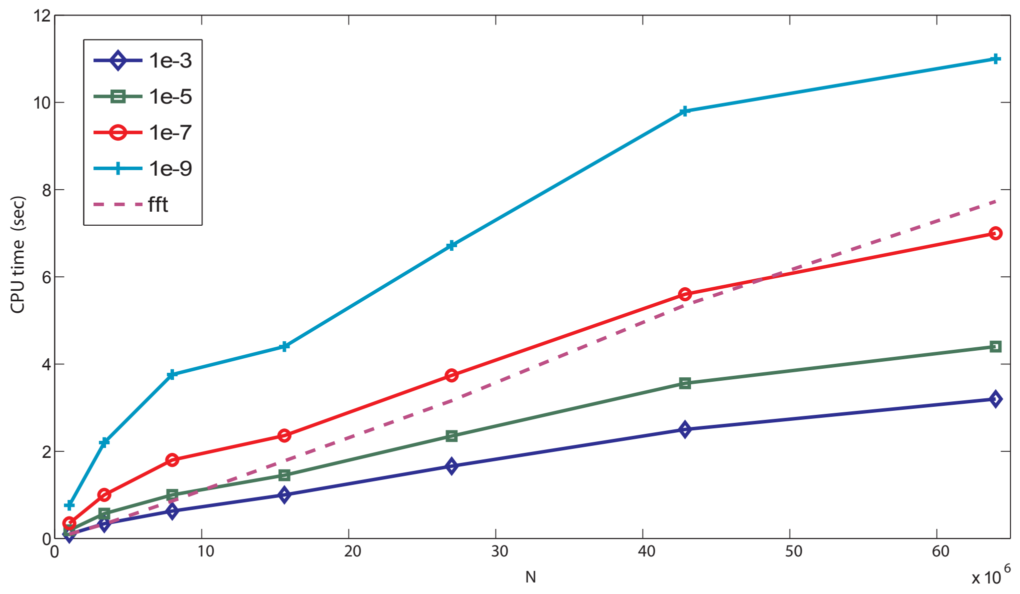

Fig. 4.2: *CPU times for the accelerated FGT (version vi), with sources distributed on a tensor-product grid. The timings for the three-dimensional FFT are shown (dashed curve) as a benchmark. We set $\delta = \frac{4}{N^{1/3}}$, modeling an implicit time step in a marching scheme for the heat equation. The choice of δ has a significant effect on timing. Smaller values of δ increase the cost (while still proportional to N). In the case of modeling heat flow, high order discretization in time should allow for larger time steps and, therefore, lower cost Gauss transforms.*

δ	$N = M$	First order		Third order		Sixth order	
		Total	S2W+L2T	Total	S2W+L2T	Total	S2W+L2T
0.01	1000000	2.7	1.7	3.7	2.6	6	5
0.007	3375000	5.5	3.7	7.4	5.6	13	11
0.005	8000000	12	8	15	12	27	24
0.004	15625000	16	12	22	18	39	34
0.003	27000000	27	19	37	29	66	58

Table 4.4: *CPU times for the fast Gauss transform, the convolutions in (4.4), and the convolutions in (4.5). These correspond to first, third and sixth order accuracy. In each case, the influence of distinct source strengths is combined into just one plane wave expansion using Lemma 3.4. Thus, the number of translations required in each case is the same, resulting in extremely efficient high order schemes.*

transform are not known analytically. The only requirement is that the kernel should be smooth, and approximately space-limited and band-limited. Needed are the values of the Fourier transform of the kernel at the plane-wave discretization points - the analog of the values $C_n(z_k)$ in (2.8). These can be precomputed numerically by means of the discrete Fourier transform of the kernel sampled on a sufficiently fine mesh. The remaining steps are *kernel-independent*.

Appendix A. Truncation error estimate.

Proof. Cramer's inequality

$$|H_n(t)| \le 1.09\, 2^{n/2}\sqrt{n!}\, e^{t^2/2} \tag{A.1}$$

implies that

$$\left| h_{2n}\left(\frac{z}{2}\right)\right| \le 1.09\, 2^n \sqrt{(2n)!}\, e^{-\frac{z^2}{8}} . \tag{A.2}$$

Consequently, we have the following estimates in 1D

$$\left|\int_0^L h_{2n}\left(\frac{z}{2}\right) e^{izx} dz\right| \le 1.09\, 2^n \sqrt{(2n)!}\, \sqrt{2\pi} \quad (\text{using} \quad \int_0^\infty e^{-\frac{z^2}{8}}\, dz = \sqrt{2\pi}) \tag{A.3}$$

$$\left|\int_L^\infty h_{2n}\left(\frac{z}{2}\right) e^{izx} dz\right| \le 1.09\, 2^n \sqrt{(2n)!}\, \operatorname{erfc}\left(\frac{L}{2\sqrt{2}}\right) \tag{A.4}$$

