A new version of the Fast Multipole Method for the evaluation of potential fields on one-dimensional structures is introduced. The scheme uses a new representation of potential fields, based on generalized Gaussian quadratures [6,9]; in this representation, most translation operators are diagonal. To incorporate this representation into the FMM, an apparatus is introduced for transforming between different types of expansions; this apparatus is somewhat general, and is based on formulae for the least squares approximation of linear operators. The performance of the method is illustrated with several numerical examples; it is roughly twice as fast as previously published algorithms.

An Improved Fast Multipole Algorithm for Potential Fields On One-Dimensional Structures

N. Yarvin and V. Rokhlin
Research Report YALEU/DCS/RR-1119
January 24, 1997

The authors were supported in part by DARPA/AFOSR under Grant F49620-95-1-0575, in part by ONR under Grants N00014-89-J-1527 and N00014-96-1-0188, and in part by a fellowship from the Fannie and John Hertz Foundation.
Approved for public release: distribution is unlimited.
Keywords: *Singular Value Decompositions, Fast Algorithms.*

1 Introduction

This paper describes an algorithm for the following problem: given two finite sequences $x_1, \ldots, x_n \in \mathbf{R}$ and $q_1, \ldots, q_n \in \mathbf{R}$, compute the quantities $\phi_1, \ldots, \phi_n \in \mathbf{R}$ given by the formula

$$\phi_j = \sum_{i=1, i \neq j}^{n} \frac{q_i}{x_j - x_i}. \tag{1}$$

This is identical to the following physical problem: n point charges are located on a line at positions $x_1, \ldots, x_n$, with the magnitudes of the charges being $q_1, \ldots, q_n$ respectively. At each of the locations x_i, the electrostatic potential ϕ_i which is due to the charges at all the other points is to be computed.

An algorithm is also presented for the variation of the above problem in which $\phi_1, \ldots, \phi_n$ are given by the formula

$$\phi_j = \sum_{i=1, i \neq j}^{n} q_i \log |x_j - x_i|. \tag{2}$$

This is identical to another physical problem: electric charge is present on n parallel, infinitely thin wires of infinite length which lie at positions $x_1, \ldots, x_n$ in a plane. At each of those positions, the electrostatic potential which is due to the other wires is to be computed.

One algorithm for these problems is the Fast Multipole Method (FMM) [3]. The FMM is an $O(n)$ scheme, which was first formulated for the two-dimensional case, that is, the computation of (2) in the case that $x_1, \ldots, x_n \in \mathbf{C}$. It uses a hierarchy of multipole expansions to compute potentials which are due to distant charges. A version of the FMM specialized to the one-dimensional case was later published [2].

The algorithm described in this paper is a variation of the FMM which differs from the FMMs of [3] and [2] in two respects. The first change is that potentials due to nearby charges are computed using a new numerical tool, namely expansions based on generalized Gaussian quadratures [6, 9]. The second change is that the multipole expansions of [3] are replaced by expansions based on singular value decompositions of integral operators. These two changes are linked together by an apparatus for transforming between different types of expansions; this apparatus is somewhat general, and is based on formulae for least squares approximation of linear operators.

This paper is organized as follows. Section 2 contains existence theorems for singular value decompositions of linear integral operators, and theorems on least squares approximation of such operators. Section 3.1 briefly describes certain standard numerical tools used by the algorithm. Section 3.2 contains derivations of error bounds for expansions based on generalized Gaussian quadratures. Section 4 contains various analytical results used in the construction of the algorithm. Section 5 contains a primitive algorithm for (1) or (2) which uses quadrature-based expansions exclusively, and which displays $O(n)$ behavior for small n, although it is asymptotically an $O(n^2)$ algorithm. Section 6 contains the main algorithm of this paper, which is an $O(n)$ algorithm for an arbitrary number of points. Finally, Section 7 contains experimental results.

2 Mathematical Preliminaries

2.1 Singular value decomposition

The singular value decomposition (SVD) is a ubiquitous tool in numerical analysis, which is given for the case of real matrices by the following lemma (see, for instance, [8] for more details).

Lemma 2.1 *For any $n \times m$ real matrix A, there exist, for some integer p, an $n \times p$ real matrix U with orthonormal columns, an $m \times p$ real matrix V with orthonormal columns, and a $p \times p$ real diagonal matrix $S = [s_{ij}]$ whose diagonal entries are non-negative, such that $A = U \cdot S \cdot V^*$ and that $s_{ii} \leq s_{i+1,i+1}$ for all $i = 1, \ldots, p-1$.*

The diagonal entries s_{ii} of S are called singular values; the columns of the matrix V are called right singular vectors; the columns of the matrix U are called left singular vectors.

2.2 Singular value decomposition of integral operators

This section, which follows [4], contains an existence theorem for a factorization of integral operators. The operators $T : L^2[c,d] \to L^2[a,b]$ to which it applies are of the form

$$(T \cdot f)(x) = \int_c^d K(x,t) f(t) dt. \tag{3}$$

in which the function $K : [a,b] \times [c,d] \to \mathbf{R}$ is referred to as the kernel of the operator T. Throughout this section, it will be assumed that all functions are square-integrable; the term "norm" will mean the L^2 norm.

The following theorem, which defines the factorization, is proven in a more general form as Theorem VI.17 in [7].

Theorem 2.2 *Suppose that the function $K : [a,b] \times [c,d] \to \mathbf{R}$ is square integrable. Then there exist two orthonormal sequences of functions $u_i : [a,b] \to \mathbf{R}$ and $v_i : [c,d] \to \mathbf{R}$ and a sequence $s_i \in \mathbf{R}$, for $i = 1, \ldots, \infty$, such that*

$$K(x,t) = \sum_{i=1}^{\infty} u_i(x) s_i v_i(t) \tag{4}$$

and that $s_1 \geq s_2 \geq \ldots \geq 0$. The sequence s_i is uniquely determined by K.

By analogy to the finite-dimensional case, we will refer to this factorization as the singular value decomposition. We will refer to the functions u_i as left singular functions of K (or of T), to v_i as right singular functions, and to s_i as singular values.

As is the case for the discrete singular value decomposition, this decomposition can be used to construct an approximation to the function K, by discarding small singular values and the associated singular functions:

$$K(x,t) \simeq \sum_{i=1}^{p} u_i(x) s_i v_i(t). \tag{5}$$

The error of this approximation can then be computed from (4):

$$K(x,t) - \sum_{i=1}^{p} u_i(x) s_i v_i(t) = \sum_{i=p+1}^{\infty} u_i(x) s_i v_i(t), \tag{6}$$

and, therefore,

$$\left|\left| K(x,t) - \sum_{i=1}^{p} u_i(x) s_i v_i(t) \right|\right| = \sqrt{\sum_{i=p+1}^{\infty} s_i^2}. \tag{7}$$

2.2.1 Notation

A more compact notation for the above SVD, in which the analogy to the discrete singular value decomposition is more clearly displayed, is as follows. Let p be the number of singular values which are greater than zero. Let the operator $U : \mathbf{R}^p \to L^2[a,b]$ be defined by the formula

$$(Uy)(x) = \sum_{i=1}^{p} y_i u_i(x), \tag{8}$$

where the numbers $y_1, \ldots, y_p \in \mathbf{R}$ are the elements of the vector $y \in \mathbf{R}^p$. Let the operator $V : \mathbf{R}^p \to L^2[c,d]$ be defined by the formula

$$(Vy)(x) = \sum_{i=1}^{p} y_i v_i(x), \tag{9}$$

and let the matrix S be the $p \times p$ diagonal matrix whose i'th diagonal entry is s_i. Then the SVD can be written as

$$T = U \cdot S \cdot V^*. \tag{10}$$

In addition the orthogonality properties of the functions $\{u_i\}$ and $\{v_i\}$ can be written as

$$U^* \cdot U = I, \tag{11}$$

$$V^* \cdot V = I. \tag{12}$$

2.2.2 Singular value decomposition of linear operators with finite-dimensional input and infinite-dimensional output

The following theorem is analogous to Theorem 2.2, from which it easily follows (the proof is omitted). The theorem applies to operators whose input is finite-dimensional and whose output is infinite-dimensional, or vice versa; in particular, to, operators whose kernel is a function of the form $K : [a,b] \times \mathcal{N} \to \mathbf{R}$, for some interval $[a,b]$, where $\mathcal{N}$ denotes the set of integers i such that $1 \le i \le n$, for some integer $n > 0$.

Theorem 2.3 *Suppose that the function $K : [a,b] \times \mathcal{N} \to \mathbf{R}$ is square integrable. Then there exist a finite orthonormal sequence of functions $u_1, \ldots, u_p : [a,b] \to \mathbf{R}$, an $n \times p$ matrix $V = [v_{ij}]$*

with orthonormal columns, and a sequence $s_1 \geq s_2 \geq \ldots \geq s_p > 0 \in \mathbf{R}$, *for some integer* p, *such that*

$$K(x,j) = \sum_{i=1}^{p} u_i(x) s_i v_{ij}, \tag{13}$$

for all $x \in [a,b]$ *and all* $j = 1, \ldots, n$. *The sequence* $\{s_i\}$ *is uniquely determined by* K.

