Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2021-0031

Vol. **31**, No. 1, pp. 1-26
January 2022

# Approximating the Gaussian as a Sum of Exponentials and its Applications to the Fast Gauss Transform

Shidong Jiang[1,3,*] and Leslie Greengard[2,3]

[1] *Department of Mathematical Sciences, New Jersey Institute of Technology, Newark, NJ 07102, USA.*
[2] *Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA.*
[3] *Center for Computational Mathematics, Flatiron Institute, Simons Foundation, New York, NY 10010, USA.*

Received 7 February 2021; Accepted (in revised version) 21 October 2021

**Abstract.** We develop efficient and accurate sum-of-exponential (SOE) approximations for the Gaussian using rational approximation of the exponential function on the negative real axis. Six digit accuracy can be obtained with eight terms and ten digit accuracy can be obtained with twelve terms. This representation is of potential interest in approximation theory but we focus here on its use in accelerating the fast Gauss transform (FGT) in one and two dimensions. The one-dimensional scheme is particularly straightforward and easy to implement, requiring only twenty-four lines of MATLAB code. The two-dimensional version requires some care with data structures, but is significantly more efficient than existing FGTs. Following a detailed presentation of the theoretical foundations, we demonstrate the performance of the fast transforms with several numerical experiments.

**AMS subject classifications**: 31A10, 65F30, 65E05, 65Y20

**Key words**: Fast Gauss transform, sum-of-exponential approximation, best rational approximation, model reduction.

## 1 Introduction

In this paper, we consider the approximation of the Gaussian kernel

$$G(x;\delta)=e^{-\frac{x^2}{4\delta}}, \tag{1.1}$$

*Corresponding author. *Email addresses:* `shidong.jiang@njit.edu` (S. Jiang), `greengard@courant.nyu.edu` (L. Greengard) using a sum-of-exponential (SOE) representation:

$$G(x;\delta) \approx S_{K_e}(x;\delta) := \sum_{k=1}^{K_e} w_k e^{-t_k \frac{|x|}{\sqrt{\delta}}}, \tag{1.2}$$

where $w_k$ and $t_k$ are complex weights and nodes and $K_e$ is the number of such terms. We show numerically that the SOE approximation converges geometrically, with maximum error of the order $\mathcal{O}\left(c^{-K_e}\right)$. The optimal value for $c$ is difficult to determine, but using ideas from rational approximation (see, for example, [7,11,36]), we show that only twelve terms are needed for ten-digit accuracy. Moreover, when $K_e$ is even, the weights and nodes are constructed in complex conjugate pairs so that (with a suitable ordering) we may write

$$S_{K_e}(x;\delta) = \Re\mathrm{e}\left(\sum_{k=1}^{K_e/2} w_k e^{-t_k \frac{|x|}{\sqrt{\delta}}}\right). \tag{1.3}$$

Thus, in one dimension, only six terms are needed for ten-digit accuracy.

**Remark 1.1.** Unfortunately, the factor of two reduction in (1.3) can only be used in one dimension. When approximating the two dimensional Gaussian, we will make use of the separable approximation

$$G(x,y;\delta) = e^{-\frac{x^2+y^2}{4\delta}} \approx \Re\mathrm{e}\left(\sum_{k=1}^{K_e/2} w_k e^{-t_k \frac{|x|}{\sqrt{\delta}}} \sum_{l=1}^{K_e} w_l e^{-t_l \frac{|y|}{\sqrt{\delta}}}\right). \tag{1.4}$$

We show here that the SOE approximation of the Gaussian can be used to develop a new version of the fast Gauss transform (FGT), which computes sums of the form

$$u_i = \sum_{j=1}^{N} G(x_i - y_j;\delta) q_j, \quad i = 1, \cdots M, \tag{1.5}$$

in $\mathcal{O}(N+M)$ time. The main advantage of exponential functions in this context follows from the fact that they are eigenfunctions of the translation operator, which leads to a simple "sweeping" algorithm in one dimension, whose performance is entirely independent of the variance $\delta$. In higher dimensions, the SOE approximation can be used to accelerate existing FGTs [13, 14, 29, 38]. The SOE-based scheme shares some feature with the "plane wave" versions of the FGT [14, 29, 38], with an important difference. The earlier plane-wave schemes use the Fourier transform to develop an approximation of the Gaussian in terms of oscillatory exponentials with a restricted range of validity. The SOE approximation uses the Laplace transform, involves many fewer terms, and is uniformly accurate in the ambient space. It outperforms existing FGTs in the literature. A slight difficulty arises from the fact that the argument in the SOE approximation involves the modulus of the argument, so that we have

$$G(x;\delta) \approx \begin{cases} \sum_{k=1}^{K_e} w_k e^{-t_k x \sqrt{\delta}} & \text{for } x > 0, \\ \sum_{k=1}^{K_e} w_k e^{t_k x \sqrt{\delta}} & \text{for } x < 0. \end{cases} \tag{1.6}$$

In one dimension, this permits the construction of a simple FGT by carrying out two sweeps (one in the positive direction and one in the negative direction). In higher dimensions, a somewhat more complicated algorithm is required.

The paper is organized as follows. In Section 2, we discuss several methods for obtaining SOE approximations of the Gaussian kernel. Numerical experiments corresponding to these methods are presented in Section 3. The one-dimensional sweeping scheme for the FGT is described in Section 4. In Section 5, we present a method which combines SOE approximations with Hermite expansions (as in the original FGT [13]) for the two-dimensional case. Numerical results are presented in Section 6 and future directions for research are discussed in Section 7.

## 2 SOE approximation of the Gaussian kernel

A straightforward calculation shows that the Laplace transform of the one-dimensional heat kernel $\frac{1}{\sqrt{4\pi t}} e^{-\frac{|x|^2}{4t}}$ is $\frac{1}{2\sqrt{s}} e^{-\sqrt{s}|x|}$ (see, for example, [20]). Taking the inverse Laplace transform, we have

$$\frac{1}{\sqrt{4\pi t}} e^{-\frac{|x|^2}{4t}} = \frac{1}{2\pi i} \int_\Gamma e^{st} \frac{1}{2\sqrt{s}} e^{-\sqrt{s}|x|} ds, \tag{2.1}$$

where $\Gamma$ is a suitably chosen contour. Multiplying both sides by $\sqrt{4\pi t}$, replacing $t$ by $\delta$, and introducing the change of variables $z = s\delta$, we obtain

$$G(x;\delta) = e^{-\frac{x^2}{4\delta}} = \frac{1}{2\pi i} \int_\Gamma e^z \sqrt{\frac{\pi}{z}} e^{-\frac{\sqrt{z}|x|}{\sqrt{\delta}}} dz. \tag{2.2}$$

It remains to define a suitable family of contours $\Gamma$. For this, we assume that the branch cut of the square root function is chosen to be the negative real axis, $\mathbb{R}^-$, with the branch taken to be the principal branch: $\arg(z) \in (-\pi, \pi]$. Note that on $\mathbb{C} \backslash \mathbb{R}^-$, the square root function $\sqrt{z}$ has positive real part and thus

$$\left| e^{-\frac{\sqrt{z}|x|}{\sqrt{\delta}}} \right| \leq 1 \tag{2.3}$$

for all $x \in \mathbb{R}$ and $\delta > 0$. Thus, by Cauchy's theorem, we may use for $\Gamma$ any contour in the complex plane that starts from $-\infty$ in the third quadrant, circles the origin, and loops back to $-\infty$ in the second quadrant (Fig. 1). Clearly, discretization of the integral in (2.2) leads to an SOE approximation of the form (1.2) for the Gaussian kernel. It remains to choose an optimal contour and quadrature rule.

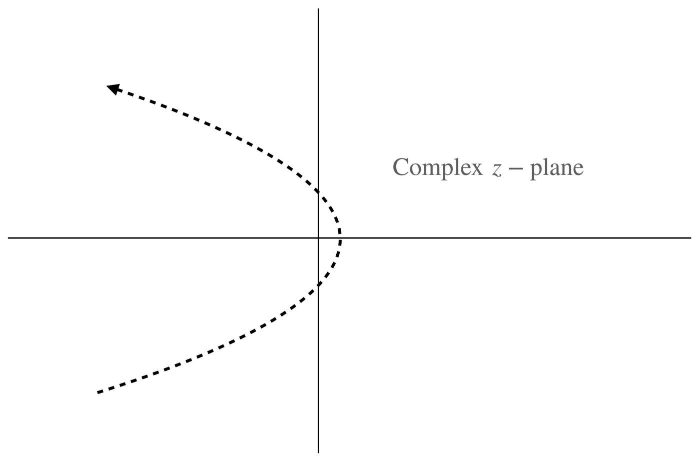

Figure 1: Allowable contours for the inverse Laplace transform (2.2). Discretization of the contour integral yields the sum-of-exponential approximation. For efficiency, this requires joint optimization of the curve $\Gamma$ and the quadrature rule.

## 2.1 Numerical evaluation of the contour integral

For notational simplicity, we will consider integrals of the form

$$I=\frac{1}{2\pi i}\int_\Gamma e^z f(z)dz. \tag{2.4}$$

The case of the Gaussian kernel then corresponds to

$$f(z)=\sqrt{\frac{\pi}{z}}e^{-\frac{\sqrt{z}|x|}{\sqrt{\delta}}}. \tag{2.5}$$

The numerical evaluation of the contour integral (2.4) is a problem which has received significant attention. It is convenient to parametrize the integral in terms of a real parameter $\theta$. That is $z=z(\theta)$ and

$$I=\frac{1}{2\pi i}\int_{-\infty}^{\infty} e^{z(\theta)}f(z(\theta))z'(\theta)d\theta. \tag{2.6}$$

The integral in (2.6) is then truncated and discretized via either the trapezoidal or midpoint rule, resulting in an approximation of the form

$$I_n=\frac{h}{2\pi i}\sum_{k=1}^{n} e^{z(\theta_k)}f(z(\theta_k))z'(\theta_k), \tag{2.7}$$

where $\theta_k$ are $n$ equispaced points with spacing $h$. Substituting (2.5) into (2.7), we obtain an SOE approximation for the Gaussian kernel

$$S_{K_e}(x;\delta)=\sum_{k=1}^{K_e} w_k e^{-t_k\frac{|x|}{\sqrt{\delta}}} \tag{2.8}$$

with

$$w_k = \frac{h}{2\sqrt{\pi}i}\frac{z'(\theta_k)}{\sqrt{z(\theta_k)}}e^{z(\theta_k)}, \quad t_k = \sqrt{z(\theta_k)}. \tag{2.9}$$

Obviously, the efficiency and accuracy of the approximation $I_n$ depends on the contour and its parametrization. The classical theory summarized in [30] shows that $I_n$ converges to $I$ at least subgeometrically, with an error decaying at the rate $\mathcal{O}(e^{-c\sqrt{n}})$. Recent developments have shown that the convergence rate can actually be geometric $\mathcal{O}(e^{-cn})$ (see, for example, [35] for an excellent review and discussion).

