The Fast Gauss Transform with Variable Scales [*]

John Strain [†]
Courant Institute of Mathematical Sciences
251 Mercer Street
New York, NY 10012

20 April 1990

To appear in *SIAM J. Sci. Stat. Comput.*

Keywords: Gauss Transform, Fast Algorithms, Hermite Polynomials.

AMS(MOS) Subject Classifications: 65R10,33A65,35K05,62G20,65P05.

Electronic Mail: strain@math.princeton.edu

Running Head: The Fast Gauss Transform

[*]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, Princeton, NJ 08544

Abstract

This paper presents a fast algorithm which evaluates a d-dimensional Gaussian convolution sum with scales which vary from point to point. This algorithm evaluates the sum of N Gaussians at M arbitrarily distributed points in $C(N+M)$ work, where C depends only on the precision required and the essential minimum of the scales. It achieves a speedup of 1776 with $d=2$, $N=M=100,000$ random uniformly distributed points in $[0,1]^2$, and scales bounded below by 1/100, and breaks even at less than 50 points.

1 Introduction

Mnay numerical calculations require the evaluation of d-dimensional Gaussian convolution sums having one of the two forms

$$Sf(t_i) = \sum_{j=1}^{N} f_j e^{-|t_i - s_j|^2/\delta_j} \tag{1}$$

or

$$Tf(t_i) = \sum_{j=1}^{N} f_j e^{-|t_i - s_j|^2/\delta_i} \tag{2}$$

at a collection of M points $t = t_i \in \mathbf{R}^d$. Here s_j are given points in $\mathbf{R}^d$, $|x|^2 = x_1^2 + \cdots + x_d^2$, and δ_i are given positive real numbers with a lower bound δ_m. The difference between (1) and (2) is that in (1) the "scale" δ depends on the source index j, while in (2) δ depends on the target index i.

Since this calculation requires the application of an $M \times N$ matrix to an $N-$vector, it would seem to require $O(MN)$ computational work, making it CPU-intensive when M and N are large. Since we begin with $O(N)$ data and get $O(M)$ output values, it is natural to seek an algorithm which requires only $O(N+M)$ work. Fortunately, the matrix has enough structure to make this possible.

This paper presents an algorithm for evaluating (1) and (2) in $O(M+N)$ time, with the constant factor in $O(N+M)$ depending only on the precision desired and the lower bound δ_m of the scales δ_i or δ_j. Our algorithm is an extension of the scheme presented in [5], which treated the special case when $\delta_j = \delta$ are all the same. It combines Taylor or Hermite expansion with scaling and direct evaluation, after organizing either sources or targets into a convenient data structure. The algorithms for source-dependent (1) and target-dependent (2) scales are similar in organization.

§2 of the paper reviews the necessary facts about Hermite expansions, §3 presents the source-dependent scale algorithm and §4 treats the target-dependent case. §5 presents numerical examples which show that our algorithm is much faster than direct evaluation for large-scale problems, achieving a speedup of 2400 when $N = M = 100,000$ and the scales are not too small. §6 presents our conclusions.

2 Hermite expansions

This section describes the properties of Hermite expansions we will need. Good references for this material are [1, 2, 4] and particularly Hille's classical paper [6].

The Hermite polynomials $H_n(t)$ are defined by the Rodrigues formula

$$H_n(t) = (-1)^n e^{t^2} D^n e^{-t^2} \qquad\qquad t \in \mathbf{R}$$

where $D = d/dt$. This implies the generating function

$$e^{2ts-s^2} = \sum_{n=0}^{\infty} \frac{s^n}{n!} H_n(t) \ .$$

Multiplying through by e^{-t^2} gives

$$e^{-(t-s)^2} = \sum_{n=0}^{\infty} \frac{s^n}{n!} h_n(t),$$

where the Hermite functions $h_n(t)$ are defined by

$$h_n(t) = e^{-t^2} H_n(t). \tag{3}$$

(Note that these are not the usual orthonormal Hermite functions; these Hermite functions form a biorthogonal set with Hermite polynomials, rather than themselves forming an orthogonal set.) In practice, we will use a shifted and scaled version of this formula: for $s_0 \in \mathbf{R}$ and $\delta > 0$, we have