As in the case of the factorization of integral operators presented in Section 2.2, we will refer to the above factorization as a singular value decomposition; since these factorizations apply to different objects, the meaning is unambiguous. We will refer to the functions $\{u_i\}$ as singular functions, to the columns of the matrix V as singular vectors, and to the numbers $\{s_i\}$ as singular values.

A more compact notation for the above factorization is as follows. Let the operator $U : \mathbf{R}^p \rightarrow L^2[a,b]$ and the $p \times p$ matrix S be defined as in Section 2.2.1. Let the operator $T : \mathbf{R}^n \rightarrow L^2[a,b]$ be defined by the formula

$$(T \cdot y)(x) = \sum_{i=1}^{n} y_i K(x,i), \tag{14}$$

where the numbers $y_1, \ldots, y_n \in \mathbf{R}$ are the elements of the vector $y \in \mathbf{R}^n$. Then equation (13) can also be written as

$$T = U \cdot S \cdot V^*, \tag{15}$$

and the orthogonality properties of the singular functions and singular vectors can be written as

$$U^* \cdot U = I, \tag{16}$$

$$V^* \cdot V = I. \tag{17}$$

2.3 Least squares approximation of linear operators

This section contains six lemmas on least squares approximation of linear operators. In this section, and in the remainder of the paper, the vector norms and function norms used are L^2 norms, except where explicitly indicated otherwise. For more complicated objects, the norms used are as follows.

- 1. For an $n \times m$ real matrix $A = [a_{ij}]$, the norm is

$$||A|| = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij}^2}. \tag{18}$$

- 2. For a linear operator $V : \mathbf{R}^n \rightarrow L^2[a,b]$ which is given by the formula

$$(V \cdot y)(x) = \sum_{i=1}^{n} y_i v_i(x), \tag{19}$$

the norm is

$$||V|| = \sqrt{\sum_{i=1}^{n} \int_a^b (v_i(x))^2 dx}. \tag{20}$$

The norm of its transpose $V^* : L^2[a,b] \to \mathbf{R}^n$ is identical.

- 3. For an integral operator $T : L^2[c,d] \to L^2[a,b]$ which is given by the formula

$$(T \cdot f)(x) = \int_c^d K(x,t) f(t) dt, \tag{21}$$

the norm is

$$||T|| = \sqrt{\int_c^d \int_a^b (K(x,t))^2 dx dt}. \tag{22}$$

The proof of the following lemma is an exercise in elementary matrix algebra, and is omitted.

Lemma 2.4 *Suppose that U is an $n \times p$ matrix with orthonormal columns, that is, $U^*U = I$. Then for any $n \times m$ matrix B,*

$$||B||^2 = ||(I - U \cdot U^*)B||^2 + ||U \cdot U^* \cdot B||^2. \tag{23}$$

The following simple lemma is well-known in the case when $B = I$ and $k = 1$.

Lemma 2.5 *Suppose A is a $p \times n$ real matrix, B is a $m \times k$ real matrix, and C is an $p \times k$ real matrix, for some m, p, n, and k. Let $A = U_A \cdot S_A \cdot V_A^*$ be a singular value decomposition of A, and let $B = U_B \cdot S_B \cdot V_B^*$ be a singular value decomposition of B. Then an $n \times m$ real matrix X which minimizes the quantity $||A \cdot X \cdot B - C||$ is given by the formula*

$$X = V_A \cdot S_A^{-1} \cdot U_A^* \cdot C \cdot V_B \cdot S_B^{-1} \cdot U_B^*. \tag{24}$$

The quantity which is thus minimized is given by the formula

$$||A \cdot X \cdot B - C|| = ||C - U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^*||. \tag{25}$$

Proof. Using the singular value decompositions of A and B, we have

$$||A \cdot X \cdot B - C||^2 = ||U_A \cdot S_A \cdot V_A^* \cdot X \cdot U_B \cdot S_B \cdot V_B^* - C||^2. \tag{26}$$

Using Lemma 2.4 and equation (26), we get

$$\begin{aligned} ||A \cdot X \cdot B - C||^2 &= ||(I - U_A \cdot U_A^*)(U_A \cdot S_A \cdot V_A^* \cdot X \cdot U_B \cdot S_B \cdot V_B^* - C)||^2 \\ &+ ||(U_A \cdot U_A^*)(U_A \cdot S_A \cdot V_A^* \cdot X \cdot U_B \cdot S_B \cdot V_B^* - C)||^2. \end{aligned} \tag{27}$$

Using the fact that $U_A^* \cdot U_A = I$, it follows that

$$||A \cdot X \cdot B - C||^2 = ||(I - U_A \cdot U_A^*)C||^2 + ||U_A \cdot S_A \cdot V_A^* \cdot X \cdot U_B \cdot S_B \cdot V_B^* - U_A \cdot U_A^* \cdot C||^2. \tag{28}$$

Using Lemma 2.4 and equation (28), we have

$$\begin{aligned} ||A \cdot X \cdot B - C||^2 &= ||(I - U_A \cdot U_A^*)C||^2 \\ &+ ||(U_A \cdot S_A \cdot V_A^* \cdot X \cdot U_B \cdot S_B \cdot V_B^* - U_A \cdot U_A^* \cdot C)(I - V_B \cdot V_B^*)||^2 \\ &+ ||(U_A \cdot S_A \cdot V_A^* \cdot X \cdot U_B \cdot S_B \cdot V_B^* - U_A \cdot U_A^* C)(V_B \cdot V_B^*)||^2. \end{aligned} \quad (29)$$

Since $V_B^* \cdot V_B = I$,

$$\begin{aligned} ||A \cdot X \cdot B - C||^2 &= ||(I - U_A \cdot U_A^*)C||^2 \\ &+ ||U_A \cdot U_A^* \cdot C(I - V_B \cdot V_B^*)||^2 \\ &+ ||U_A \cdot S_A \cdot V_A^* \cdot X \cdot U_B \cdot S_B \cdot V_B^* - U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^*||^2. \end{aligned} \quad (30)$$

Since $U_A^* \cdot U_A = I$,

$$\begin{aligned} ||A \cdot X \cdot B - C||^2 &= ||(I - U_A \cdot U_A^*)(C - U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^*)||^2 \\ &+ ||U_A \cdot U_A^*(C - C \cdot V_B \cdot V_B^*)||^2 \\ &+ ||U_A \cdot S_A \cdot V_A^* \cdot X \cdot U_B \cdot S_B \cdot V_B^* - U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^*||^2. \quad (31) \\ &= ||(I - U_A \cdot U_A^*)(C - U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^*)||^2 \\ &+ ||U_A \cdot U_A^*(C - U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^*)||^2 \\ &+ ||U_A \cdot S_A \cdot V_A^* \cdot X \cdot U_B \cdot S_B \cdot V_B^* - U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^*||^2. \quad (32) \end{aligned}$$

Using Lemma 2.4, it follows that

$$||A \cdot X \cdot B - C||^2 = ||C - U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^*||^2 + ||U_A \cdot S_A \cdot V_A^* \cdot X \cdot U_B \cdot S_B \cdot V_B^* - U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^*||^2. \quad (33)$$

The quantity $||A \cdot X \cdot B - C||^2$ which was to be minimized has now been broken into two parts, one of which is not dependent upon X and one of which is. Substituting (24) into the latter part, we get

$$\begin{aligned} &||U_A \cdot S_A \cdot V_A^* \cdot X \cdot U_B \cdot S_B \cdot V_B^* - U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^*||^2 \\ &= ||U_A \cdot S_A \cdot V_A^* \cdot V_A \cdot S_A^{-1} \cdot U_A^* \cdot C \cdot V_B \cdot S_B^{-1} \cdot U_B^* \cdot U_B \cdot S_B \cdot V_B^* - U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^*||^2 \\ &= ||U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^* - U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^*||^2 \\ &= 0. \end{aligned} \quad (34)$$

Thus, by the substitution (24), the quantity $||A \cdot X \cdot B - C||$ is minimized, with the minimum value being given by (25). □

It can easily be seen that the above proof does not depend on the operators A, B, C, and X being finite-dimensional, except in two respects. The first respect is that Lemma 2.4 is used; however that lemma possesses an infinite-dimensional counterpart, Lemma 2.6 (below), which is equally simple to prove and whose proof is again omitted. The second respect is that for infinite-dimensional operators A and B, the convergence of the formula (24) is problematic; however, convergence is obviously assured in the case that the operator C has finite norm, and that each of the operators A and B either has finite norm and finite rank, or is the identity operator. Thus Lemma 2.7 (below) holds.