There are several classes of contours that have been considered in detail, beginning with Talbot's proposal of cotangent contours [32]. Analysis of these "Talbot contours" can be found in [23]. Simpler parabolic contours were subsequently proposed in [26] and hyperbolic contours in [24,25]. The hyperbolic contours are applicable to a rather broad class of functions $f$ in (2.4) in the sense that $f$ can be allowed to have singularities in a sectorial region around $\mathbb{R}^-$. Optimal choices for all three contours were first discussed systematically in [36], with detailed analysis of parabolic and hyperbolic contours in [39, 40], and modified Talbot contours in [8].

## 2.2 Best rational approximations to $e^z$ on $\mathbb{R}^-$

When considering the discrete approximation $I_n$ to $I$, it will be helpful to write it in the general form

$$I_n = -\sum_{k=1}^{n} c_k f(z_k). \tag{2.10}$$

As above, the corresponding SOE approximation for the Gaussian kernel becomes

$$S_{K_e}(x;\delta) = \sum_{k=1}^{K_e} w_k e^{-t_k\frac{|x|}{\sqrt{\delta}}}, \quad w_k = -c_k\sqrt{\frac{\pi}{z_k}}, \quad t_k = \sqrt{z_k}. \tag{2.11}$$

The distinction between (2.10) and (2.7) is that the points $z_k$ can be chosen in suitable regions of the complex plane without necessarily lying along a given contour. It is natural to ask whether this additional flexibility can yield more efficient approximations (assuming the nonlinear optimization problem it leads to is tractable). It turns out there is a direct connection between the approximation of $I$ by $I_n$ and the best rational approximation of the exponential function $e^x$ on $\mathbb{R}^-$. More precisely, it is shown in [36], using the residue theorem and Cauchy's theorem, that

$$I - I_n = \frac{1}{2\pi i}\int_{\Gamma'}(e^z - r(z))f(z)dz, \tag{2.12}$$

where

$$r(z) = \sum_{k=1}^{n}\frac{c_k}{z - z_k} \tag{2.13}$$

and $\Gamma'$ is a contour lying between $\mathbb{R}^-$ and the points $z_k$. The formula (2.12) indicates that $I-I_n$ will be small if $r(z)$ is a good approximation to $e^z$ on $\mathbb{R}^-$. This is particularly true when all singularities of $f$ lie on $\mathbb{R}^-$ since $\Gamma'$ can then be chosen arbitrarily close to $\mathbb{R}^-$, which is the case for the Gaussian kernel. In [11], it is proven that

$$\max_{z\in R^-}\left|e^z - r^*_{n,n}(z)\right| = \mathcal{O}\left(9.28903\cdots^{-n}\right), \tag{2.14}$$

where $r^*_{n,n}$ is the best rational approximation of type $(n,n)$. In [7], a method based on the classical Remez algorithm was devised to calculate the best rational approximation $r^*_{n,n}$ and the approximation error to very high precision. In [36], an algorithm based on the Carathéodory–Fejér (CF) method was developed to compute a nearly best rational approximation of type $(n-1,n)$, which includes (2.13), for $n$ up to 14 (see the MATLAB code cf.m in [36]).

### 2.3 Selecting SOE approximation via the balanced truncation method

The rational function $r(z)$ in (2.13) is, for obvious reasons, also called a "sum of poles". When all poles lie in the left half of the complex plane, various model order reduction or balanced truncation methods can be used to try to reduce the number of poles while achieving a specified $L^\infty$ error. As noted in [12], since the Laplace transform of an exponential function is a pole function, such a reduction leads to a more efficient SOE approximation as well. We do not seek to review the literature here, but note simply that balanced truncation methods have been investigated in the context of infinite Hankel matrices [1–3, 27], time-invariant linear systems [10], and circuit models [22]. They are related to the CF method for rational approximation [15–18, 33, 34], and Prony's method for SOE approximations [5, 6].

## 3 Optimized SOE approximations of the Gaussian kernel

In this section, we develop efficient SOE approximations of the Gaussian kernel using the methods discussed in Section 2. Our objective function is the maximum error

$$E_n = \max_{x\in\mathbb{R}}\left|G(x;\delta) - S_n(x;\delta)\right| \tag{3.1}$$

estimated by setting $\delta = 1$ and sampling $x$ at 0 and at 100,000 equally spaced points on a logarithmic scale covering the interval $[10^{-5}, 10^2]$. (Note that $E_n$ is independent of $\delta$ for any $\delta > 0$ by a suitable rescaling.)

### 3.1 Parabolic, hyperbolic, and modified Talbot contours

Without entering into detail, three types of contours are considered in [36], which can be discretized via the midpoint rule. Fig. 2 shows $E_n$ as a function of $n$ for these choices. The

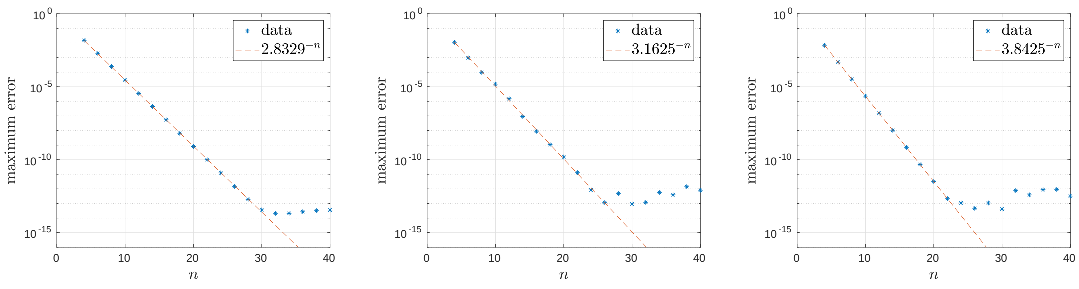

Figure 2: Maximum error of the SOE approximation of the Gaussian kernel as a function of $n$ using various contours (Left: optimal parabolic contour, Middle: optimal hyperbolic contour, Right: optimal modified Talbot contour). Dashed lines show the estimated convergence rates obtained by least squares fitting of the data points. They are close to the optimal values listed in [36].

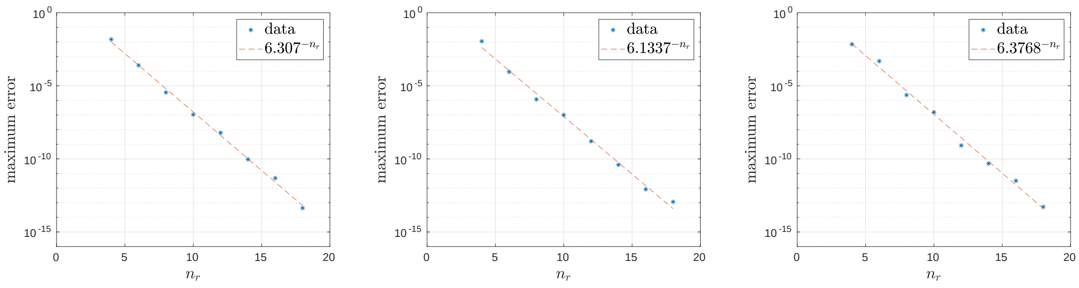

Figure 3: Reducing the number of exponentials in the SOE approximation of the Gaussian kernel using balanced truncation. (Left: optimal parabolic contour, Middle: optimal hyperbolic contour, Right: optimal modified Talbot contour).

convergence rates are very close to the theoretical values found in [36] in all three cases. In double precision arithmetic, the best that can be achieved is about 13-digit accuracy due to modest "catastrophic cancellation" errors.

Given an SOE approximation obtained from discretization of one of these contour integrals, we may reduce the number of exponentials required by means of a simple balanced truncation method outlined in [41]. The prescribed precision for the balanced truncation method is set to $E_n/3$ to ensure that the reduced SOE approximation has about the same accuracy as the original one. The results are summarized in Fig. 3. For all three contours, the reduced number $n_r$ of exponentials in the SOE approximation saturates at about 18 (for the twelve digit accuracy obtained from the optimal contours by quadrature). Note that the convergence rate after this model reduction/compression phase is improved to about $\mathcal{O}(6^{-n})$.