$$\begin{aligned} e^{-(t-s)^2/\delta} &= e^{-(t-s_0-(s-s_0))^2/\delta} \\ &= \sum_{n=0}^{\infty} \frac{1}{n!} \left(\frac{s-s_0}{\sqrt{\delta}}\right)^n h_n\left(\frac{t-s_0}{\sqrt{\delta}}\right) \\ &= e^{-(t-s_0)^2/\delta} \sum_{n=0}^{\infty} \frac{1}{n!} \left(\frac{s-s_0}{\sqrt{\delta}}\right)^n H_n\left(\frac{t-s_0}{\sqrt{\delta}}\right). \end{aligned}$$

This formula tells us how to evaluate the Gaussian field $e^{-(t-s)^2/\delta}$ at the *target* t due to the *source* at s, as an Hermite expansion centered at s_0. Thus we can shift a Gaussian centered at s to a sum of Hermite polynomials times a Gaussian, all centered at s_0.

We can also interchange t and s to write

$$e^{-(t-s)^2/\delta} = \sum_{n=0}^{\infty} \frac{1}{n!} h_n\left(\frac{s-t_0}{\sqrt{\delta}}\right)\left(\frac{t-t_0}{\sqrt{\delta}}\right)^n . \tag{4}$$

Looked at this way, the expansion expresses a Gaussian with target t as a Taylor series about a nearby target t_0; the coefficients of the Taylor series are the Hermite functions evaluated at t_0. Thus the same expansion serves as both a near-field (Taylor) and a far-field (Hermite) expansion.

The final one-dimensional results we will need are the recurrence relation

$$h_{n+1}(t) = 2t\, h_n(t) - 2n\, h_{n-1}(t) \qquad t \in \mathbf{R},$$

for Hermite functions and Cramer's inequality for Hermite polynomials:

$$|H_n(t)| \le K 2^{n/2} \sqrt{n!} e^{t^2/2}$$

where K is a numerical constant less than 1.09 in value.

In summarizing the extensions of these facts to $d > 1$ dimensions, we will find it convenient to adopt multiindex notation. A d-dimensional multiindex $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_d)$ is a d-tuple of nonnegative integers, playing the role of a multidimensional index. For any multiindex α and any $t \in \mathbf{R}^d$, we define

$$|\alpha| = \alpha_1 + \alpha_2 + \ldots + \alpha_d$$

$$\alpha! = \alpha_1! \alpha_2! \ldots \alpha_d!$$

$$t^\alpha = t_1^{\alpha_1} t_2^{\alpha_2} \ldots t_d^{\alpha_d}$$

$$D^\alpha = \partial_1^{\alpha_1}\partial_2^{\alpha_2}\dots\partial_d^{\alpha_d}$$

where ∂_i is differentiation with respect to the ith coordinate in $\mathbf{R}^d$. If p is an integer, we say $\alpha \geq p$ if $\alpha_i \geq p$ for $1 \leq i \leq d$.

The multidimensional Hermite polynomials and Hermite functions are then defined by

$$H_\alpha(t) = H_{\alpha_1}(t_1)\dots H_{\alpha_d}(t_d)$$

$$h_\alpha(t) = e^{-|t|^2} H_\alpha(t) = h_{\alpha_1}(t_1)\dots h_{\alpha_d}(t_d) \tag{5}$$

where $|t|^2 = t_1^2 + \dots + t_d^2$.

The Hermite expansion of a Gaussian in $\mathbf{R}^d$ is then simply

$$e^{-|t-s|^2} = \sum_{\alpha\geq 0} \frac{(t-s_0)^\alpha}{\alpha!}\, h_\alpha(s-s_0)\,. \tag{6}$$

Cramer's inequality implies

$$\frac{1}{\alpha!}|h_\alpha(t)| \leq K\, e^{-|t|^2/2}\, 2^{|\alpha|/2}\, \frac{1}{\sqrt{\alpha!}}, \tag{7}$$

where K is less than $(1.09)^d$.