Lemma 2.6 *Suppose that $U : \mathbf{R}^p \to \Gamma_2$ and $B : \Gamma_1 \to \Gamma_2$ are linear operators such that each of the spaces Γ_1 and Γ_2 is either a function space of the form $L^2[a,b]$ for some $a, b \in \mathbf{R}$, or a vector space of the form $\mathbf{R}^m$ for some m. Suppose in addition that $U^* \cdot U = I$, and that B has finite norm. Then*

$$||B||^2 = ||(I - U \cdot U^*)B||^2 + ||U \cdot U^* \cdot B||^2. \tag{35}$$

Lemma 2.7 *Suppose the operators $A : \Gamma_2 \to \Gamma_1$, $B : \Gamma_4 \to \Gamma_3$, and $C : \Gamma_4 \to \Gamma_1$, are linear operators, such that each of the spaces $\Gamma_1, \ldots, \Gamma_4$ is either a function space of the form $L^2[a,b]$ for some $a, b \in \mathbf{R}$, or a vector space of the form $\mathbf{R}^p$ for some p. Suppose in addition that C has finite norm, and that each of the operators A and B is either the identity operator, or has finite norm and finite rank. Let the singular value decompositions of A and B be denoted by $A = U_A \cdot S_A \cdot V_A^*$ and $B = U_B \cdot S_B \cdot V_B^*$ respectively. Then an operator $X : \Gamma_3 \to \Gamma_2$ which minimizes the quantity $||A \cdot X \cdot B - C||$ is given by the formula*

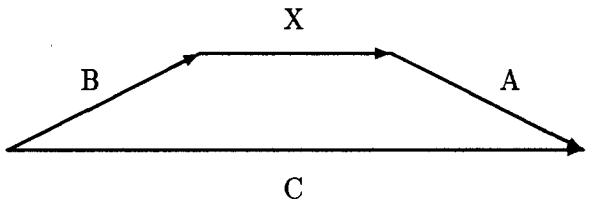
$$X = V_A \cdot S_A^{-1} \cdot U_A^* \cdot C \cdot V_B \cdot S_B^{-1} \cdot U_B^*. \tag{36}$$

The quantity which is thus minimized is given by the formula

$$||A \cdot X \cdot B - C|| = ||C - U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^*||. \tag{37}$$

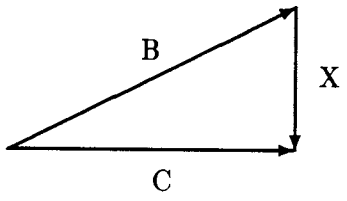
Remark 2.1 *An obvious application of Lemma 2.7 is depicted in Figure 1, and is as follows. If B is an operator which generates an expansion (such as a multipole expansion) from a function on some interval Γ_1, A is an operator which evaluates an expansion (perhaps of a different type, such as a Taylor series) on a distant interval Γ_4, and C is the operator which generates the desired result on Γ_4 directly from the input on Γ_1, then Lemma 2.7 provides a formula for a matrix X such that constructing an expansion with B, converting it with X, then evaluating it with A, produces a result which is as close as possible (in the least squares sense) to the result produced using C alone.*

Figure 1: The operators of Lemma 2.7



Remark 2.2 *Lemma 2.7 also yields an alternative formula for a matrix which converts between two types of expansions. If (as depicted in Figure 2) B is an operator which generates an expansion from a function on some interval Γ_1, and C is a operator which generates a different type of expansion from a function on Γ_1, then (setting A to the identity matrix) Lemma 2.7 provides a formula for a matrix X such that constructing an expansion with B, then converting it with X, produces an expansion which is as close as possible (in the least squares sense) to the expansion produced using C alone.*

Figure 2: The operators of Remark 2.2



The following lemma provides a bound, in certain situations, on the error of the approximation given by Lemma 2.7.

Lemma 2.8 *Under the conditions of Lemma 2.7, suppose that there exist linear operators* $D : \Gamma_4 \to \Gamma_2$ *and* $E : \Gamma_3 \to \Gamma_1$ *such that*

$$||A \cdot D - C|| < \varepsilon_1, \tag{38}$$

and

$$||E \cdot B - C|| < \varepsilon_2. \tag{39}$$

Then

$$||U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^* - C|| < \varepsilon_1 + \varepsilon_2. \tag{40}$$

Proof. By Lemma 2.7, the minimum value of $||A \cdot Y - C||$, for any linear operator $Y : \Gamma_2 \to \Gamma_3$, is given by $||C - U_A \cdot U_A^* \cdot C||$. Thus, using (38),

$$||C - U_A \cdot U_A^* \cdot C|| < \varepsilon_1. \tag{41}$$

Similarly, from (39), it follows that

$$||C - C \cdot V_B \cdot V_B^*|| < \varepsilon_2. \tag{42}$$

Using the triangle inequality, we get

$$\begin{aligned} ||U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^* - C|| &= ||U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^* - C \cdot V_B \cdot V_B^* + C \cdot V_B \cdot V_B^* - C|| \\ &\leq ||(U_A \cdot U_A^* \cdot C - C)V_B \cdot V_B^*|| + ||C \cdot V_B \cdot V_B^* - C||. \end{aligned} \tag{43}$$

By Lemma 2.6, for any linear operator $Z : \Gamma_4 \to \Gamma_1$,

$$||Z \cdot V_B \cdot V_B^*|| \leq ||Z||. \tag{44}$$

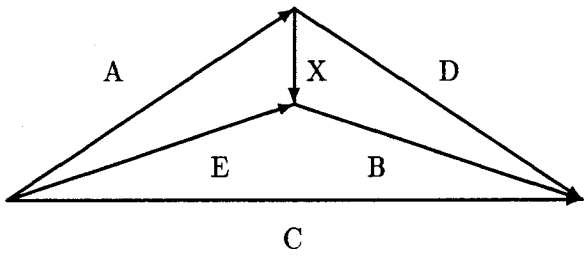
Combining (43) and (44), we get

$$\begin{aligned} ||U_A \cdot U_A^* \cdot C \cdot V_B \cdot V_B^* - C|| &\leq ||U_A \cdot U_A^* \cdot C - C|| + ||C \cdot V_B \cdot V_B^* - C|| \\ &< \varepsilon_1 + \varepsilon_2. \end{aligned} \tag{45}$$

□

Remark 2.3 *Lemma 2.8 provides an error bound for the application of Lemma 2.7 mentioned in Remark 2.1, which is depicted (with the additional operators introduced in Lemma 2.8) in Figure 3. It requires that both types of expansion (the type computed by B, and the type evaluated by A) be suitable for computing the desired output on the target interval Γ_1 from the input function on the source interval Γ_4. It then states that expansions can be converted accurately from one type to the other, using the conversion formula given in Lemma 2.7.*

Figure 3: The operators of Lemma 2.8



As shown by the following lemma, the error bound in Lemma 2.8 also applies when a different formula for the operator X is used.

Lemma 2.9 *Under the conditions of Lemma 2.8, let the operator $X : \Gamma_3 \to \Gamma_2$ be given by the formula*

$$X = D \cdot V_B \cdot S_B^{-1} \cdot U_B^*. \tag{46}$$

Then

$$||C - A \cdot X \cdot B|| < \varepsilon_1 + \varepsilon_2. \tag{47}$$

Proof. Using (46), we get

$$\begin{aligned} ||C - A \cdot X \cdot B|| &= ||C - A \cdot D \cdot V_B \cdot S_B^{-1} \cdot U_B^* \cdot B|| \\ &= ||C - A \cdot D \cdot V_B \cdot S_B^{-1} \cdot U_B^* \cdot U_B \cdot S_B \cdot V_B^*|| \\ &= ||C - A \cdot D \cdot V_B \cdot V_B^*|| \\ &= ||C - C \cdot V_B \cdot V_B^* + C \cdot V_B \cdot V_B^* - A \cdot D \cdot V_B \cdot V_B^*|| \\ &= ||C(I - V_B \cdot V_B^*) + (C - A \cdot D)V_B \cdot V_B^*||. \end{aligned} \tag{48}$$

Applying the triangle inequality to (48), we obtain

$$||C - A \cdot X \cdot B|| \leq ||C(I - V_B \cdot V_B^*)|| + ||(C - A \cdot D)V_B \cdot V_B^*||. \tag{49}$$

Combining (44) and (49), we get

$$||C - A \cdot X \cdot B|| \leq ||C(I - V_B \cdot V_B^*)|| + ||C - A \cdot D||. \tag{50}$$

Then (47) follows immediately from (42), (38), and (50). □

Remark 2.4 *Lemma 2.9 provides an error bound for the application of Lemma 2.7 which is mentioned in Remark 2.2. It requires that the two expansions between which the conversion is made (the expansion produced by B, and the expansion produced by D) both be accurate on some interval Γ_3; it then states that the conversion matrix X given by (46) produces an expansion which is accurate on Γ_3.*

3 Numerical Preliminaries

3.1 Gaussian integration and interpolation

Classical Gaussian quadrature rules are a well-known numerical tool (see, for instance, [8]); they integrate polynomials of order $2n - 1$ exactly with respect to some weight function, and consist of n weights and nodes. A variety of Gaussian quadratures were analyzed in the last century, each being defined by a distinct weight function. Of these, the algorithm presented in this paper uses only the Gaussian quadratures for the weight function $\omega(x) = 1$ on the region of integration $[-1, 1]$. These quadratures are closely associated with the Legendre polynomials; we will refer to their nodes as Legendre nodes.