## 3.2 Best rational approximation

One of the reasons that we don't discuss the contour integral approach in more detail, is that equally good or better SOE approximations are obtained by best rational approx-

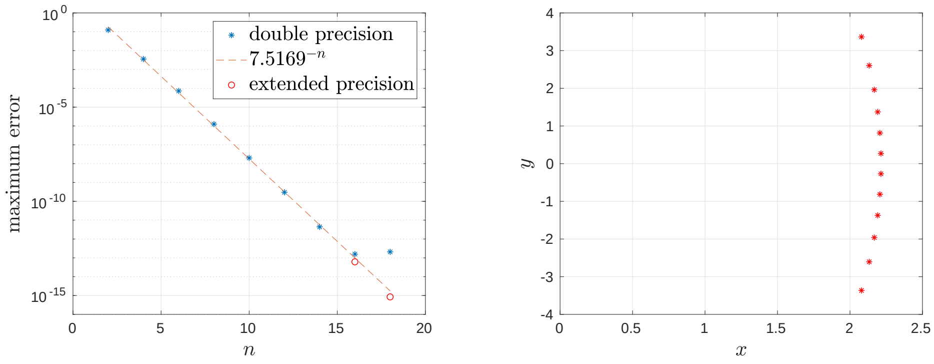

Figure 4: The SOE approximation of the Gaussian kernel obtained by best rational approximation to the exponential function on the negative real axis. Left: maximum error $E_n$ as a function of $n$. Dashed line is the least squares fit to the first seven data points. The red circles for $n=16,18$ are calculated in extended precision. Right: locations of exponential nodes $t_k$ for $n=12$.

imations to the exponential function on the negative real axis. For $n \le 14$, we use the MATLAB code cf.m in [36] to compute pole locations and weights. For $n=16$ and 18, we have implemented the same CF algorithm in FORTRAN and run the code in quadruple precision to obtain additional, more accurate pole locations and weights. Unfortunately, the balanced truncation method fails to yield further reduction in the number of terms, so that $n_r = n$ is the needed number of exponentials. Fig. 4 summarizes our numerical results. Note that the convergence rate is about $\mathcal{O}(7.5^{-n})$, better than those obtained by contour integration, even after the application of balanced truncation. This is also consistent with the discussion in [36]. Note also that catastrophic cancellation errors cause the approximation to stop converging in double precision arithmetic at about 13-digit accuracy (as was the case for the contour integration approach). When $n = n_r$ is even, the exponentials come in complex conjugate pairs. Since the strengths and output are real, we need only keep one half of the exponentials indicated in Fig. 4. Thus, 3 exponentials achieve four-digit accuracy, and 6 exponentials achieve about ten-digit accuracy, uniformly in $x$!

## 3.3 Reducing catastrophic cancellation errors

All of the SOE approximations discussed above have errors that saturate at about thirteen digits. This catastrophic cancellation can be attributed to the well-known ill-conditioning of the inverse Laplace transform (see, for example, [25,40]). As a result, various attempts have been made to stabilize the inversion process. Strategies for hyperbolic, parabolic and modified Talbot contours can be found in [25], [39] and [8], respectively.

We have tested these techniques and all of them do, indeed, yield more stable SOE approximations for the Gaussian kernel. In our experiments, stabilization of the parabolic and Talbot contours yields 14-digit accuracy and stabilization of the hyperbolic contours yields 15-digit accuracy. Subsequent application of balanced truncation reduces the num-

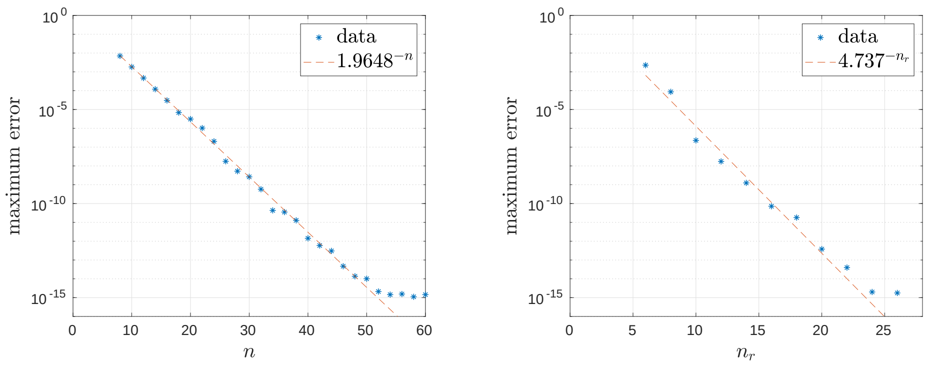

Figure 5: SOE approximation of the Gaussian kernel with smallest roundoff error. Left: error of the SOE approximation obtained using the hyperbolic contour in [25]. Right: error of the SOE approximation after balanced truncation. Dashed lines show the estimated convergence rate via least squares fitting of the data points.

ber of exponentials and improves the convergence rate greatly. In Fig. 5, we present the results from one of our experiments, using the hyperbolic contour in [25]:

$$z = \lambda(1 - \sin(0.8 + iu)),$$

with $a = \cosh^{-1}(2/((1-\theta)\sin(0.8)))$, $\lambda = 0.6\pi(1-\theta)n/a$, and step size $h = 2a/n$. The parameter $\theta$ is set to $1/4$ for $n \leq 16$ and $12/n$ otherwise. We observe that when the number of terms in the SOE approximation $n_r = 24$, the error is about $2.0 \times 10^{-15}$, the best we have achieved with any method.

**Remark 3.1.** In some sense, the ill-conditioning of the inverse Laplace transform can be viewed as an advantage in the present context. More precisely, there are a wide range of parameters and contours that will lead to quite different SOE approximations with about the same accuracy and the same number of exponentials. In the remainder of this paper, we restrict our attention to the SOE approximation obtain using the Carathéodory–Fejér (CF) method and code cf.m in [36] for for $n$ up to 12, yielding 10-digit accuracy.

## 4 A new sweeping scheme for the 1D FGT

We turn now to a fast algorithm for the Gauss transform in one dimension using SOE approximation, originally presented in [19]. While the approximation aspects are different, the implementation is very similar to that in [9], where SOE approximations are developed for the kernel $\frac{1}{x}$ on $[\delta, R]$ (away from the singularity). Since the SOE approximation of the Gaussian kernel is valid on the whole real line, the algorithm presented below is a bit simpler than that in [9].

Our target problem is the calculation of sums of the form (1.5), under the assumption that the target and source locations have been sorted in ascending order. Substituting the SOE approximation (1.3), assuming $K_e$ is even, into (1.5) and exchanging the order of summation, we obtain

$$u_i \approx \Re\mathrm{e}\left(\sum_{k=1}^{K_e/2} w_k h_{k,i}\right), \quad i=1,\cdots,M, \tag{4.1}$$

where

$$h_{k,i}=\sum_{j=1}^{N} q_j e^{-t_k\frac{|x_i-y_j|}{\sqrt{\delta}}}. \tag{4.2}$$

In order to eliminate the absolute value symbol, we further split $h_{ki}$ into two terms

$$h_{k,i}=h^+_{k,i}+h^-_{k,i}, \tag{4.3}$$

where

$$h^+_{k,i}=\sum_{y_j\le x_i} q_j e^{-t_k\frac{x_i-y_j}{\sqrt{\delta}}}, \quad h^-_{k,i}=\sum_{y_j> x_i} q_j e^{-t_k\frac{y_j-x_i}{\sqrt{\delta}}}. \tag{4.4}$$

It is clear that $h^+_{k,i}$ satisfies the forward or *left-to-right* recurrence

$$h^+_{k,i+1}=e^{-t_k\frac{x_{i+1}-x_i}{\sqrt{\delta}}}h_{k,i}+\sum_{x_i<y_j\le x_{i+1}} q_j e^{-t_k\frac{x_{i+1}-y_j}{\sqrt{\delta}}}, \tag{4.5}$$

and that $h^-_{k,i}$ satisfies the backward or *right-to-left* recurrence

$$h^-_{k,i-1}=e^{-t_k\frac{x_i-x_{i-1}}{\sqrt{\delta}}}h_{k,i}+\sum_{x_i\ge y_j> x_{i-1}} q_j e^{-t_k\frac{y_j-x_{i-1}}{\sqrt{\delta}}}. \tag{4.6}$$

Hence, $h^\pm_{k,i}$ can be computed in $\mathcal{O}(N+M)$ time for each $k$ and all $i=1,\cdots,M$.

The implementation of this scheme is so simple that we present a 24-line MATLAB code for the case when the target points are identical to the source points in Fig. 6. A few remarks are in order. First, we call cf.m from [36] directly in the code. These weights and nodes can be precomputed and stored. Second, for faster memory access in MATLAB, we have written the right-to-left pass as a forward loop. The code runs at about one million points per second on one core of a laptop with an Intel(R) 2.10GHz i7-4600U CPU. Third, when the target and source points are distinct, the left-to-right pass is replaced by the code fragment in Fig. 7. The changes to the right-to-left pass are similar and omitted.

This sweeping algorithm is similar to that in [14, 29] except that the complex exponentials in the earlier papers are purely oscillatory and approximately 16 exponentials are required for 10-digit accuracy instead of 6. An even faster FGT could be developed by imposing a box structure on the sources and targets and making use of Hermite expansions as well as SOE expansions. We are more interested, however, in the higher dimensional consequences of the SOE approximation.

```
function u=fgt1d(x,q,delta)       % x=locations, q=strengths
[zs,cs] = cf(12);                 % call cf.m in Trefethen et al. (2006)
zs = zs(1:2:12); cs = cs(1:2:12);% take only half of the exponentials
ws = -2*sqrt(pi)*cs./sqrt(zs);    % compute SOE approximation weights
ts = sqrt(zs/delta);              % compute SOE approximation nodes
[xs,I] = sort(x);                 % sort the points in ascending order
beta = q(I);                      % align the strength vector with points
e1 = exp(-ts*diff(xs));           % compute all needed complex exponentials
n2 = length(ts); nx = length(x);
hp = zeros(n2,nx);
e2 = ones(n2,1);
hp(:,1) = beta(1)*e2;
for i = 2:nx                      % left-to-right (forward) pass
  hp(:,i) = beta(i)*e2+e1(:,i-1).*hp(:,i-1);
end
hm = zeros(n2,nx);
e1 = fliplr(e1);
beta = fliplr(beta);
for i = 2:nx                      % right-to-left (backward) pass
  hm(:,i) = e1(:,i-1).*(beta(i-1)+hm(:,i-1));
end
u = real(ws.'*(hp+fliplr(hm)));  % sum over all exponential modes
Iinv(I) = 1:nx;
u = u(Iinv);                      % reorder output in original target order
```

Figure 6: MATLAB code for the fast Gauss transform in one dimension when the target points are identical to the source points.