In the multi-dimensional case, we have

$$e_T(L) = \frac{1}{2^{2n+d}\pi^{d/2}} \left| \int_{[-\infty,\infty]^d \setminus [-L,L]^d} \sum_{\|k\|_1=n} \frac{n!}{k!} h_{2k}\left(\frac{z}{2}\right)\, dz \right| \tag{A.5}$$

$$= 2^d\, \frac{1}{2^{2n+d}\pi^{d/2}} \sum_{\|k\|_1=n} \frac{n!}{k!} \left| \int_{[0,\infty]^d \setminus [0,L]^d} h_{2k}\left(\frac{z}{2}\right)\, dz \right| . \tag{A.6}$$

Since the integrand is a separable function, we can integrate in each dimension separately. The intervals of integration are all possible combinations of $[0, L]$ and $[L, \infty]$ except the case where all of them are $[0, L]$. Let $\beta = \sqrt{2\pi}$ and $\alpha = \operatorname{erfc}\left(\frac{L}{2\sqrt{2}}\right)$. Then, using (A.3) and (A.4), the truncation error can be derived as

$$e_T(L) \le \left(\frac{1}{2^{2n}\pi^{d/2}}\right) 1.09^d\, 2^n \left(\sum_{\|k\|_1=n} \frac{n!}{k!}\sqrt{(2k)!}\right) \left(\alpha^d + \binom{d}{1}\alpha^{d-1}\beta + \ldots + \binom{d}{d-1}\alpha\beta^{d-1}\right) . \tag{A.7}$$

Since $\operatorname{erfc}(\cdot) \le 1$, we have $\beta > \alpha$. Using this and the identity $\sum_{\|k\|_1=n} \frac{n!}{k!} = d^n$, we can simplify the estimate further:

$$e_T(L) \le \left(\frac{1}{2^{2n}\pi^{d/2}}\right) 1.09^d\, 2^n\, d^n\, \sqrt{(2n)!}\, 2^d \alpha\beta^{d-1} \tag{A.8}$$

$$= \frac{(3.08)^d}{\sqrt{2\pi}} \left(\frac{d}{2}\right)^n \sqrt{(2n)!}\, \operatorname{erfc}\left(\frac{L}{2\sqrt{2}}\right) . \tag{A.9}$$

□

REFERENCES

[1] K. Brattkus and D. I. Meiron. Numerical simulations of unsteady crystal-growth. *SIAM Journal On Applied Mathematics*, 52:1303–1320, 1992.

[2] C. A. Brebbia, J. C. F. Telles, and L. C. Wrobel. *Boundary Element Techniques*. Springer, 1984.

[3] Mark Broadie and Yusaku Yamamoto. Application of the fast Gauss transform to option pricing. *Management Science*, 49(8):1071–1088, 2003.

[4] G. F. Dargush and P. K. Banerjee. Application of the boundary element method to transient heat conduction. *International Journal of Numerical Methods in Engineering*, 31:1231–1247, 1991.

[5] P. J. Davis and P. Rabinowitz. *Methods of Numerical Integration*. Academic Press, San Diego, 1984.

[6] Ahmed Elgammal, Ramani Duraiswami, and Larry S. Davis. Efficient kernel density estimation using the fast Gauss transform with applications to color modeling and tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1499–1504, 2003.

[7] L. Greengard and P. Lin. Spectral approximation of the free–space heat kernel. *Applied and Computational Harmonic Analysis*, 9:83–97, 2000.

[8] L. Greengard and J. Strain. A fast algorithm for the evaluation of heat potentials. *Communications on Pure and Applied Mathematics*, XLIII:949–963, 1990.

[9] L. Greengard and J. Strain. The fast Gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 12(1):79–94, 1991.

[10] Leslie Greengard and Xiaobai Sun. A new version of the fast Gauss transform. *Documenta Mathematica*, III:575–584, 1998.

[11] R. B. Guenther and J. W. Lee. *Partial Differential Equations of Mathematical Physics and Integral Equations*. Prentice-Hall, 1988.

[12] Junmo Kim, III Fisher, J.W., A. Yezzi, M. Cetin, and A.S. Willsky. A nonparametric statistical method for image segmentation using information theory and curve evolution. *Image Processing, IEEE Transactions on*, 14(10):1486 –1502, oct. 2005.

[13] Jing-Rebecca Li and Leslie Greengard. High order accurate methods for the evaluation of layer heat potentials. *SIAM Journal on Scientific Computing*, 31(5):3847–3860, 2009.

[14] Johannes Tausch. A fast method for solving the heat equation by layer potentials. *Journal of Computational Physics*, 224(2):956–969, 2007.

[15] Shravan K. Veerapaneni and George Biros. The Chebyshev fast Gauss and nonuniform fast fourier transforms and their application to the evaluation of distributed heat potentials. *Journal of Computational Physics*, 227:7768–7790, 2008.

[16] Shravan K. Veerapaneni and George Biros. Arbitrary-order accurate schemes for computing boundary heat potentials. *Xxxxx, preprint*, 2009.

[17] Changjiang Yang, Ramani Duraiswami, Nail Gumerov, and Larry S. Davis. Improved fast Gauss transform and efficient kernel density estimation. *Proceedings of Ninth IEEE International Conference on Computer Vision*, pages 664–671, 2003.