Another numerical tool used in this paper is polynomial interpolation on Legendre nodes. Interpolation refers to the following problem: given two finite real sequences $f_1, \ldots, f_n \in \mathbf{R}$ and $x_1, \ldots, x_n \in [a, b]$, construct a function $f : [a, b] \to \mathbf{R}$ such that $f(x_i) = f_i$ for all $i = 1, \ldots, n$. One interpolation scheme is polynomial interpolation, in which the interpolating function f is a polynomial of degree $n - 1$. As is well-known, such a polynomial always exists and is unique. However, in general two numerical difficulties arise with polynomial interpolation using polynomials of high order. The first is that for many sequences of points $\{x_i\}$, the values of the interpolating polynomial between the points $\{x_i\}$ are not well-conditioned as a function of the values $\{f_i\}$ to be interpolated. The second is that even for those sequences of points where the computation of the values of the interpolating polynomial is well-conditioned, the computation of the coefficients of the power series of the interpolating polynomial is extremely ill-conditioned.

As is well-known, these difficulties do not arise if the points $\{x_i\}$ are taken to be Chebyshev nodes and the interpolating polynomial is computed as a series of Chebyshev polynomials rather than as a power series. As is shown in [9], the difficulties also do not arise if the points $\{x_i\}$ are taken to be Legendre nodes and the interpolating polynomial is computed as a series of Legendre polynomials.

3.2 Exponential expansions

The following theorem is the basis for the algorithms presented in this paper for the evaluation of (1).

Theorem 3.1 *Suppose there exist numbers $a, b, \varepsilon \in \mathbf{R}$ such that the quadrature formula with weights $w_1, \ldots, w_m \in \mathbf{R}$ and nodes $t_1, \ldots, t_m \in \mathbf{R}$ satisfies the inequality*

$$\left| \int_0^\infty e^{-xt} dt - \sum_{j=1}^{m} w_j e^{-xt_j} \right| \leq \varepsilon, \tag{51}$$

for all $x \in [a,b]$. Suppose in addition that the points $x_1, x_2, \ldots, x_n \in \mathbf{R}$ are such that $x_1 < x_2 < \ldots < x_n$, that $x_n - x_1 \leq b$, and that $x_n - x_{n-1} \geq a$. Then

$$\left| \sum_{i=1}^{n-1} \frac{q_i}{x_n - x_i} - \sum_{j=1}^{m} w_j \alpha_j e^{(y-x_n)t_j} \right| \leq \varepsilon \sum_{i=1}^{n-1} |q_i|, \tag{52}$$

where the coefficients $\alpha_1, \ldots, \alpha_m \in \mathbf{R}$ are defined by the formula

$$\alpha_j = \sum_{i=1}^{n-1} q_i e^{(x_i - y)t_j}, \tag{53}$$

and where y is an arbitrary real number.

Proof. Let i be an integer such that $1 \leq i < n$. Then $(x_n - x_i) \in [a,b]$, so by (51),

$$\left| \int_0^\infty e^{-(x_n - x_i)t} dt - \sum_{j=1}^{m} w_j e^{-(x_n - x_i)t_j} \right| \leq \varepsilon. \tag{54}$$

Evaluating the integral, we get

$$\left| \frac{1}{x_n - x_i} - \sum_{j=1}^{m} w_j e^{-(x_n - x_i)t_j} \right| \leq \varepsilon. \tag{55}$$

Taking the sum over all charges yields the inequality

$$\left| \sum_{i=1}^{n-1} \frac{q_i}{x_n - x_i} - \sum_{i=1}^{n-1} q_i \sum_{j=1}^{m} w_j e^{-(x_n - x_i)t_j} \right| \leq \varepsilon \sum_{i=1}^{n-1} |q_i|, \tag{56}$$

thus

$$\left| \sum_{i=1}^{n-1} \frac{q_i}{x_n - x_i} - \sum_{j=1}^{m} w_j e^{-(x_n - y)t_j} \sum_{i=1}^{n-1} q_i e^{-(y - x_i)t_j} \right| \leq \varepsilon \sum_{i=1}^{n-1} |q_i|, \tag{57}$$

from which (52) follows immediately. □

We will refer to the expansion (52) as an exponential expansion, to the quantities $\alpha_1, \ldots, \alpha_n$ as its coefficients, and to the quantity y as its location. Clearly, an exponential expansion computed at one location can be moved to a different location without effect on its accuracy: given the coefficients $\alpha_1, \ldots, \alpha_m$ of an expansion located at any point y_1, the coefficients $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_m$ of an expansion located at a point y_2 which evaluates to the same potential are given by the obvious formula

$$\tilde{\alpha}_j = e^{(y_1 - y_2)t_j} \alpha_j. \tag{58}$$

3.2.1 Quadrature rescaling

In the computation of potentials using exponential expansions, the quadrature which is available often is not on the same scale as the points; that is, the requirement that the points $\{x_i\}$ are such that $x_n - x_1 \leq b$ and that $x_n - x_{n-1} \geq a$ is often not satisfied. This situation can sometimes be remedied by rescaling either the points or the quadrature. The former option is to perform the computations

$$x_i := sx_i, \tag{59}$$
$$q_i := sq_i, \tag{60}$$

for an appropriate number $s \in \mathbf{R}$, and for all $i = 1, \ldots, n$. As can be seen by inspection of (1), this does not change the resulting potential. The latter option is to perform the computations

$$w_j := sw_j, \tag{61}$$
$$t_j := st_j, \tag{62}$$

for an appropriate number $s \in \mathbf{R}$, and for all $j = 1, \ldots, m$; as can easily be verified, this changes the region of accuracy of the quadrature from $[a, b]$ to $[a/s, b/s]$. We will refer to the number a/s as the minimum range, and to the number b/s as the maximum range, of the expansions based on the rescaled quadrature.

3.2.2 Exponential expansions for the logarithmic potential

The following theorem is the basis for the algorithms presented in this paper for the evaluation of (2).

Theorem 3.2 *Suppose there exist numbers $a, b, \varepsilon \in \mathbf{R}$ such that the quadrature formula with weights $w_1, \ldots, w_m \in \mathbf{R}$ and nodes $t_1, \ldots, t_m \in \mathbf{R}$ satisfies the inequality*

$$\left| \int_0^\infty e^{-xt} dt - \sum_{j=1}^{m} w_j e^{-xt_j} \right| \leq \varepsilon, \tag{63}$$

for all $x \in [a, b]$. Suppose in addition that the points $x_1, x_2, \ldots, x_n \in \mathbf{R}$ are such that

$$x_1 < x_2 < \ldots < x_n, \tag{64}$$
$$x_n - x_1 \leq b, \tag{65}$$
$$x_n - x_{n-1} \geq a \quad . \tag{66}$$

Then

$$\left| \left(\sum_{i=1}^{n-1} q_i \log(x_n - x_i) \right) - \left(cp_{n-1} + \sum_{j=1}^{m} -\frac{w_j}{t_j} \alpha_j e^{-t_j(x_n - y)} \right) \right| \leq \varepsilon \sum_{i=1}^{n-1} |q_i(x_n - x_i - a)|, \tag{67}$$

where the coefficients $\alpha_1, \ldots, \alpha_m \in \mathbf{R}$ are defined by the formula

$$\alpha_j = \sum_{i=1}^{n-1} q_i e^{-t_j(y - x_i)}, \tag{68}$$

where $p_{n-1} \in \mathbf{R}$ is defined by the formula

$$p_{n-1} = \sum_{i=1}^{n-1} q_i, \tag{69}$$

where $c \in \mathbf{R}$ is defined by the formula

$$c = \log a - \sum_{j=1}^{m} -\frac{w_j}{t_j} e^{-t_j a}, \tag{70}$$

and where y is an arbitrary real number.