## 5 An SOE-Hermite-based FGT in two dimensions

Before investigating how the SOE approximation can be used to accelerate the FGT in two dimensions, we briefly review the original FGT [13] which makes use of Hermite expansions as outgoing representations and Taylor expansions as incoming representations (analogous to multipole and local expansions in the fast multipole method).

Since there are many variants of the FGT described in the literature, we will simply sketch the main ideas here, beginning with the Hermite expansion induced by a collection of sources in a box $B$. For this, following the discussion in [13], we make use of the Hermite functions

$$h_n(x) = (-1)^n \frac{d^n}{dx^n} e^{-x^2}, \quad x \in \mathbb{R}.$$

Letting $D$ be a box with center $\mathbf{c}_D = (c_{D,1}, c_{D,2})$ and side length $2r\sqrt{\delta}$, we denote by $\phi(\mathbf{t}) = \phi(t_1,t_2)$ the field induced by a collection of $N_s$ sources $\mathbf{s}_j = (s_{j,1}, s_{j,2})$ lying in $D$:

$$\phi(\mathbf{t}) = \sum_{j=1}^{N_s} q_j e^{-\frac{|\mathbf{t}-\mathbf{s}_j|^2}{4\delta}}. \tag{5.1}$$

```
nx = length(x); ny = length(y);
[xs,I] = sort(x);                % sort the target points
[ys,J] = sort(y);                % sort the source points
beta = q(J);                     % align strength vector with sources
e1=exp(-ts*diff(xs));            % compute all needed complex exponentials
hp = zeros(length(ts),nx);
i = 1; j = 1;
while i <= nx && j <= ny         % sweep through targets and sources
  if ys(j) < xs(i)               % contribution from sources to the left
    hp(:,i) = hp(:,i)+beta(j)*exp(-ts*(xs(i)-ys(j)));
    j = j+1;
  elseif ys(j) == xs(i)          % if target and source are identical
    hp(:,i) = hp(:,i)+beta(j);
    j = j+1;
  else                           % forward recurrence over targets
    if i < nx
      hp(:,i+1) = e1(:,i).*hp(:,i);
    end
    i = i+1;
  end
end
```

Figure 7: MATLAB code fragment for the forward pass when the target points and source points are different.

A straightforward Taylor series calculation shows that

$$\phi(\mathbf{t}) = \sum_{l_1,l_2\geq 0} M_{l_1,l_2} h_{l_1}\left(\frac{t_1-c_{D,1}}{2\sqrt{\delta}}\right) h_{l_2}\left(\frac{t_2-c_{D,2}}{2\sqrt{\delta}}\right), \tag{5.2}$$

where

$$M_{l_1,l_2} = \frac{1}{l_1! l_2!}\sum_{j=1}^{N_s} q_j \left(\frac{s_{j,1}-c_{D,1}}{2\sqrt{\delta}}\right)^{l_1} \left(\frac{s_{j,2}-c_{D,2}}{2\sqrt{\delta}}\right)^{l_2}. \tag{5.3}$$

The error in truncating the Hermite expansion (5.2) when $l_1 = l_2 = p-1$ (with a total of $p^2$ terms) is given by

$$|E_H(p)| \leq K^2 Q_B (2S_r(p) + T_r(p)) T_r(p), \tag{5.4}$$

where

$$Q_B = \sum_{j=1}^{N_s} |q_j|, \tag{5.5}$$

$$S_r(p) = \sum_{n=0}^{p-1} \frac{r^n}{\sqrt{n!}}, \quad T_r(p) = \sum_{n=p}^{\infty} \frac{r^n}{\sqrt{n!}}, \tag{5.6}$$

and $K < 1.09$. It is important to note that the Hermite expansion (5.2) is valid for any target location in the plane, with the error controlled entirely by the box size parameter

$r$ and the expansion length $p$. We refer the reader to [13] and to [4,21,31,37] for further discussion.

Suppose now that $B$ is a second box with center $\mathbf{c}_B=(c_{B,1},c_{B,2})$ and side length $2r\sqrt{\delta}$ and that the target $\mathbf{t}$ lies in $B$. We may represent $\phi$ within $B$ as a truncated Taylor series of the form

$$\phi(\mathbf{t})=\sum_{l_1,l_2\le p-1} N_{l_1,l_2}\left(\frac{t_1-c_{B,1}}{2\sqrt{\delta}}\right)^{l_1}\left(\frac{t_2-c_{B,2}}{2\sqrt{\delta}}\right)^{l_2}. \tag{5.7}$$

Assuming the field $\phi$ is induced by the sources in $D$, there is a standard translation operator that maps the truncated Hermite expansion for box $D$ to a truncated Taylor series for box $B$:

$$N_{l_1,l_2}=\frac{(-1)^{l_1+l_2}}{l_1!l_2!}\sum_{j_1,j_2\le p-1} M_{j_1,j_2}h_{l_1+j_1}\left(\frac{c_{B,1}-c_{D,1}}{2\sqrt{\delta}}\right)h_{l_2+j_2}\left(\frac{c_{B,2}-c_{D,2}}{2\sqrt{\delta}}\right). \tag{5.8}$$

Rather than discussing detailed error estimates, available in the literature, the important thing to note is that the number of terms needed grows with the box size. The estimates in [31, 37] show that for a box of size $2.5\sqrt{\delta}$, $p=20$ terms are needed to yield 10-digit accuracy. For a box of size $5\sqrt{\delta}$, 33 terms are needed to yield 10-digit accuracy and for a box of size $10\sqrt{\delta}$, 65 terms are needed.

For a specified precision $\epsilon$, let us define the cutoff length $R_{cut}$ by

$$R_{cut}=\sqrt{4\delta\log(1/\epsilon)}. \tag{5.9}$$

That is, $R_{cut}$ is the distance from a source at which the Gaussian has decayed to $\epsilon$. For ten digits, $R_{cut}\approx 9.6\sqrt{\delta}$. If we superimpose on the source and target distribution a grid of cells (boxes) with spacing $R_{cut}$, then - to precision $\epsilon$ - for a target in a box $B$, we need only consider the influence within $B$ itself and its nearest eight neighbors (see Fig. 8).

A very simple FGT can be implemented as follows: for each cell in a grid with spacing $R_{cut}$, compute the Hermite expansion induced by the sources within that cell. Second, convert the Hermite expansion to a Taylor series in each of its eight neighbors as well as itself. Third, for each box, add together all of the resulting Taylor series. Fourth, evaluate the resulting expansion at each target within the box. While this is a linearly scaling method, the associated constant is very large because, as indicated above $p\approx 65$ for 10-digit accuracy and the translation operators require $\mathcal{O}(p^3)$ operations.

In the original FGT, the suggested data structure used smaller boxes, resulting in much shorter expansions. As a result, however, many more boxes have to be taken into account before the influence of a Gaussian source can be considered negligible. With a spacing of $2.5\sqrt{\delta}$, for example, and a 10-digit tolerance, $9^2=81$ boxes are within the region that needs to be considered. Thus, 81 translation operators need to be applied for each box.

In the plane-wave (Fourier-transform) versions of the FGT [14,29], two improvements were made. First, translation is diagonal in a basis of complex exponentials and second,

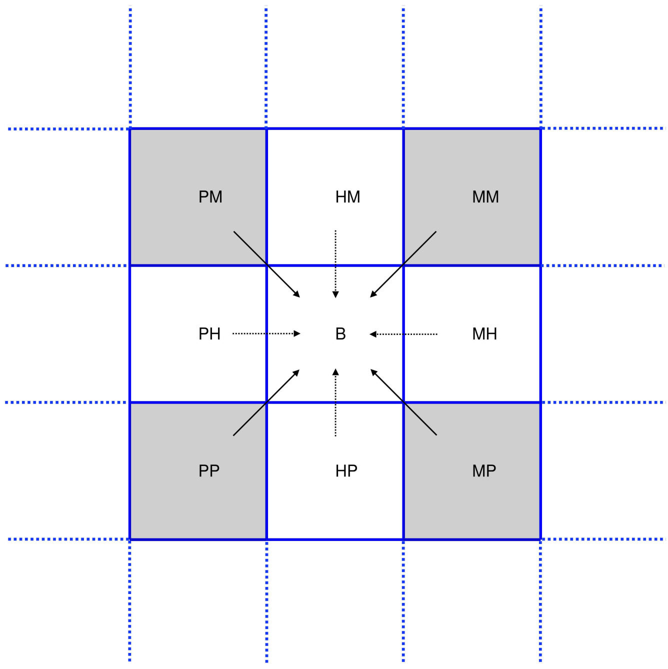

Figure 8: Consider the simple FGT using a grid of boxes with dimension equal to the cutoff length $R_{cut}$. For the central box $B$, we must accurately represent the field to sources in $B$ itself and its nearest neighbors (shown). (More distant interactions are exponentially small.) While the Hermite expansion at this spatial scale requires an exorbitant number of terms, a directional, SOE-based expansion can be extremely efficient. Such expansions can be constructed for any of the grey boxes, which are *fully separated* from $B$ in the sense of Definition 5.1. For 10-digit accuracy, these expansions require only $K_e/2\times Ke$ terms and represent the field for any target inside $B$, according to (1.4) with $K_e=12$ for a total of 72 exponentials. Four different types of expansions are need, denoted by $PM$,$MM$,$PP$ and $MP$. $P$ in the first position indicates that all targets in the box $B$ lie in the positive $x$ direction with respect to the source box, while $M$ in the first position indicated that all targets in the box $B$ lie in the negative $x$ direction with respect to the source box. The same notion applies in the second position, but with respect to the $y$ direction. Expansions for the remaining boxes, which are separated in one variable only, can be obtained using SOE approximation in the separated direction and Hermite approximation in the remaining direction (see (5.10)). Four expansion types are needed here as well, denoted by $PH$,$MH$,$HP$ and $HM$. $P$ and $M$ have the same meaning as above and $H$ denotes the use of Hermite approximation in the corresponding coordinate direction. See Algorithm 1.

sweeping algorithms reduce the number of translations from 81 to 6 (more generally, $9^d\to 3d$). Unfortunately, it is difficult to develop an efficient sweeping method that can be used on an adaptive data structure. For this, hierarchical versions of the FGT were developed [21,38], using quad-tree or oct-tree subdivisions of space.