We can now present the two key expansions on which our algorithms rely.

First, scale the Hermite expansion (6) by δ_j to get

$$\begin{aligned} e^{-|t-s_j|^2/\delta_j} &= \sum_{\beta\geq 0} \frac{1}{\beta!} h_\beta\left(\frac{s_j - t_C}{\sqrt{\delta_j}}\right)\left(\frac{t-t_C}{\sqrt{\delta_j}}\right)^\beta \\ &= \sum_{\beta\geq 0} \frac{1}{\beta!}\left(\frac{t-t_C}{\sqrt{\delta}}\right)^\beta \left(\frac{\delta}{\delta_j}\right)^{|\beta|/2} h_\beta\left(\frac{s_j-t_C}{\sqrt{\delta_j}}\right). \end{aligned} \tag{8}$$

This expresses a Gaussian centered at s_j as a Taylor series about an arbitrary target t_C, with a scaling factor δ independent of j. By Cramer's inequality, the error in truncating the Taylor series (8) after the first p^d terms is bounded by

$$|E_p| \leq K \sum_{\text{some} \quad \beta_i \geq p} \frac{1}{\sqrt{\beta!}} 2^{|\beta|/2}\left(\frac{|t-t_C|_\infty}{\sqrt{\delta_j}}\right)^{|\beta|}.$$

Here $|t|_\infty = \max_{1\leq i\leq d}|t_i|$. Suppose t lies in a box C with center t_C and sides of length $r\sqrt{2\delta} \leq r\sqrt{2\delta_j}$ parallel to the axes. Then Stirling's formula implies that

$$|E_p| \leq \frac{r_p^p}{1-r_p} \tag{9}$$

whenever $r_p^2 = er^2\delta/\delta_j(p+1) < 1$. This error bound decreases very rapidly as p increases. If $r = 1$ and $\delta_j = \delta$, then $|E_p| \le 10^{-4}$ when $p = 12$ and $|E_p| \le 10^{-6}$ when $p = 17$. Even if $r = 2$, accuracy 10^{-4} is attained with $p = 24$ and accuracy 10^{-6} with $p = 31$. In practice, this error bound overestimates the error by two orders of magnitude or more.

This error bound is more useful than the (erroneous) bound used in [5], because that bound required $r < 1$. The current bound allows us to use $r \ge 1$ if we use more terms in the Hermite series. This is practically useful because we can cluster the points into larger groups and use longer vectors, which makes the code run much faster on vector procesors.

Our second formula is the dual of the first, obtained by scaling with a target-dependent scale δ_i rather than δ_j and shifting to a source center s_B rather than a target center t_C. We find

$$(10) \qquad e^{-|t_i-s_j|^2/\delta_i} = \sum_{\beta\ge 0} \frac{1}{\beta!}\left(\frac{\delta}{\delta_i}\right)^{|\beta|/2} h_\beta\left(\frac{t_i - s_B}{\sqrt{\delta_i}}\right)\left(\frac{s_j - s_B}{\sqrt{\delta}}\right)^{\beta}.$$

This expresses a Gaussian centered at s_j as a scaled Hermite series centered at a nearby source box center s_B, with coefficients independent of i. If the source s_j lies inside a box of side length $r\sqrt{2\delta_i}$ centered at s_B, then the error in truncating the Hermite series after the first p^d terms is bounded by (9).

These two formulas provide the basis for the two fast algorithms developed in this paper.

3 Source-dependent Scales

In this section, we present our algorithm for evaluating

$$(11) \qquad Sf(t_i) = \sum_{j=1}^{N} f_j e^{-|t_i-s_j|^2/\delta_j}$$

at M targets t_i. We assume (by scaling the δ_j's if necessary) that all the targets lie in the unit box $B_0 = [0,1]^d$, for convenience of exposition. The sources s_j may lie anywhere in $\mathbf{R}^d$.

Our algorithm requires $O(M+N)$ work; the constant factor in $O(M+N)$ depends only on the precision ϵ and the minimum δ_m of the scales δ_j.