Proof. Let f be any number such that $f \in [a, b]$. Then, using (63), we get

$$\left| \int_a^f \frac{1}{y} dy - \int_a^f \sum_{j=1}^{m} w_j e^{-t_j y} dy \right| \leq \left| \int_a^f \varepsilon \, dy \right|. \tag{71}$$

Evaluating both integrals in the left hand side of 71 yields the inequality

$$\left| (\log f - \log a) - \left(\sum_{j=1}^{m} -\frac{w_j}{t_j} e^{-t_j f} - \sum_{j=1}^{m} -\frac{w_j}{t_j} e^{-t_j a} \right) \right| \leq \varepsilon \, |f - a|. \tag{72}$$

Due to (64-66), $(x_n - x_i) \in [a, b]$ for all $i = 1, 2, \ldots, n-1$; replacing f with $x_n - x_i$, we rewrite (72) in the form

$$\left| (\log(x_n - x_i) - \log a) - \left(\sum_{j=1}^{m} -\frac{w_j}{t_j} e^{-t_j (x_n - x_i)} - \sum_{j=1}^{m} -\frac{w_j}{t_j} e^{-t_j a} \right) \right| \leq \varepsilon |(x_n - x_i - a)|. \tag{73}$$

Summing up (73) for all $i = 1, 2, \ldots, n-1$, we have

$$\left| \sum_{i=1}^{n-1} q_i (\log(x_n - x_i) - \log a) - \sum_{i=1}^{n-1} q_i \left(\sum_{j=1}^{m} -\frac{w_j}{t_j} e^{-t_j (x_n - x_i)} - \sum_{j=1}^{m} -\frac{w_j}{t_j} e^{-t_j a} \right) \right| \leq \varepsilon \sum_{i=1}^{n-1} |q_i (x_n - x_i - a)|. \tag{74}$$

Thus

$$\left| \sum_{i=1}^{n-1} q_i \log(x_n - x_i) - \left(\sum_{i=1}^{n-1} q_i \sum_{j=1}^{m} -\frac{w_j}{t_j} e^{-t_j (x_n - x_i)} + \sum_{i=1}^{n-1} q_i \left(\log a - \sum_{j=1}^{m} -\frac{w_j}{t_j} e^{-t_j a} \right) \right) \right| \leq \varepsilon \sum_{i=1}^{n-1} |q_i (x_n - x_i - a)|. \tag{75}$$

Using (75) and the definition (69) of p_{n-1}, we get

$$\left| \sum_{i=1}^{n-1} q_i \log(x_n - x_i) - \left(\sum_{j=1}^{m} -\frac{w_j}{t_j} e^{-t_j(x_n - y)} \sum_{i=1}^{n-1} q_i e^{-t_j(y-x_i)} - c p_{n-1} \right) \right| \leq \varepsilon \sum_{i=1}^{n-1} |q_i(x_n - x_i - a)|. \tag{76}$$

Substituting (68) into (76) then immediately yields (67). □

3.2.3 Exponential expansions for the square root kernel

A similar theorem forms the basis for the algorithm presented in Section 5.2 for the following problem: given two finite sequences $x_1, \ldots, x_n \in \mathbf{R}$ and $q_1, \ldots, q_n \in \mathbf{R}$, compute the quantities $\phi_1, \ldots, \phi_n \in \mathbf{R}$ given by the formula

$$\phi_k = \sum_{i=1}^{k-1} \frac{q_i}{\sqrt{x_k^2 - x_i^2}}. \tag{77}$$

In this case the quadrature formula which is used is for the integral

$$\int_0^\infty I_0(yt) e^{-xt} dt = \frac{1}{\sqrt{x^2 - y^2}}. \tag{78}$$

(This, and formulae suitable for many other kernels, can be found in tables of Laplace transforms; see for instance [1].)

Theorem 3.3 *Suppose there exist numbers $a, b, \varepsilon \in \mathbf{R}$ such that the quadrature formula with weights $w_1, \ldots, w_m \in \mathbf{R}$ and nodes $t_1, \ldots, t_m \in \mathbf{R}$ satisfies the inequality*

$$\left| \int_0^\infty I_0(yt) e^{-xt} dt - \sum_{j=1}^{m} w_j I_0(y t_j) e^{-x t_j} \right| \leq \varepsilon, \tag{79}$$

for all $x \in [a, b]$ and $y \in [0, x - a]$. Suppose in addition that the points $x_1, x_2, \ldots, x_n \in \mathbf{R}$ are such that $0 \leq x_1 < x_2 < \ldots < x_n \leq b$, and that $x_n - x_{n-1} \geq a$. Then

$$\left| \sum_{i=1}^{n-1} \frac{q_i}{\sqrt{x_n^2 - x_i^2}} - \sum_{j=1}^{m} w_j \alpha_j e^{(y - x_n) t_j} \right| \leq \varepsilon \sum_{i=1}^{n-1} |q_i| \tag{80}$$

where the coefficients $\alpha_1, \ldots, \alpha_m \in \mathbf{R}$ are given by the formula

$$\alpha_j = \sum_{i=1}^{n-1} q_i (I_0(x_i t_j) / e^{x_i t_j}) e^{(x_i - y) t_j} \tag{81}$$

and where y is an arbitrary real number.

Proof. Since $0 \leq x_i < x_n \leq b$, and $x_n - x_{n-1} \geq a$, it follows that $x_n \in [a,b]$, and that $x_i \in [0, x_n - a]$. Thus, making the substitutions $x = x_n$ and $y = x_i$ in (79), we get

$$\left| \int_0^\infty I_0(x_i t) e^{-x_n t} dt - \sum_{j=1}^{m} w_j I_0(x_i t_j) e^{-x_n t_j} \right| \leq \varepsilon \tag{82}$$

$$\left| \frac{1}{\sqrt{x_n^2 - x_i^2}} - \sum_{j=1}^{m} w_j I_0(x_i t_j) e^{-x_n t_j} \right| \leq \varepsilon. \tag{83}$$

Therefore

$$\left| \sum_{i=1}^{n-1} \frac{q_i}{\sqrt{x_n^2 - x_i^2}} - \sum_{j=1}^{m} w_j \sum_{i=1}^{n-1} q_i I_0(x_i t_j) e^{-x_n t_j} \right| \leq \varepsilon \sum_{i=1}^{n-1} |q_i|, \tag{84}$$

from which (80) follows immediately. □

4 Analytical Apparatus

4.1 Approximation of singular value decompositions

4.1.1 Interpolation

This section contains two basic lemmas about interpolation. The following lemma shows that any interpolation scheme whose output depends linearly on its input is characterized by a finite sequence of functions $[a,b] \to \mathbf{R}$; it is proven in [9].

Lemma 4.1 *Suppose $L : \mathbf{R}^n \to L^2[a,b]$ is a interpolation scheme with n nodes $x_1, \ldots, x_n \in [a,b]$, and that L is a linear mapping. Then there exists a sequence of functions $\alpha_1, \ldots, \alpha_n : [a,b] \to \mathbf{R}$ such that for any vector $f \in \mathbf{R}^n$, with elements $f = (f_1, \ldots, f_n)^T$,*

$$(L \cdot f)(x) = \sum_{i=1}^{n} f_i \alpha_i(x), \tag{85}$$

for all $x \in [a,b]$.

In the case of polynomial interpolation, the functions α_i are referred to as Lagrange polynomials; by analogy to that case, we will in general refer to the functions α_i as the Lagrange functions of the interpolation scheme.

The following lemma provides an error bound for approximation of a function of two variables using two one-dimensional interpolation formulae, expressed in terms of error bounds for each one-dimensional interpolation scheme applied separately. Its proof is an exercise in elementary analysis, and is omitted.

Lemma 4.2 *Suppose that $x_1, x_2, \ldots, x_n \in [a,b]$ and $t_1, t_2, \ldots, t_m \in [c,d]$ are two finite real sequences, and that $\alpha_1, \alpha_2, \ldots, \alpha_n : [a,b] \to \mathbf{R}$ and $\beta_1, \beta_2, \ldots, \beta_m : [c,d] \to \mathbf{R}$ are two sequences of bounded functions. Suppose further that $L_1 : \mathbf{R}^n \to L^\infty[a,b]$ is an interpolation formula*

with the nodes $x_1, \ldots, x_n$ and Lagrange functions $\alpha_1, \ldots, \alpha_n$, and $L_2 : \mathbf{R}^m \rightarrow L^\infty[c,d]$ is an interpolation formula with the nodes $t_1, \ldots, t_m$ and Lagrange functions $\beta_1, \ldots, \beta_m$. Suppose that $\eta \in \mathbf{R}$ is such that

$$\sum_{i=1}^{n} |\alpha_i(x)| < \eta, \tag{86}$$

for all $x \in [a,b]$, or

$$\sum_{j=1}^{m} |\beta_j(t)| < \eta, \tag{87}$$

for all $t \in [c,d]$. Finally, suppose that K is a function $[a,b] \times [c,d] \rightarrow \mathbf{R}$, and that for all $x \in [a,b]$ and $t \in [c,d]$,

$$\left| K(x,t) - \sum_{i=1}^{n} K(x_i,t)\alpha_i(x) \right| < \varepsilon \tag{88}$$

and

$$\left| K(x,t) - \sum_{j=1}^{m} K(x,t_j)\beta_j(t) \right| < \varepsilon. \tag{89}$$

Then

$$\left| K(x,t) - \sum_{i=1}^{n}\sum_{j=1}^{m} K(x_i,t_j)\alpha_i(x)\beta_j(t) \right| < \varepsilon(1+\eta), \tag{90}$$

for all $x \in [a,b]$ and $t \in [c,d]$.