Given the one-dimensional analysis above, it seems compelling to develop an SOE approximation that permits both short expansions and diagonal translation as well. Unfortunately, unlike the Fourier/plane-wave expansion, the SOE approximation requires removal of the absolute value notation to permit diagonal translation. Thus, it is conditionally convergent in the sense described in (1.6). For our scheme, we will need to introduce two-dimensional SOE expansions and hybrid SOE-Hermite expansions as well.

**Definition 5.1.** Suppose that $B=[a,a+L]\times[b,b+L]$ is a box of side length $L$ centered at $c_B=(a+L/2,b+L/2)$ and that $D=[c,c+L']\times[d,d+L']$ is a box of side length $L'$ centered at $(c+L'/2,d+L'/2)$. We say that $B$ and $D$ are *fully separated* if $[a,a+L]\cap[c,c+L']=\emptyset$ and $[b,b+L]\cap[d,d+L']=\emptyset$ (see Fig. 8).

If $B$ and $D$ are fully separated with $a>c$ and $b>d$, then the field defined by (5.1) can be represented in $B$ (with 10-digit accuracy) in the form: $\phi(\mathbf{t})\approx\Re\mathrm{e}(PP[D](\mathbf{t}))$ with

$$PP[D](\mathbf{t}))=\sum_{k=1}^{K_e/2}\sum_{l=1}^{K_e}S_{k,l}e^{-t_k(t_1-(c+L'/2))/\sqrt{\delta}}e^{-t_l(t_2-(d+L'/2))/\sqrt{\delta}},$$

where

$$S_{k,l}=w_kw_l\sum_{j=1}^{N_s}e^{+t_k(s_1-(c+L'/2))/\sqrt{\delta}}e^{+t_l(t_2-(d+L'/2))/\sqrt{\delta}}.$$

This follows immediately from (1.4). The notation $PP[D]$ is meant to convey that $B$ is fully separated from $D$ with $a>c$ and $b>d$. The formulae for fully separated boxes with with $a>c,b<d$, $a<c,b>d$, and $a<c,b<d$ are essentially the same, with appropriate sign flips to account for the absolute value in (1.2). We denote these SOE-expansions by $PM[D]$, $MP[D]$, and $MM[D]$, respectively.

If $B$ and $D$ are not fully separated, we can construct hybrid SOE-Hermite expansions. In this case, assuming $a>c$, $b=d$ and $L=L'$, then the field defined by (5.1) can be represented in $B$ (with 10-digit accuracy) in the form: $\phi(\mathbf{t})\approx\Re\mathrm{e}(PH[D](\mathbf{t}))$ with

$$PH[D](\mathbf{t})=\sum_{k=1}^{K_e/2}\sum_{l=0}^{p}J_{k,l}e^{-t_k(t_1-(c+L/2))/\sqrt{\delta}}h_l\left(\frac{t_2-(d+L/2)}{2\sqrt{\delta}}\right), \tag{5.10}$$

where

$$J_{k,l}=w_k\sum_{j=1}^{N_s}e^{+t_k(s_{j,1}-(c+L/2))/\sqrt{\delta}}\left(\frac{s_{j,2}-(d+L/2)}{2\sqrt{\delta}}\right)^l.$$

This follows again from (1.2), Hermite expansion in the second coordinate alone, and the tensor product nature of the Gaussian. The notation $PH[D]$ is meant to convey that $B$ is separated from $D$ in $x$ with $a>c$. The expansion when $B$ is separated from $D$ in $x$ but $a<c$ is denoted by $MH[D]$. When $B$ is separated from $D$ in $y$, the analogous hybrid SOE-Hermite expansions are denoted by $HP[D]$ and $HM[D]$. The formulas are straightforward and analogous, taking appropriate care of sign flips to account for the absolute value in (1.2).

**Remark 5.1.** All of the SOE-SOE and SOE-Hermite expansions are amenable to diagonal translation. For the hybrid SOE-Hermite expansions, this requires that the expansion center in the coordinate where Hermite expansions are being used doesn't change. More precisely, for a source box (such a the lower left corner in Fig. 8, with $PP$ expansion whose coefficients are $\{S_{k,l}\}$, the shifted $PP[B]$ expansion centered at $(a+L/2,b+L/2)$ has coefficients

$$S'_{k,l}=S_{k,l}e^{+t_k((a+L/2)-(c+L'/2))/\sqrt{\delta}}e^{+t_l((b+L/2)-(d+L'/2))/\sqrt{\delta}}.$$

Likewise, for a source box such as the one immediately to the left of $B$ in Fig. 8, with $PH$ expansion coefficients $\{J_{k,l}\}$, the shifted $PH[B]$ expansion centered at $(a+L/2,b+L/2)$ has coefficients

$$J'_{k,l}=J_{k,l}e^{+t_k((a+L/2)-(c+L'/2))/\sqrt{\delta}}.$$

The remaining translation operators are analogous.

**Remark 5.2.** Conversion of an Hermite expansion into an SOE-based expansion or an SOE-based expansion into a Taylor series is easy to carry out because of the tensor product structure of the Gaussian. We only need the corresponding one-dimensional formulas. For this, the Hermite expansion

$$\phi(t)=\sum_{i=0}^{p}A_ih_i\left(\frac{t-c}{2\sqrt{\delta}}\right) \tag{5.11}$$

can be converted to a $P$ or $M$-type SOE expansion

$$P(t;c,\delta)=\sum_{j=1}^{K_e}C_je^{-t_j(t-c)/\sqrt{4\delta}}, \tag{5.12}$$

and

$$M(t;c,\delta)=\sum_{j=1}^{K_e}D_je^{t_j(t-c)/\sqrt{4\delta}}, \tag{5.13}$$

with

$$C_j=w_j\sum_{i=0}^{p}t_j^iA_i,\quad D_j=w_j\sum_{i=0}^{p}(-t_j)^iA_i, \tag{5.14}$$

where $\{w_j\}$ are the weights in the SOE expansion of the Gaussian in (1.2) and $K_e$ is the number of exponentials used in the approximation. Likewise, the SOE expansions (5.12) and (5.13) can be converted to a Taylor expansion $T(t;c,\delta)$ with

$$T(t;c,\delta)=\sum_{i=0}^{p}B_i\left(\frac{t-c}{\sqrt{4\delta}}\right)^i, \tag{5.15}$$

where

$$B_i=\sum_{j=1}^{K_e}\frac{(-t_j)^i}{i!}C_j,\quad B_i=\sum_{j=1}^{K_e}\frac{(t_j)^i}{i!}D_j. \tag{5.16}$$

There are many cases to consider, and we omit the details which are straightforward from the formulas above.

**Algorithm 1** A simple "SH"-based FGT in two dimensions.

```
1: Divide the computational box uniformly into boxes of side length $R_{cut}$, where $R_{cut}$ is the cutoff length.
2: Sort sources and targets into these small boxes.
3: Form the Hermite expansion with respect to each source box center.
4: Convert the Hermite expansion into the $PP,PM,MP,MM,PH, MH,HP,HM$ expansions with respect to the source box center.
5: For each target box, translate eight expansions from neighbors to target box center.
6: Convert all eight expansions into Taylor series with respect to the target box center and add to the box's own Taylor series (induced by the sources within the box).
7: Evaluate the Taylor series at each target point.
```

We now have all the machinery required to develop a nonadaptive, single-level SOE-Hermite-based scheme, which we will abbreviate as an "SH scheme", outlined in Algorithm 1.

A few remarks are in order. The reader may wonder why we begin with an Hermite expansion rather than forming the SOE-SOE or hybrid SOE-Hermite expansions from the source locations directly. This is a matter of code optimization, and mainly avoids the calculation of complex exponentials for each source point. The operators mapping Hermite expansions to SOE-SOE or hybrid expansions (which do involve such exponentials) can be precomputed and stored. The second thing to recall is that while the SOE expansion lengths are independent of the box size, that is not true of the Hermite expansion where, as noted earlier, $p$ is very large for a box of dimension $R_{cut}$. A *multi-level* version of the above scheme, however, can make repeated use of SOE-SOE expansions until the box size is small enough that the Hermite expansion length is acceptable.

A three-level version of the scheme is outlined in Fig. 9. In the left figures ("level 1"), the box dimensions are the cutoff-length $R_{cut}$, and we translate only the highly efficient $PP,PM,MP$ and $MM$ expansions to the box $B$. Those expansions are then translated to $B$'s four children at level 2, where the box dimension is now $R_{cut}/2$. At this level, there are additional boxes which are fully separated from the child boxes. Focusing on the child marked $C_1$, we can shift the seven indicated $PP,PM,MP$ and $MM$ expansions to the center of $C_1$ and add to those obtained from the parent $B$. That step can be repeated at level 3, where additional boxes are now fully separated from the child $C_2$ of dimension $R_{cut}/4$. The $n$-level version of the SH scheme for the FGT is described in Algorithm 2.

We omit the tedious but straightforward analysis of exactly which boxes correspond to the various pure SOE and hybrid SOE-Hermite interactions at each level. Compared with the original FGT or the Fourier transform/plane wave based schemes without shifting, the new SH scheme requires fewer translations with shorter expansion lengths. The three-level scheme, for example, requires 42 translations for each target box, and this number can be further reduced by a systematic merging of expansions before shifting. We have not implemented such optimizations in the existing implementation.