Choose cutoffs a and b and split the scales into three groups; $0 \leq \delta \leq a$, $a < \delta < b$, and $b \leq \delta$. We treat each group by an appropriate method, and choose a and b to minimize the work required.

First consider $0 \leq \delta \leq a$. Gaussians with small scales decay rapidly away from the source, so each of these sources can affect only targets which are quite nearby. Hence we treat these sources by direct evaluation, with a data structure constructed as in [11].

To do this, let R be the range of the longest-range Gaussian in this group; thus $e^{-R^2/a} \leq \epsilon$ and $R = \sqrt{-a \log \epsilon}$. Divide $[0,1]^d$ into n^d boxes of side length $\geq R$, so $n = \lfloor 1/R \rfloor$. Sort the targets t_i into boxes by location. Now run through the sources s_j. Put s_j into a box, B say, by location, and let $k_j = \lfloor \sqrt{-\delta_j/a} \rfloor$. Then s_j affects boxes with indices within k_j of B, to accuracy ϵ. Run through these boxes, incrementing $Sf(t_i)$ at each target t_i in each box. This concludes the treatment of sources with $0 \leq \delta_j \leq a$.

Now consider the large scales, with $b \leq \delta_j < \infty$. These Gaussians have very long ranges, so direct evaluation is very expensive. For the same reason, however, they are very smooth. If b is chosen large enough, all these sources can be incorporated into a single p^d-term truncated Taylor expansion about the box center $t_0 = (1/2, \ldots, 1/2)$ of $[0,1]^d$. with error less than $F\epsilon$ where $F = \sum |f_j|$. Indeed, (8) implies that

$$\sum_{\delta_j \geq b} f_j e^{-|t-s_j|^2/\delta_j} = \sum_{\beta \leq p} C_\beta \left(\frac{t - t_0}{\sqrt{\delta}} \right)^\beta + E_p$$

where

$$C_\beta = \frac{1}{\beta!} \sum_{\delta_j \geq b} \left(\frac{\delta}{\delta_j} \right)^{|\beta|/2} h_\beta \left(\frac{s_j - t_0}{\sqrt{\delta_j}} \right)$$

and

$$E_p \leq F \frac{r_p^p}{1 - r_p} \qquad\qquad r_p = \sqrt{\frac{e}{p+1}} r < 1.$$

Here $r = 1/\sqrt{2\delta} \leq 1/\sqrt{2b}$ and $F = \sum |f_j|$.

Thus the large variances are very easy to deal with; we simply form p^d coefficients C_β at a cost of $O(p^d N)$ work and evaluate the Taylor series at M targets, with a total cost of $O(p^d(M+N))$. The coefficient p^d depends only on the desired precision ϵ.

Finally, we must deal with the intermediate sources with $a < \delta_j < b$. We treat these by a combination of the previous two methods. The targets are sorted into boxes and each source contributes to a Taylor expansion about the center of each box within range. When each Taylor expansion is complete, it is evaluated at each target in the box.

We begin by sorting the targets into boxes C say, with centers t_C and sides of lengths $r\sqrt{2\delta}$ parallel to the axes. Here $\delta = \min_{a \le \delta_j \le b} \delta_j \ge a$. Now each source is converted into a Taylor expansion (8) about the center t_C of each target box C within range $R_j = \sqrt{-\delta_j \log \epsilon}$ of s_j. Thus we have, for $t \in C$,

$$\sum_{|s_j - t_C| \le R_j} f_j e^{-|t-s_j|^2/\delta_j} = \sum_{\beta \le p} C_\beta \left(\frac{t - t_C}{\sqrt{\delta}} \right)^\beta + E_p$$

where

$$C_\beta = \frac{1}{\beta!} \sum_{\delta_j \ge b} \left(\frac{\delta}{\delta_j} \right)^{|\beta|/2} h_\beta \left(\frac{s_j - t_C}{\sqrt{\delta_j}} \right)$$

and E_p satisfies the error bound

$$E_p \le F \frac{r_p^p}{1 - r_p}$$

with $r_p = \sqrt{\frac{e}{p+1}} r < 1$. We choose r and p to make $|E_p| \le F\epsilon$. After each source has contributed to all Taylor expansions in range, each Taylor expansion is evaluated at the targets t_i in its box.