4.1.2 Approximation of the SVD of an integral operator

This section describes a numerical procedure for computing an approximation to the singular value decomposition of an integral operator.

The algorithm uses quadratures which possess the following property.

Definition 4.1 *We will say that the combination of a quadrature and an interpolation scheme* preserves inner products *on an interval $[a,b]$ if it possesses the following properties.*

- *1. The nodes of the quadrature are identical to the nodes of the interpolation scheme.*
- *2. The function which is output by the interpolation scheme depends in a linear fashion on the values input to the interpolation scheme.*
- *3. The quadrature integrates exactly any product of two interpolated functions; that is, for any two functions $f, g : [a,b] \rightarrow \mathbf{R}$ produced by the interpolation scheme, the integral*

$$\int_a^b f(x)g(x)dx \tag{91}$$

is computed exactly by the quadrature.

Quadratures and interpolation schemes which possess this property include:

Example 4.1 *The combination of a (classical) Gaussian quadrature at Legendre nodes and polynomial interpolation at the same nodes preserves inner products, since polynomial interpolation on n nodes produces an interpolating polynomial of order $n-1$, the product of two such polynomials is a polynomial of order $2n-2$, and a Gaussian quadrature integrates exactly all polynomials up to order $2n-1$.*

Example 4.2 *If an interval is broken into several subintervals, and a quadrature and interpolation scheme which preserves inner products is used on each subinterval, then the arrangement as a whole preserves inner products on the original interval. (This follows directly from the definition.)*

Example 4.3 *The combination of the trapezoidal rule on the interval $[0, 2\pi]$, and Fourier interpolation (using the interpolation functions $1, \cos x, \sin x, \cos 2x, \sin 2x, \ldots, \cos nx, \sin nx$) preserves inner products.*

The algorithm takes as input a function $K : [a,b] \times [c,d] \rightarrow \mathbf{R}$. It uses the following numerical tools:

- 1. A quadrature and an interpolation scheme on the interval $[a,b]$ which preserve inner products. Let the weights and nodes of this quadrature be denoted by $w_1^x, \ldots, w_n^x \in \mathbf{R}$ and $x_1, \ldots, x_n \in [a,b]$ respectively. Let the Lagrange functions (see Section 4.1.1) of the interpolation scheme be denoted by $\alpha_1, \ldots, \alpha_n : [a,b] \rightarrow \mathbf{R}$.

- 2. A quadrature and an interpolation scheme on the interval $[c,d]$ which preserve inner products. Let the weights and nodes of this quadrature be denoted by $w_1^t, \ldots, w_m^t \in \mathbf{R}$ and $t_1, \ldots, t_m \in [c,d]$ respectively. Let the Lagrange functions of the interpolation scheme be denoted by $\beta_1, \ldots, \beta_m : [c,d] \rightarrow \mathbf{R}$.

As will be shown below, the accuracy of the algorithm is then determined by the accuracy to which the above two interpolation schemes approximate K.

The output of the algorithm is a sequence of functions $u_1, \ldots, u_p : [a,b] \rightarrow \mathbf{R}$, a sequence of functions $v_1, \ldots, v_p : [c,d] \rightarrow \mathbf{R}$, and a sequence of singular values $s_1, \ldots, s_p \in \mathbf{R}$, which form an approximation to the singular value decomposition of K.

Description of the algorithm:

- 1. Construct the $n \times m$ matrix $A = [a_{ij}]$ defined by the formula

$$a_{ij} = K(x_i, t_j)\sqrt{w_i^x \cdot w_j^t}. \tag{92}$$

- 2. Compute the singular value decomposition of A, to produce the factorization

$$A = U \cdot S \cdot V^*, \tag{93}$$

where $U = [u_{ij}]$ is an $n \times p$ matrix with orthonormal columns, $V = [v_{ij}]$ is an $m \times p$ matrix with orthonormal columns, and S is a $p \times p$ diagonal matrix whose j'th diagonal entry is s_j.

- 3. Construct the $n \times p$ matrix $\hat{U} = [\hat{u}_{ij}]$ and the $m \times p$ matrix $\hat{V} = [\hat{v}_{ij}]$ defined by the formulae

$$\hat{u}_{ik} = u_{ik}/\sqrt{w_i^x}, \tag{94}$$

$$\hat{v}_{jk} = v_{jk}/\sqrt{w_j^t}. \tag{95}$$

- 4. For any points $x \in [a,b]$ and $t \in [c,d]$, evaluate the functions $u_k : [a,b] \to \mathbf{R}$ and $v_k : [c,d] \to \mathbf{R}$ via the formulae

$$u_k(x) = \sum_{i=1}^{n} \hat{u}_{ik} \cdot \alpha_i(x), \tag{96}$$

$$v_k(t) = \sum_{j=1}^{m} \hat{v}_{jk} \cdot \beta_j(t), \tag{97}$$

for all $k = 1, \ldots, p$.

Theorem 4.3 *Suppose that the combination of the quadrature with weights and nodes $w_1^x, \ldots, w_n^x \in \mathbf{R}$ and $x_1, \ldots, x_n \in [a,b]$, respectively, and the interpolation scheme with Lagrange functions $\alpha_1, \ldots, \alpha_n : [a,b] \to \mathbf{R}$, preserves inner products on $[a,b]$.*

Suppose in addition that the combination of the quadrature with weights and nodes $w_1^t, \ldots, w_m^t \in \mathbf{R}$ and $t_1, \ldots, t_m \in [c,d]$, respectively, and the interpolation scheme with Lagrange functions $\beta_1, \ldots, \beta_m : [c,d] \to \mathbf{R}$, preserves inner products on $[c,d]$.

For any function $K : [a,b] \times [c,d] \to \mathbf{R}$, let $u_i : [a,b] \to \mathbf{R}$, $v_i : [c,d] \to \mathbf{R}$, and $s_i \in \mathbf{R}$ be defined in (92)-(97), for all $i = 1, \ldots, p$. Then

- *1. The functions u_i are orthonormal, i.e.*

$$\int_a^b u_i(x) u_k(x) dx = \delta_{ik} \tag{98}$$

for all $i, k = 1, \ldots, p$, with δ_{ik} the Kronecker symbol ($\delta_{ij} = 1$ if $i = j$, 0 otherwise).

- *2. The functions v_i are orthonormal, i.e.*

$$\int_c^d v_i(t) v_k(t) dx = \delta_{ik} \tag{99}$$

for all $i, k = 1, \ldots, p$.

- *3. The function $\tilde{K} : [a,b] \times [c,d] \to \mathbf{R}$ defined by the formula*

$$\tilde{K}(x,t) = \sum_{j=1}^{p} s_j u_j(x) v_j(t), \tag{100}$$

is identical to the function produced by sampling K on the grid of points (x_i, t_j), then interpolating with the two interpolation schemes. That is,

$$\tilde{K}(x,t) = \sum_{i=1}^{n} \sum_{j=1}^{m} K(x_i, t_j) \alpha_i(x) \beta_j(t). \tag{101}$$

A proof of the above theorem can be found in [9].

4.1.3 Approximation of SVDs of linear operators with finite-dimensional input and infinite-dimensional output

The above algorithm can be modified so that it produces an approximation of a singular value decomposition of a linear operator whose input is finite-dimensional and whose output is infinite-dimensional, or vice versa. (Such singular value decompositions are defined in Section 2.2.2.) Crudely speaking, the modification consists of removing all portions of the algorithm which discretize the kernel of the operator, in the variable in which it is already discrete. More precisely, the modification is as follows:

- 1. The input to the algorithm is the kernel $K : \mathcal{N} \times [c, d] \to \mathbf{R}$ of the linear operator.
- 2. The $n \times m$ matrix $A = [a_{ij}]$ whose singular value decomposition is computed is defined not by (92) but rather by the formula

$$a_{ij} = \sqrt{w_j^t} K(i, t_j). \tag{102}$$

- 3. The output of the algorithm contains, in place of the functions $u_1, \ldots, u_n$, the $m \times p$ matrix $U = [u_{ij}]$ (as produced by the SVD of the matrix A); the columns of this matrix are orthonormal.

4.2 Far field and local expansions

In the terminology of FMM type algorithms, the term "far field expansion" is used to refer to an expansion which represents the potential due to a set of charges, on regions distant from those charges. The term "local expansion" is used to refer to an expansion which represents the potential on a region, due to charges in distant regions. In the original FMM of [3], multipole expansions were used as far field expansions, and Taylor series were used as local expansions. In the FMM described in this paper, both types of expansions are based on singular value decompositions of integral operators. Two types of far field expansions are used, one which is valid only to the right of the interval on which the charges lie, and one which is valid only to the left; likewise two types of local expansions are used, one for charges to the left, and another for charges to the right. One set of expansions is used for the computation of (1), and another is used for the computation of (2).