**Algorithm 2** The $n$-level hybrid SH scheme for the FGT.

```
 1: At each level $n,n-1,\cdots,1$, divide the computational domain uniformly into boxes of
    side length $L=R_{cut}/2^{n-1}$, where $R_{cut}$ is the cutoff length. These grids come with
    a natural hierarchy of parent/child relations, with each box at level $l-1$ being the
    parent of four child boxes at level $l$ for $l=n,n-1,\cdots,2$.
 2: Bin sort sources and targets into the boxes at level $n$.
 3: Form the Hermite expansion with respect to each source box center at level $n$.
 4: Convert the Hermite expansion into the corresponding $PP,PM,MP,MM$ SOE expan-
    sions and the corresponding $PH,MH,HP,HM$ hybrid SOE-Hermite expansions with
    respect to the box centers at level $n$.
 5: for $l=n-1,\cdots,1$ do                          ▷ merge outgoing expansions
 6:     Translate the $PP,PM,MP,MM$ SOE expansions from child boxes to their parent.
 7: end for
 8: for $l=1,\cdots,n-1$ do                          ▷ translate SOE expansions
 9:     Shift $PP,PM,MP,MM$ expansions of source boxes to relevant target boxes (as out-
    lined in Fig. 9).
10: end for
11: At level $n$, shift $PH,MP,HP,HM$ expansions from source boxes to relevant target
    boxes (as outlined in Fig. 9).
12: for l$=1,\cdots,n-1$ do                          ▷ propagate to children
13:     Shift $PP,PM,MP,MM$ expansions of the parent box to its children.
14: end for
15: Convert the Hermite expansion to a Taylor expansion for each target box at level $n$.
16: Convert the incoming $PP,PM,MP,MM$ and $PH,MP,HP,HM$ expansions to a Taylor
    expansion for each target box at level $n$ and add to the Taylor expansion created in
    the previous step.
17: For each target box, evaluate the resulting Taylor expansion at each target lying in the
    box.
```

## 6 Numerical results

We have implemented the algorithms in Section 4 and Section 5 in Fortran, compiled using gfortran 6.3.0 with the -O3 flag, and run on a single core of a laptop with an Intel(R) 2.10GHz i7-4600U CPU.

### 6.1 One dimensional performance

For the initial sorting algorithm, we have modified the function dlasrt.f from LAPACK 3.8.0 so that it outputs the sorted array $xs$ and an integer array $I$ with $xs=x(I)$. The function uses Quick Sort, reverting to Insertion Sort on arrays of size $\leq 20$. Thus, the average complexity of the sorting step in our implementation is $\mathcal{O}(N\log N)$. In some

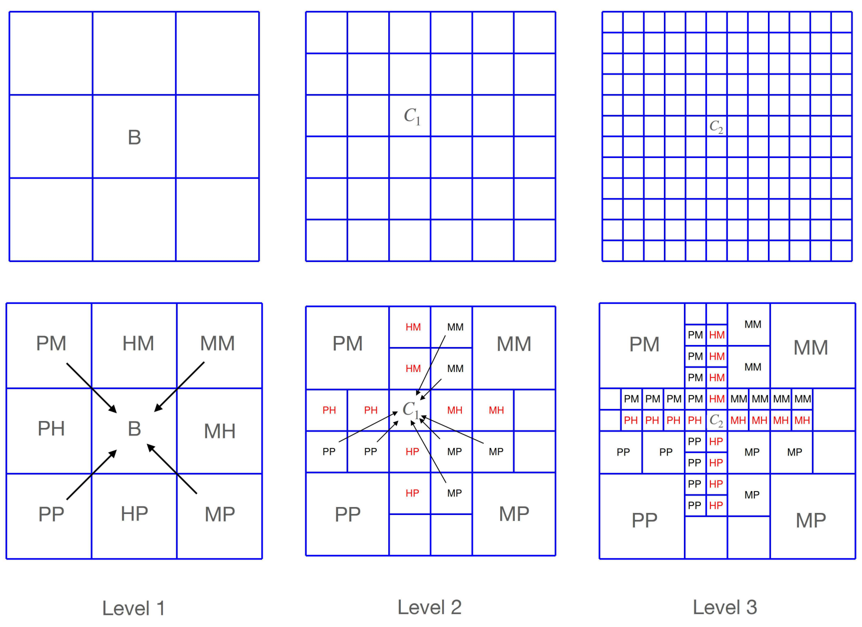

Figure 9: In a three-level SH scheme, we begin at level 1 (left), where the side length of each box equals the cutoff length $R_{cut}$. Only the fully separated interactions are accumulated in a typical box (denoted here by $B$). At level 2 (middle), the side length is now equal to $R_{cut}/2$ and additional fully separated interactions can be accounted for using pure SOE approximations. At level 3 (right), additional fully separated boxes of side length $R_{cut}/4$ can again be accounted for using pure SOE approximations. At this level, we shift the indicated $PH,MH,HP$ and $HM$ expansions as well. The length of the Hermite expansion is now much smaller since (for 10-digit accuracy) $R_{cut}/4 \approx 2.5\sqrt{\delta}$ and $p=20$.

applications, one may need to apply the Gauss transform many times with fixed point locations but different strengths $\{q_j\}$. In this case, it is advantageous to pre-sort the points and to precompute and store all needed complex exponentials in a table.

We first assume that the target points are coincident with the source points. We have tested both uniform distributions (on $[0,1]$ and points located at the Chebyshev nodes which cluster toward the interval endpoints. Table 1 shows the results for one sample run of a uniform distribution with $\delta=1$. In the table, $N$ is the total number of source/-target points, $n_e = K_e/2$ is the actual number of complex exponentials needed in the computation, $t_{\text{sort}}$ is the time for the sorting step, and $t_{\text{pre}}$ is the time for precomputing all complex exponentials. $t_{\text{FGT}}$ is the time for the FGT and $t_{\text{total}}$ is the total computation time - all measured in seconds. The error is the estimated maximum relative error computed at 100 randomly selected target points. The performance results for points distributed at Chebyshev nodes are essentially the same, as expected. Note that the most expensive step is precomputing all of the complex exponentials, and that sorting already takes a significant portion of the total computational time for the numbers $N$ reported here. Indeed, if we precompute the exponentials, then the code spends about as much time in sorting as in the FGT itself.

Table 1: Results of the FGT when the sources and targets coincide, with points chosen from a uniform distribution.

| $N$ | $n_e$ | $t_{\text{sort}}$ | $t_{\text{pre}}$ | $t_{\text{FGT}}$ | $t_{\text{total}}$ | Error |
|---|---|---|---|---|---|---|
| 100,000 | 3 | 0.008 | 0.016 | 0.008 | 0.036 | $4.4\times10^{-6}$ |
| 1,000,000 | 3 | 0.092 | 0.16 | 0.068 | 0.34 | $4.3\times10^{-6}$ |
| 10,000,000 | 3 | 1.0 | 1.6 | 0.64 | 3.8 | $4.3\times10^{-6}$ |
| 100,000 | 4 | 0.008 | 0.020 | 0.004 | 0.032 | $5.5\times10^{-8}$ |
| 1,000,000 | 4 | 0.088 | 0.21 | 0.080 | 0.40 | $5.5\times10^{-8}$ |
| 10,000,000 | 4 | 1.0 | 2.1 | 0.78 | 4.4 | $5.5\times10^{-8}$ |
| 100,000 | 5 | 0.008 | 0.024 | 0.004 | 0.044 | $6.3\times10^{-10}$ |
| 1,000,000 | 5 | 0.092 | 0.26 | 0.096 | 0.46 | $6.2\times10^{-10}$ |
| 10,000,000 | 5 | 1.0 | 2.5 | 0.94 | 5.1 | $5.6\times10^{-10}$ |
| 100,000 | 6 | 0.008 | 0.032 | 0.008 | 0.048 | $7.6\times10^{-12}$ |
| 1,000,000 | 6 | 0.088 | 0.30 | 0.12 | 0.53 | $4.9\times10^{-12}$ |
| 10,000,000 | 6 | 1.0 | 3.1 | 1.1 | 5.7 | $9.5\times10^{-11}$ |

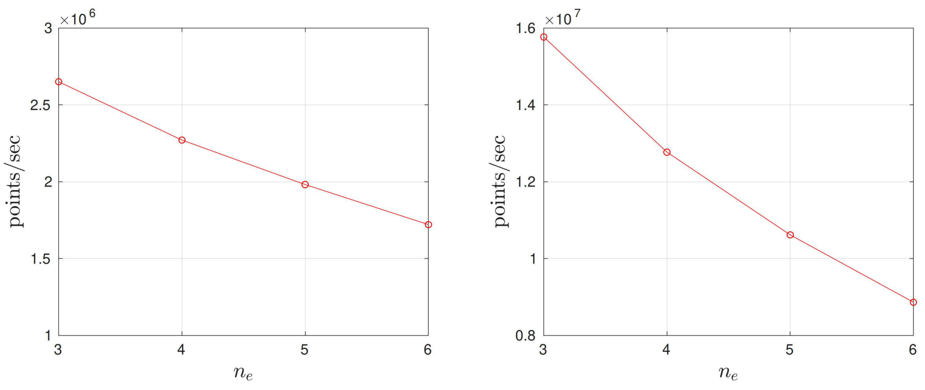

Figure 10: Throughput of the 1D FGT as a function of $n_e$ when targets are identical to sources. $n_e = K_e/2$ is half the number of exponentials in the SOE approximation of the Gaussian kernel. Left: Throughput of the FGT without any precomputation. Right: Throughput of the FGT when points are pre-sorted and exponentials are precomputed and stored.

The throughput is shown in Fig. 10, where each data point is obtained by averaging 10 sample runs with $N$ ranging from $10^6$ to $10^7$. We observe that the throughput of the whole algorithm ranges from 1.7 to 2.6 million points per second as the number of SOE terms $K_e$ decreases from 12 to 6. Assuming all exponentials have been stored and the points are sorted, the throughput ranges from 9 to 16 million points per second.