Two slight refinements of this procedure reduce the overhead considerably in practice. First, it is inefficient to form a Taylor series and evaluate it for a box containing only a few targets. Thus a breakpoint M_T is introduced, boxes with more than M_T targets receive Taylor expansions, while direct evaluation is more efficient for boxes with fewer than M_T targets. Typically $M_T = 8$ or 10 is a reasonable value when single precision accuracy is desired. The right value of M_T can be determined by the program when it is run, in a reasonably straightforward way.

The second refinement is introduced in order to avoid a needlessly accurate computation. If a scaling factor $\delta = \min \delta_j$ is used to determine the box size, then many sources will have δ_j much larger than δ. For these sources, p^d terms is far too many to provide the desired accuracy, and p should be reduced. Hence we can compute how many terms are necessary for each source independently, and increment only those coefficients. Thus we compute q^d terms where q is chosen to satisfy the error estimate

$$\frac{r_{qj}^q}{1 - r_{qj}} \le \epsilon$$

with $r_{qj} = r\sqrt{e\delta/\delta_j(q+1)}$. The small overhead required to compute q for each source is compensated for by the reduction in effort required to evaluate Hermite coefficients. In practice, this refinement often produces a factor of two speedup in the calculation.

A third refinement is possible; one can subdivide the center interval (a, b) further, in order to avoid treating longer-range smoother Gaussians on wastefully small scales. A geometric subdivision as in [12] is possible, or some other problem-dependent subdivision may be more appropriate.

The algorithm is completed by the choice of the cutoffs a and b. If the scales δ_j are distributed in a fairly uniform way over the interval $[\delta_m, \delta_M]$, then one can take $a = \delta_m + O(1/N)$ and b fixed, say $b = 1/10$. The choice of a is made because we can afford to have only a fixed number of sources below a. The choice of $b = 1/10$ is about the smallest for which we can make the Hermite series accurate with a practical number of terms, like 20^d or 30^d. The choice of a is more important when there are a few outlying scales δ_j, with all the rest lying considerably higher. Then we should choose a to be the essential minimum of the δ_j's, in other words the smallest lower bound for all but a fixed number of the scales as $N \to \infty$. Thus only the outliers will be less than a. This can substantially improve the performance of the medium scales, because we can take larger boxes, and the cost of evaluating a few sources directly is then worth it.

4 Target-dependent Scales

In this section, we describe how to evaluate

$$Tf(t_i) = \sum_{j=1}^{N} f_j e^{-|t_i - s_j|^2/\delta_i}$$

at M points $t_i \in \mathbf{R}^d$, under the assumption (for convenience) that all the sources s_j lie in $[0,1]^d$. The algorithm is structurally dual to the algorithm of §3, so we concentrate on the differences.

We divide the δ_i's into three sets in the same way as before, and treat each set of *targets* as a group.

For targets with $0 \le \delta_i \le a$, we sort the sources into boxes rather than the targets, and apply direct evaluation.

For the middle interval $a < \delta_i < b$), we use the Hermite expansion (10) with $\delta = \min_{a \leq \delta_i \leq b} \delta_i \geq a$. We impose a box structure on the sources and collect all the sources in each box B (with center s_B and sides of length $r\sqrt{2\delta}$ parallel to the axes) into a scaled Hermite expansion about the box center, valid everywhere in $[0,1]^d$:

$$T_B f(t_i) = \sum_{s_j \in B} f_j e^{-|t_i - s_j|^2/\delta_i} = \sum_{\beta \leq p} B_\beta \left(\frac{\delta}{\delta_i}\right)^{|\beta|/2} h_\beta \left(\frac{t_i - s_B}{\sqrt{\delta_i}}\right)^\beta + E_p$$

where

$$B_\beta = \frac{1}{\beta!} \sum_{s_j \in B} f_j \left(\frac{s_j - s_B}{\sqrt{\delta}}\right)^\beta$$

and we make $E_p \leq F\epsilon$ by choice of r and p. Then we evaluate a fixed number of Hermite expansions at each target t_i:

$$Tf(t_i) = \sum_{|s_B - t_i| \leq R_i} T_B f(t_i),$$

with $R_i = \sqrt{-\delta_i \log \epsilon}$. Some overhead can be avoided if we form Hermite series only for boxes with more than say N_S sources; boxes with fewer than N_S sources are handled by direct evaluation. Also, targets with δ_i much larger than δ need only evaluate a smaller Hermite series, since the error decays faster for them.