4.2.1 Expansions for the $1/x$ kernel

For any numbers $a, b, c, d \in \mathbf{R}$ such that $b > 0$ and $d > 0$, we will refer to the integral operator $A_{ab,cd} : L^2[a, a+b] \to L^2[c, c+d]$ given by the formula

$$(A_{ab,cd} \cdot q)(x) = \int_a^{a+b} \frac{q(s)}{x - s} ds, \tag{103}$$

as the $1/x$ potential operator with input on $[a, a+b]$ and output on $[c, c+d]$. This operator possesses the obvious symmetry that the operator $-A^*_{ab,cd}$ is the $1/x$ potential operator with input on $[c, c+d]$ and output on $[a, a+b]$.

Far field and local expansions (for the $1/x$ kernel) on an interval $[a, a+b]$ and for an accuracy ε_A are defined as follows. Let the integral operator $A : L^2[a, a+b] \rightarrow L^2[a+2b, \infty)$ be the $1/x$ potential operator with input on $[a, a+b]$ and output on $[a+2b, \infty)$. As shown in Section 2.2, A can be approximated to any desired accuracy by the truncation of its singular value decomposition. Let the approximation of A to the accuracy ε_A be denoted by $U_A \cdot S_A \cdot V_A^*$, so that

$$\frac{||A - U_A \cdot S_A \cdot V_A^*||}{||A||} < \varepsilon_A. \tag{104}$$

Let the number of singular values used in this approximation be denoted by p. Let the operator $\tilde{V}_A : L^2[a, a+b] \rightarrow \mathbf{R}^p$ be defined by the formula

$$\tilde{V}_A^* \cdot \tilde{q} \;=\; V_A^* \cdot q, \tag{105}$$

where the function $\tilde{q} : [a, a+b] \rightarrow \mathbf{R}$ is given by the formula $\tilde{q}(s) = q(2a + b - s)$.

Then we will refer to the operator V_A (or respectively $\tilde{V}_A$) as the leftgoing (rightgoing) local expansion evaluation operator on $[a, a+b]$, and to its adjoint V_A^* ($\tilde{V}_A^*$) as the rightgoing (leftgoing) far field expansion creation operator on $[a, a+b]$.

4.2.2 A Simple Example

Following is an example of a way in which expansions defined in Section 4.2.1 can be used to calculate sums such as (1). Suppose that the point charges $q_1, \ldots, q_n \in \mathbf{R}$ are located at the points $x_1, \ldots, x_n \in [0, 1]$, respectively, and that the resulting potentials $\phi_1, \ldots, \phi_m$, at the points $y_1, \ldots, y_m \in [2, \infty)$ respectively, are to be calculated. Let the operator $A_{01} : L^2[0, 1] \rightarrow L^2[2, \infty)$ be the $1/x$ potential operator with input on $[0, 1]$ and output on $[2, \infty)$, and let the operator $V_{01}^* : L^2[0, 1] \rightarrow \mathbf{R}^p$ be the rightgoing far field creation operator on $[0, 1]$. V_{01} is then one component of the truncation of the SVD of the operator A_{01} to the rank p; let the remaining components be denoted by $U_{01} : \mathbf{R}^p \rightarrow L^2[2, \infty)$ and $S_{01} : \mathbf{R}^p \rightarrow \mathbf{R}^p$, so that $A_{01} \simeq U_{01} \cdot S_{01} \cdot V_{01}^*$. Let the charge distribution $q : [0, 1] \rightarrow \mathbf{R}$ be defined by the formula

$$q(x) = \sum_{i=1}^{n} q_i \delta(x - x_i), \tag{106}$$

where δ denotes the Dirac δ-function. Then using (103), we get

$$\begin{aligned} (A_{01} \cdot q)(x) \;&=\; \int_0^1 \left(\sum_{i=1}^{n} q_i \delta(s - x_i) \right) \frac{1}{x - s} ds, \\ &=\; \sum_{i=1}^{n} \frac{q_i}{x - x_i}; \end{aligned} \tag{107}$$

that is to say, the product $A_{01} \cdot q$ yields the potential on the interval $[2, \infty)$ which is due to the charges $q_1, \ldots, q_n$. Thus $\phi_j = (A_{01} \cdot q)(y_j) \simeq (U_{01} \cdot S_{01} \cdot V_{01}^* \cdot q)(y_j)$, for all $j = 1, \ldots, m$. This calculation can be done in two steps, the first being the computation $\alpha = V_{01}^* \cdot q$, and the second being the computation $\phi_j = (U_{01} \cdot S_{01} \cdot \alpha)(y_j)$, for all $j = 1, \ldots, m$. Let the left and right singular functions of A_{01} be denoted by $u_1, \ldots, u_p : [2, \infty) \rightarrow \mathbf{R}$ and $v_1, \ldots, v_p : [0, 1] \rightarrow \mathbf{R}$,

respectively, and let the singular values of A_{01} be denoted by $s_1, \ldots, s_p \in \mathbf{R}$. Let M denote the CPU time required to evaluate any of the singular functions at a point. Then the calculation $\alpha = V_{01} \cdot q$ amounts to the computation of the quantities $\alpha_1, \ldots, \alpha_p \in \mathbf{R}$ which are given by the formula

$$\begin{aligned} \alpha_k &= \int_0^1 v_k(s) q(s) ds \\ &= \int_0^1 v_k(s) (\sum_{i=1}^{n} \delta(s - x_i)) ds \\ &= \sum_{i=1}^{n} v_k(x_i) q_i, \end{aligned} \tag{108}$$

and can be done in $O(npM)$ time. The calculation $\phi_j = (U_{01} \cdot S_{01} \cdot \alpha)(y_j)$ amounts to the computation of the quantities $\phi_1, \ldots, \phi_m \in \mathbf{R}$ given by the formula

$$\phi_j = \sum_{k=1}^{p} \alpha_k s_k u_k(y_i), \tag{109}$$

and can be done in $O(mpM)$ time, for a total of $O((m+n)pM)$ time, as compared to $O(mn)$ time for the direct calculation.

4.2.3 Expansions for the logarithmic kernel

For any numbers $a, b, c, d \in \mathbf{R}$ such that $b > 0$ and $d > 0$, we will refer to the integral operator $B_{ab,cd} : L^2[a, a+b] \to L^2[c, c+d]$ which is given by the formula

$$(B_{ab,cd} q)(x) = \int_a^{a+b} q(s) log|x - s| ds, \tag{110}$$

as the logarithmic potential operator with input on $[a, a+b]$ and output on $[c, c+d]$. This operator possesses the obvious symmetry that the operator $B^*_{ab,cd}$ is the logarithmic potential operator with input on $[c, c+d]$ and output on $[a, a+b]$.

Far field and local expansions for the logarithmic kernel on an interval $[a, a+b]$ for an accuracy ε_A are defined as follows. Let the operator $B : L^2[a, a+b] \to C[a+2b, \infty)$ be the logarithmic potential operator with input on $[a, a+b]$ and output on $[a+2b, \infty)$. Let the operator $C : L^2[a, a+b] \to L^2[a+2b, \infty)$ be defined by the formula

$$C = B(I - u \cdot u^*), \tag{111}$$

where the operator $u : \mathbf{R} \to L^2[a, a+b]$ is given by the formula $(uy)(x) = y/\sqrt{b}$. Clearly, C is the operator which orthogonalizes its input to the constant function (producing, in physical terms, a charge distribution with no net charge), then computes the resulting potential. We will refer to C as the modified logarithmic potential operator on $[a, a+b]$. Let the approximation of C to within the accuracy ε_A by the truncation of its singular value decomposition be denoted by $U_C \cdot S_C \cdot V_C^*$, so that

$$\frac{||C - U_C \cdot S_C \cdot V_C^*||}{||C||} < \varepsilon_A, \tag{112}$$

and let the number of singular values therein be denoted by p_c. Let the operator $V_B^* : [a, a+b] \rightarrow \mathbf{R}^{p_c+1}$ be defined by the formula

$$(V_B^* \cdot q)^* \cdot e_i = \begin{cases} \sqrt{b}(V_C^* \cdot q)^* \cdot e_i, & i = 1, \ldots, p_c, \\ \sqrt{b}(u^* \cdot q), & i = p_c + 1, \end{cases} \tag{113}$$

where $e_1, \ldots, e_{p_c+1} \in \mathbf{R}^{p_c+1}$ are the standard basis vectors in $\mathbf{R}^{p_c+1}$. Clearly, V_B is the operator which computes V_C^*, and which in addition computes the integral of its input function (in physical terms, the net charge on the interval). Let the operator $\tilde{V}_B : L^2[a, a+b] \rightarrow \mathbf{R}^p$ be defined by the formula

$$(\tilde{V}_B \cdot q) \;=\; V_B \cdot \tilde{q}, \tag{114}$$

where $\tilde{q} : [a, a+b] \rightarrow \mathbf{R}$ is given by the formula $\tilde{q}(s) = q(2a + b - s)$.