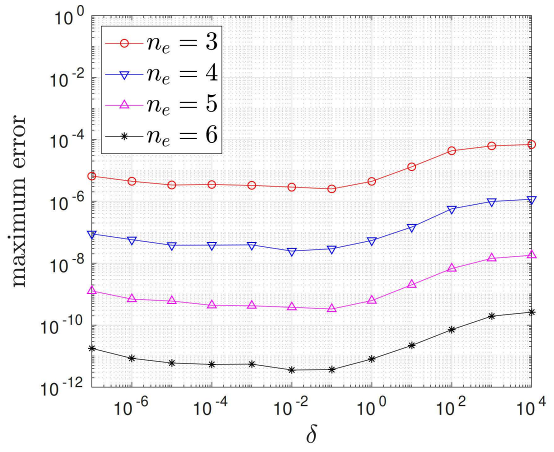

Figure 11: Accuracy of the 1D FGT as a function of $\delta$.

It is clear that the complexity of the algorithm is independent of $\delta$. Fig. 11 shows the accuracy dependence on $\delta$ for $\delta=10^{-7},\cdots,10^{4}$. We observe that the errors slowly increases as $\delta$ increases, and saturates at the errors shown in Fig. 4.

Next we consider the one-dimensional FGT when the target points are distinct from the source points. The forward (left-to-right) pass is described by the MATLAB code fragment in Fig. 7 and the backward (right-to-left) pass is similar. This requires calculating the exponentials $e^{-t_k\frac{x_{i+1}-x_i}{\sqrt{\delta}}}$ to march from one target to the next. It also requires the calculation of exponentials with the argument $x_{i+1}-x_i$ replaced by $x_i-y_j$ or $y_j-x_i$. Results for points drawn from a uniform distribution on $[0,1]$ are presented in Table 2 and Fig. 12. In Table 2, $t_{\text{linear}}$ is the computation time excluding the sorting step. Note that the cost is about double that in Table 1, since both target and source points need to be sorted. $t_{\text{linear}}$ is almost double as well since there are twice as many exponentials that needed to be calculated. Fig. 12 shows the throughput of the FGT, where the data points are obtained as in Fig. 10. The throughput of the full algorithm ranges from 0.95 to 1.5 million points per second for $K_e=12,\cdots,6$. With precomputation, the throughput of the algorithm will be very close to the right panel of Fig. 10 - on the order of 10 million points per second.

## 6.2 Two-dimensional performance

We now consider the speed of the new SH scheme for the FGT in two dimensions. Unlike the sweeping algorithm for the one-dimensional FGT, in which we made no use of a box structure and whose performance is independent of $\delta$, the performance of the SH scheme depends on the number of boxes created and the magnitude of $\delta$ plays a significant role. This is generally true for tree-based algorithms, including existing FGTs and the fast multipole method, where much of the work is proportional to the number of boxes

Table 2: Results of the 1D FGT on different target and source points.

| $N=M$ | $n_e$ | $t_{\text{sort}}$ | $t_{\text{linear}}$ | $t_{\text{total}}$ | Error |
|---|---|---|---|---|---|
| 100,000 | 3 | 0.0160 | 0.0440 | 0.0640 | $4.4\times10^{-6}$ |
| 1,000,000 | 3 | 0.188 | 0.432 | 0.664 | $4.4\times10^{-6}$ |
| 10,000,000 | 3 | 2.05 | 4.18 | 6.75 | $4.3\times10^{-6}$ |
| | | | | | |
| 100,000 | 4 | 0.0120 | 0.0520 | 0.0680 | $5.6\times10^{-8}$ |
| 1,000,000 | 4 | 0.188 | 0.556 | 0.856 | $5.5\times10^{-8}$ |
| 10,000,000 | 4 | 2.04 | 5.41 | 8.00 | $5.5\times10^{-8}$ |
| | | | | | |
| 100,000 | 5 | 0.0160 | 0.0680 | 0.0760 | $4.2\times10^{-9}$ |
| 1,000,000 | 5 | 0.172 | 0.672 | 0.876 | $6.2\times10^{-10}$ |
| 10,000,000 | 5 | 2.04 | 6.76 | 9.18 | $5.7\times10^{-10}$ |
| | | | | | |
| 100,000 | 6 | 0.0120 | 0.0760 | 0.0920 | $7.9\times10^{-12}$ |
| 1,000,000 | 6 | 0.172 | 0.792 | 1.01 | $6.8\times10^{-12}$ |
| 10,000,000 | 6 | 2.20 | 8.31 | 11.0 | $1.0\times10^{-10}$ |

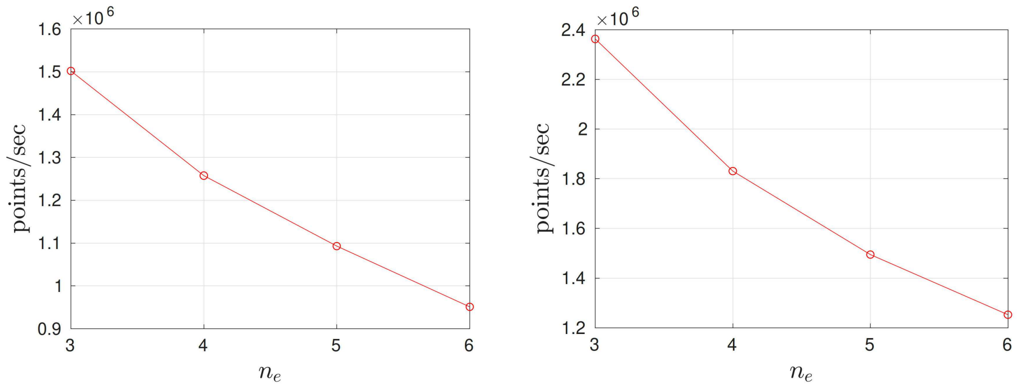

Figure 12: Similar to Fig. 10, but targets and sources are different. Left: Throughput of the FGT without any precomputation. Right: Throughput of the FGT when points are pre-sorted.

in the tree hierarchy. Following the original FGT [13], we study performance for two very different distributions of sources and targets. For the first, they are uniformly distributed in the unit box and, for the second, they are distributed along a circle of radius 0.5, with $\delta=10^{-1}$, $10^{-3}$, and $10^{-5}$. The desired precision is set to $\epsilon=10^{-10}$. In Tables 3 and 4, the first column lists the number of sources and targets. Both sources and targets are sorted on the tree and the potential is evaluated at *both* sources and targets. The second column lists the values of $\delta$. The third column lists the time for sorting the points. The fourth

Table 3: Results of the SH scheme of the 2D FGT when $N$ sources and $M$ targets are uniformly distributed in the unit box.

| $N=M$ | $\delta$ | $t_{\text{tree}}$ | $t_{\text{FGT}}$ | $t_{\text{total}}$ | Error |
|---|---|---|---|---|---|
| 100,000 | $1.0\times10^{-1}$ | 0.036 | 0.052 | 0.1 | $5.8\times10^{-13}$ |
| 1,000,000 | $1.0\times10^{-1}$ | 0.44 | 0.492 | 1.03 | $5.8\times10^{-13}$ |
| 10,000,000 | $1.0\times10^{-1}$ | 4.48 | 4.86 | 10.3 | $5.8\times10^{-13}$ |
| | | | | | |
| 100,000 | $1.0\times10^{-3}$ | 0.0400 | 0.216 | 0.268 | $3.8\times10^{-12}$ |
| 1,000,000 | $1.0\times10^{-3}$ | 0.548 | 0.840 | 1.49 | $5.3\times10^{-12}$ |
| 10,000,000 | $1.0\times10^{-3}$ | 6.55 | 7.20 | 14.8 | $5.2\times10^{-12}$ |
| | | | | | |
| 100,000 | $1.0\times10^{-5}$ | 0.028 | 3.57 | 3.67 | $3.3\times10^{-12}$ |
| 1,000,000 | $1.0\times10^{-5}$ | 0.456 | 7.05 | 7.78 | $2.3\times10^{-11}$ |
| 10,000,000 | $1.0\times10^{-5}$ | 5.52 | 18.5 | 25.3 | $2.2\times10^{-11}$ |

Table 4: Results of the SH scheme of the 2D FGT when $N$ sources and $M$ targets are uniformly distributed on a circle.

| $N=M$ | $\delta$ | $t_{\text{tree}}$ | $t_{\text{FGT}}$ | $t_{\text{total}}$ | Error |
|---|---|---|---|---|---|
| 100,000 | $1.0\times10^{-1}$ | 0.028 | 0.052 | 0.088 | $1.9\times10^{-12}$ |
| 1,000,000 | $1.0\times10^{-1}$ | 0.388 | 0.488 | 0.968 | $1.4\times10^{-12}$ |
| 10,000,000 | $1.0\times10^{-1}$ | 3.82 | 4.94 | 9.62 | $1.5\times10^{-12}$ |
| | | | | | |
| 100,000 | $1.0\times10^{-3}$ | 0.032 | 0.148 | 0.192 | $7.6\times10^{-12}$ |
| 1,000,000 | $1.0\times10^{-3}$ | 0.448 | 0.792 | 1.34 | $7.1\times10^{-12}$ |
| 10,000,000 | $1.0\times10^{-3}$ | 5.00 | 7.13 | 13.1 | $5.0\times10^{-12}$ |
| | | | | | |
| 100,000 | $1.0\times10^{-5}$ | 0.044 | 3.22 | 3.44 | $1.1\times10^{-11}$ |
| 1,000,000 | $1.0\times10^{-5}$ | 0.352 | 4.38 | 5.01 | $1.2\times10^{-11}$ |
| 10,000,000 | $1.0\times10^{-5}$ | 4.35 | 15.7 | 21.2 | $1.8\times10^{-11}$ |

column lists the time for the FGT, including the time for initialization of all arrays and all precomputation. The fifth column lists the total computation time. The last column lists the relative $l^2$ error as compared with direct calculation at a random subset of 20 points.

As can be seen from Tables 3 and 4, uniform distributions generally have worse performance than nonuniform distributions. Moreover, the average number of points per box greatly affects the throughput, especially for small $\delta$. The apparent sublinear scaling in the tables for $\delta=10^{-5}$ is a consequence of the fact that many boxes are created in the algorithm because of the cutoff length $R_{cut}$. With a workload proportional to the number of boxes, $N$ and $M$ would have to be much larger before linear scaling is more evident.