Finally, the sources with $\delta_i \geq b$ are handled by forming all of them into a single Hermite expansion about the center of $[0,1]^d$ and evaluating it at each target.

The parameters a, b and N_S are chosen in a way very like the source-dependent algorithm.

5 Numerical Results

In this section, we present some numerical results showing the speedups achievable by our algorithms. We tested the source-dependent algorithm on sets of $N = M$ sources and targets in $d = 2$ dimensions; the target-dependent algorithm will exhibit highly similar statistics. We used random uniformly distributed strengths on $[0,1]$, with source and target points uniformly distributed either on $[0,1]^d$ or on a circle in $[0,1]^d$. The scales were uniformly distributed on $[\delta_m, \delta_M]$ for various choices of δ_m and δ_M.

Tables 1-4 present results for the source-dependent scale fast Gauss transform for four examples. The time taken by the fast algorithm is recorded in the column headed T_f, the time required to evaluate the Gaussian sum directly under T_d, and the error produced by the fast algorithm — relative to $F = \sum |f_j|$ — under the heading E_p/F. All computations were carried out in double precision arithmetic with the error tolerance ϵ set to 10^{-4}. (This was enough to achieve an actual error well below 10^{-6} in every case.) This required $p = 10$ with $r = 1$ for sources with $a \leq \delta_j \leq b$ and $p = 24$ for $\delta_j > b$. In Table 1, we used source and target points uniformly distributed on $[0,1]^d$ and scales δ_j uniformly distributed between $\delta_m = 0.1$ and $\delta_M = 1.0$. We took $a = 0$, $b = 0.125$, and $M_T = 8$. The results show a beautifully linear growth of computation time with N for the fast algorithm, against a quadratic time for direct evaluation. A speedup of 2275 is obtained with $N = 102400$, indicating a projected breakeven point of 45.

Case	$N = M$	T_f	T_d	E_p/F
1	100	0.05	0.10	0.222e-06
2	200	0.09	0.40	0.163e-06
3	400	0.18	1.60	0.551e-07
4	800	0.36	6.80	0.110e-07
5	1600	0.76	25.60	0.449e-07
6	3200	1.45	99.20	0.128e-07
7	6400	2.88	406.40	0.186e-07
8	12800	5.75	1638.40	0.957e-08
9	25600	11.45	6553.60	0.554e-08
10	51200	22.96	26060.80	0.114e-07
11	102400	45.94	104499.20	0.836e-08

Table 1: Times and errors for $0.1 \leq \delta_j \leq 1.0$, with targets and sources distributed uniformly in the unit box.

In Table 2, we used points randomly chosen on a circle with radius 0.29 and center (0.6,0.7), with the same parameters as in Table 1. The non-uniformity of the sources and targets did not slow down the algorithm.

Table 3 presents times and errors for the same calculation as in Example 2, but with the minimum scale reduced to $\delta_m = 0.01$. This slows down the algorithm only slightly; the speedup goes down to 1776 with 102,400 points.

Case	$N = M$	T_f	T_d	E_p/F
1	100	0.06	0.10	0.115e-06
2	200	0.10	0.40	0.553e-06
3	400	0.16	1.60	0.439e-06
4	800	0.37	6.40	0.403e-06
5	1600	0.71	24.00	0.352e-06
6	3200	1.43	102.40	0.335e-06
7	6400	2.83	409.60	0.181e-06
8	12800	5.69	1632.00	0.255e-06
9	25600	11.38	6528.00	0.226e-06
10	51200	22.62	26214.40	0.155e-06
11	102400	45.43	104192.00	0.254e-06

Table 2: Times and errors for $0.1 \leq \delta_j \leq 1.0$, with targets and sources distributed uniformly on a circle.