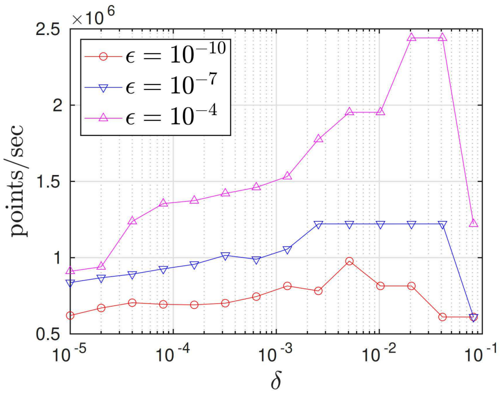

Figure 13: Throughput of the SH scheme for the FGT in two dimensions (measured in units of million points/second) with various precisions, as a function of $\delta$.

In the context of solving diffusion problems, it is natural to assume that the number of points per leaf node is in the range 16 to 64, corresponding to a 4th or 8th order discretization (as described, for example, in [38] in the context of volume integral versions of the FGT). Fig. 13 shows the throughput of the new SH scheme. It appears to have better throughput than the hierarchical FGT implementation in [38].

## 7 Conclusions

As for several of the functions discussed in [28, 36], best rational approximations to the exponential function on $\mathbb{R}^-$ may be used to construct nearly optimal sum-of-exponential (SOE) approximations for the Gaussian kernel. New fast Gauss transforms in one and two dimensions have been built upon such approximations. For one dimension, a simple sweeping scheme leads to an algorithm whose performance is independent of the Gaussian variance. For two dimensions, a nonadaptive hybrid SOE-Hermite scheme has been constructed, which is easily modified to work on adaptive data structures.

An interesting theoretical question is whether the number of exponentials can be further reduced. Those obtained by best rational approximations need roughly $n+2$ exponentials to achieve $n$-digit accuracy. We are currently developing fully adaptive versions of the FGT in two and three dimensions, for both point sources and continuous distributions. We expect that the resulting scheme will be more efficient than current methods for any value of the Gaussian variance, and that it can serve as the computational core for a variety of simulation tools involving diffusion and heat transfer.

## Acknowledgments

S. Jiang was supported in part by the United States National Science Foundation under grant DMS-1720405.

## References

[1] V. M. Adamjan, D. Z. Arov, and M. G. Kreĭn. Infinite Hankel matrices and generalized Carathéodory-Fejér and I. Schur problems. *Funkcional. Anal. i Priložen.*, 2(4):269–281, 1968.

[2] V. M. Adamjan, D. Z. Arov, and M. G. Kreĭn. Infinite Hankel matrices and generalized problems of Carathéodory-Fejér and F. Riesz. *Funkcional. Anal. i Priložen.*, 2(1):1–18, 1968.

[3] V. M. Adamjan, D. Z. Arov, and M. G. Kreĭn. Analytic properties of the Schmidt pairs of a Hankel operator and the generalized Schur-Takagi problem. *Mat. Sb. (N.S.)*, 86(128):34–75, 1971.

[4] B. J. C. Baxter and G. Roussos. A new error estimate of the fast Gauss transform. *SIAM J. Sci. Comput.*, 24:257–259, 2002.

[5] G. Beylkin and L. Monzón. On approximation of functions by exponential sums. *Appl. Comput. Harmon. Anal.*, 19(1):17–48, 2005.

[6] G. Beylkin and L. Monzón. Approximation by exponential sums revisited. *Appl. Comput. Harmon. Anal.*, 28(2):131–149, 2010.

[7] A. J. Carpenter, A. Ruttan, and R. S. Varga. Extended numerical computations on the "1/9" conjecture in rational approximation theory. In *Rational approximation and interpolation (Tampa, Fla., 1983)*, volume 1105 of *Lecture Notes in Math.*, pages 383–411. Springer, Berlin, 1984.

[8] B. Dingfelder and J. A. C. Weideman. An improved Talbot method for numerical Laplace transform inversion. *Numer. Algorithms*, 68(1):167–183, 2015.

[9] Z. Gimbutas, N. F. Marshall, and V. Rokhlin. A fast simple algorithm for computing the potential of charges on a line. *arXiv preprint arXiv:1907.03873*, 2019.

[10] K. Glover. All optimal Hankel-norm approximations of linear multivariable systems and their $L^\infty$-error bounds. *Internat. J. Control*, 39(6):1115–1193, 1984.

[11] A. A. Gonchar and E. A. Rakhmanov. Equilibrium distributions and the rate of rational approximation of analytic functions. *Math. USSR-Sb.*, 62(2):305–348, 1989.

[12] L. Greengard, S. Jiang, and Y. Zhang. The anisotropic truncated kernel method for convolution with free-space Green's functions. *SIAM J. Sci. Comput.*, 40(6):A3733–A3754, 2018.

[13] L. Greengard and J. Strain. The fast Gauss transform. *SIAM J. Sci. Statist. Comput.*, 12:79–94, 1991.

[14] L. Greengard and X. Sun. A new version of the fast Gauss transform. *Documenta Mathematica, III*, pages 575–584, 1998.

[15] M. H. Gutknecht. On complex rational approximation. II. The Carathéodory-Fejér method. In *Computational aspects of complex analysis (Braunlage, 1982)*, volume 102 of *NATO Adv. Sci. Inst. Ser. C: Math. Phys. Sci.*, pages 103–132. Reidel, Dordrecht-Boston, Mass., 1983.

[16] M. H. Gutknecht. Rational Carathéodory-Fejér approximation on a disk, a circle, and an interval. *J. Approx. Theory*, 41(3):257–278, 1984.

[17] M. H. Gutknecht and L. N. Trefethen. Real polynomial Chebyshev approximation by the Carathéodory-Fejér method. *SIAM J. Numer. Anal.*, 19(2):358–371, 1982.

[18] M. H. Gutknecht and L. N. Trefethen. Real and complex Chebyshev approximation on the unit disk and interval. *Bull. Amer. Math. Soc. (N.S.)*, 8(3):455–458, 1983.

[19] S. Jiang. A fast Gauss transform in one dimension using sum-of-exponentials approximations. *arXiv preprint arXiv:1909.09825*, 2019.

[20] S. Jiang, L. Greengard, and S. Wang. Efficient sum-of-exponentials approximations for the heat kernel and their applications. *Adv. Comput. Math.*, 41(3):529–551, 2015.

[21] D. Lee, A. Gray, and A. Moore. Dual-tree fast Gauss transforms. *Advances in Neural Informa-*

*tion Processing Systems*, 18:747–754, 2006.

[22] J.-R. Li and J. White. Reduction of large curcuit models via low rank approximate gramians. *Int. J. Appl. Math. Comp. Sci.*, 11:1151–1171, 2001.

[23] F. F. Lin. *Numerical inversion of Laplace transforms by the trapezoidal-type methods*. ProQuest LLC, Ann Arbor, MI, 2004. Thesis (Ph.D.)–Oregon State University.

[24] M. López-Fernández and C. Palencia. On the numerical inversion of the Laplace transform of certain holomorphic mappings. *Appl. Numer. Math.*, 51(2-3):289–303, 2004.

[25] M. López-Fernández, C. Palencia, and A. Schädle. A spectral order method for inverting sectorial Laplace transforms. *SIAM J. Numer. Anal.*, 44(3):1332–1350, 2006.

[26] V. L. Makarov and I. P. Gavrilyuk. Exponentially convergent parallel discretization method for the first order evolution equation. *Appl. Math. Inform.*, 5(2):47–69, 2000.

[27] V. V. Peller. *Hankel operators and their applications*. Springer Monographs in Mathematics. Springer-Verlag, New York, 2003.

[28] T. Schmelzer and L. N. Trefethen. Evaluating matrix functions for exponential integrators via Carathéodory-Fejér approximation and contour integrals. *Electron. Trans. Numer. Anal.*, 29:1–18, 2007/08.

[29] M. Spivak, S. K. Veerapaneni, and L. Greengard. The fast generalized Gauss transform. *SIAM J. Sci. Comput.*, 32(5):3092–3107, 2010.

[30] F. Stenger. *Numerical methods based on sinc and analytic functions*, volume 20 of *Springer Series in Computational Mathematics*. Springer-Verlag, New York, 1993.

[31] J. Strain. The fast Gauss transform with variable scales. *SIAM J. Sci. Stat. Comput.*, 12:1131–1139, 1991.

[32] A. Talbot. The accurate numerical inversion of Laplace transforms. *J. Inst. Math. Appl.*, 23(1):97–120, 1979.

[33] L. N. Trefethen and M. H. Gutknecht. The Carathéodory-Fejér method for real rational approximation. *SIAM J. Numer. Anal.*, 20(2):420–436, 1983.

[34] L. N. Trefethen and M. H. Gutknecht. Real vs. complex rational Chebyshev approximation on an interval. *Trans. Amer. Math. Soc.*, 280(2):555–561, 1983.

[35] L. N. Trefethen and J. A. C. Weideman. The exponentially convergent trapezoidal rule. *SIAM Rev.*, 56(3):385–458, 2014.

[36] L. N. Trefethen, J. A. C. Weideman, and T. Schmelzer. Talbot quadratures and rational approximations. *BIT*, 46(3):653–670, 2006.

[37] X. Wan and G. Karniadakis. A sharp error estimate for the fast Gauss transform. *Journal of Computational Physics*, 219:7–12, 2006.

[38] J. Wang and L. Greengard. An adaptive fast Gauss transform in two dimensions. *SIAM J. Sci. Comput.*, 40(3):A1274–A1300, 2018.

[39] J. A. C. Weideman. Improved contour integral methods for parabolic PDEs. *IMA J. Numer. Anal.*, 30(1):334–350, 2010.

[40] J. A. C. Weideman and L. N. Trefethen. Parabolic and hyperbolic contours for computing the Bromwich integral. *Math. Comp.*, 76(259):1341–1356, 2007.

[41] K. Xu and S. Jiang. A bootstrap method for sum-of-poles approximations. *J. Sci. Comput.*, 55(1):16–39, 2013.