Case	$N = M$	T_f	T_d	E_p/F
1	100	0.06	0.10	0.145e-06
2	200	0.12	0.40	0.564e-06
3	400	0.24	1.60	0.420e-06
4	800	0.45	6.40	0.252e-06
5	1600	0.90	25.60	0.311e-06
6	3200	1.81	102.40	0.275e-06
7	6400	3.60	400.00	0.170e-06
8	12800	7.28	1657.60	0.238e-06
9	25600	14.54	6502.40	0.201e-06
10	51200	29.03	26291.20	0.144e-06
11	102400	58.86	104499.20	0.236e-06

Table 3: Times and errors for $0.01 \leq \delta_j \leq 1.0$, with targets and sources distributed uniformly on a circle.

Table 4 is similar to Table 3, but with δ_j's lying between 0.002 and 0.2. Even with these fairly small scales, we still achieve a respectable speedup of 500 with $N = 102,400$.

Case	$N = M$	T_f	T_d	E_p/F
1	100	0.14	0.10	0.112e-07
2	200	0.36	0.30	0.180e-06
3	400	0.73	1.40	0.123e-06
4	800	1.55	6.00	0.110e-06
5	1600	3.18	25.60	0.623e-07
6	3200	6.59	102.40	0.683e-07
7	6400	13.22	409.60	0.460e-07
8	12800	26.88	1644.80	0.936e-07
9	25600	54.15	6720.00	0.634e-07
10	51200	107.85	26470.40	0.266e-07
11	102400	216.64	108083.20	0.720e-07

Table 4: Times and errors for $0.002 \leq \delta_j \leq 0.2$, with targets and sources distributed uniformly on a circle.

6 Conclusions

We have presented fast algorithms which evaluate d-dimensional Gaussian convolution sums (1) and (2) with scales dependent either on the source or the target. Such sums are frequently evaluated in statistical computations[3, 10] and also occur in numerical solution of the heat equation [9, 8] and other applications[7]. Our algorithms evaluate these sums in $O(M + N)$ work, with a constant factor depending on the precision required and the minimum scale which occurs.

Numerical results indicate that our method can speed up large calculations by several orders of magnitude; in one example, the computation time was reduced to *one minute* instead of the *1.2 days* required by direct evaluation. Thus this algorithm can be extremely useful in reducing the time required to carry out large-scale computations. We are currently applying a variant of this algorithm to problems in crystal growth, where it is used to evaluate free-space heat potentials.

References

[1] H. Dym and H. P. McKean, *Fourier Series and Integrals*, Academic Press, San Diego, 1972.

[2] A. Erdelyi, et. al. *Higher Transcendental Functions*, vol. II, McGraw-Hill, New York, 1953.

[3] S. Geman and C. Hwang, *Nonparametric Maximum Likelihood Estimation by the Method of Sieves*, Ann. Statist., 10 (1982), pp. 401-414.

[4] I. S. Gradshteyn and I. M. Ryzhik, *Tables of Integrals, Series, and Products*, Academic Press, New York, 1980.

[5] L. Greengard and J. Strain, *The Fast Gauss Transform*, SIAM J. Sci. Stat. Comput., in press.

[6] E. Hille, *A Class of Reciprocal Functions*, Ann. Math. 27(1926), pp. 427-464.

[7] N. Lerner, *Wick-Wigner Functions and Tomographic Methods*, preprint,1989.

[8] A. McIntyre, *Boundary Integral Solutions of the Heat Equation*, Math. Comp., 46 (1986), pp. 71-79.

[9] P. J. Noon, *The Single Layer Heat Potential and Galerkin Boundary Element Methods for the Heat Equation*, Ph.D. Thesis, University of Maryland, 1988 .

[10] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London, 1986.

[11] J. Strain, *Fast Potential Theory II: Layer Potentials and Discrete Sums*, UC Berkeley Center for Pure and Applied Mathematics Report PAM-567, April 1990, and submitted to Jour. Comput. Phys.

[12] J. Strain, *A Fast Laplace Transform Based on Laguerre Functions*, submitted to Math. Comp.