We will refer to the operator V_B (or respectively $\tilde{V}_B$) as the leftgoing (rightgoing) local expansion evaluation operator for the logarithmic potential on $[a, a+b]$, and to its adjoint V_B^* ($\tilde{V}_B^*$) as the rightgoing (leftgoing) far field expansion creation operator on $[a, a+b]$.

Lemma 4.4 *Let the operator V_B be defined by (113). Then*

$$\frac{1}{b} V_B^* \cdot V_B = I. \tag{115}$$

Proof. Using (113), we get

$$\begin{aligned} \frac{1}{b} V_B^* \cdot V_B &= \begin{pmatrix} V_C^* \cdot V_C & V_C^* \cdot u \\ u^* \cdot V_C & u^* \cdot u \end{pmatrix} \\ &= \begin{pmatrix} I & V_C^* \cdot u \\ u^* \cdot V_C & 1 \end{pmatrix}. \end{aligned} \tag{116}$$

Suppose that $V_C^* \cdot u \neq 0$. Then since (by definition) none of the singular values in S_C are zero, $S_C \cdot V_C^* \cdot u \neq 0$, and since U_C is orthogonal, $U_C \cdot S_C \cdot V_C^* \cdot u \neq 0$, so $C \cdot u \neq 0$, so $B(I - u \cdot u^*)u \neq 0$. But $B(I - u \cdot u^*)u = B(u - u \cdot u^* \cdot u) = B(u - u) = 0$. Thus, by contradiction,

$$V_C^* \cdot u = 0. \tag{117}$$

Now, the lemma follows immediately from the combination of (117) and (116). □

4.2.4 Scaling behavior

As the following two theorems establish, the operators on which far field and local expansions are based scale extremely simply with the size of the interval $[a, a+b]$. The first theorem is for the $1/x$ kernel; its proof is trivial, and is omitted. The second theorem is for the logarithmic kernel; its proof is slightly more involved, and is presented.

Theorem 4.5 *For any numbers $a, b \in \mathbf{R}$ with $b > 0$, let the operators A_{01} and A_{ab} be the $1/x$ potential operators (see (103)) with inputs on $[0,1]$ and $[a, a+b]$ respectively, and with outputs on $[2,\infty)$ and $[a+2b,\infty)$ respectively. Then for any function $q : [0,1] \to \mathbf{R}$, and for any point $x \in [2,\infty)$,*

$$(A_{ab} \cdot q_{ab})(bx + a) = (A_{01} \cdot q)(x), \tag{118}$$

where the function $q_{ab} : [a, a+b] \to \mathbf{R}$ is defined by the equation $q_{ab}(bx + a) = q(x)$.

Theorem 4.6 *For any numbers $a, b \in \mathbf{R}$ with $b > 0$, let the operators C_{01} and C_{ab} be the modified logarithmic potential operators (see (111) on $[0,1]$ and $[a, a+b]$ respectively. Then for any function $q : [0,1] \to \mathbf{R}$, and for any point $x \in [2,\infty)$,*

$$(C_{ab} \cdot q_{ab})(bx + a) = b(C_{01} \cdot q)(x), \tag{119}$$

where the function $q_{ab} : [a, a+b] \to \mathbf{R}$ is defined by the equation $q_{ab}(bx + a) = q(x)$.

Proof. For notational convenience, we first define rescalings of the operators B_{ab} and C_{ab}. Let the operators $\hat{B}_{ab} : L^2[0,1] \to C[2,\infty)$ and $\hat{C}_{ab} : L^2[0,1] \to L^2[2,\infty)$ be defined by the formulae

$$\begin{aligned} (\hat{B}_{ab} \cdot q)(x) &= (B_{ab} \cdot q_{ab})(bx + a), & (120) \\ (\hat{C}_{ab} \cdot q)(x) &= (C_{ab} \cdot q_{ab})(bx + a). & (121) \end{aligned}$$

Let the operators $u_{01} : \mathbf{R} \to L^2[0,1]$ and $u_{ab} : \mathbf{R} \to L^2[a, a+b]$ be defined by the formulae $(u_{01} \cdot y)(x) = y$ and $(u_{ab} \cdot y)(x) = y/\sqrt{b}$. Combining (121) and (111), we obtain

$$\begin{aligned} (\hat{C}_{ab} \cdot q)(x) &= (B_{ab}(I - u_{ab} \cdot u_{ab}^*)q_{ab})(bx + a) \\ &= (B_{ab} \cdot q_{ab})(bx + a) - (B_{ab} \cdot u_{ab} \cdot u_{ab}^* \cdot q_{ab})(bx + a) \\ &= (\hat{B}_{ab} \cdot q)(x) - ((\hat{B}_{ab} \cdot u_{01}/\sqrt{b})u_{ab}^* \cdot q_{ab})(x) \\ &= (\hat{B}_{ab} \cdot q)(x) - (\hat{B}_{ab} \cdot u_{01} \cdot u_{01}^* \cdot q)(x) \\ &= (\hat{B}_{ab}(I - u_{01} \cdot u_{01}^*)q)(x). \end{aligned} \tag{122}$$

The combination of (120) and (110) yields

$$(\hat{B}_{ab}q)(x) = \int_a^{a+b} q_{ab}(s) \log|(bx + a) - s| ds. \tag{123}$$

Performing the change of variables $s = bt + a$ in (123), we get

$$\begin{aligned} (\hat{B}_{ab}q)(x) &= \int_0^1 q_{ab}(bt + a) \log|(bx + a) - (bt + a)| b \, dt \\ &= b \int_0^1 q(t)(\log|x - t| + \log b) dt \\ &= b \left(\int_0^1 q(t) \log|x - t| dt + \int_0^1 q(t) \log b \, dt \right) \\ &= b((B_{01} \cdot q)(x) + (\log b)u_{01}^* \cdot q), \end{aligned} \tag{124}$$

with $B_{01} : [0,1] \to C[2,\infty]$ the logarithmic potential operator with input on $[0,1]$ and output on $[2,\infty]$. Let the operator $u_2 : \mathbf{R} \to C[2,\infty]$ be defined by the formula $(u_2 \cdot y)(x) = y$. Then (124) can be written as:

$$\hat{B}_{ab} = b(B_{01} + (\log b)u_2 \cdot u_{01}^*). \tag{125}$$

Thus

$$\begin{aligned}
\hat{C}_{ab} &= \hat{B}_{ab}(I - u_{01} \cdot u_{01}^*) \\
&= b(B_{01} + (\log b)u_2 \cdot u_{01}^*)(I - u_{01} \cdot u_{01}^*) \\
&= bB_{01}(I - u_{01} \cdot u_{01}^*) + b(\log b)u_2 \cdot u_{01}^*(I - u_{01} \cdot u_{01}^*) \\
&= (bB_{01})(I - u_{01} \cdot u_{01}^*) + b(\log b)u_2 \cdot u_{01}^*(I - u_{01} \cdot u_{01}^*) \\
&= bC_{01} + b(\log b)u_2 \cdot u_{01}^*(I - u_{01} \cdot u_{01}^*) \\
&= bC_{01} + b(\log b)u_2(u_{01}^* - u_{01}^* \cdot u_{01} \cdot u_{01}^*) \\
&= bC_{01}.
\end{aligned} \tag{126}$$

Now, (119) follows immediately from (126) and (121). □

4.3 Exponential expansions

This section defines operators which create and evaluate exponential expansions. It defines operators both for leftgoing and for rightgoing expansions. The expansions created by these operators differ slightly from those defined by Theorems 3.1 and 3.2, in that the multiplication by the quadrature weights w_j, which in those theorems is done in the evaluation of the expansion (that is, in (52) or (67)), is, for symmetry, here done partly in the evaluation and partly in the creation of the expansion, each containing multiplications by square roots of the weights. A consequence is that each evaluation operator is the adjoint of a creation operator for expansions going in the other direction.

Two variants of the operators are defined. The first variant consists of operators which ignore a section of size r of the interval $[a, a+b]$, either at the beginning of the interval or at the end. The second variant consists of operators which compute, in addition to the exponential expansion, a multipole expansion on the interval, or, in the case of evaluation operators, which evaluate, in addition to the exponential expansion, a Taylor series.

All the operators are based on rescalings of the same quadrature; for the operators on an interval $[a, a+b]$, the quadrature is rescaled (using (61),(62)) so that the maximum range of the exponential expansions is equal to $2b$. Let the number r denote the resulting minimum range, and let the weights and nodes of the rescaled quadrature be denoted by $w_1, \ldots, w_m \in \mathbf{R}$ and $t_1, \ldots, t_m \in \mathbf{R}$ respectively.

Let the operators $X : L^2[a, a+b] \to \mathbf{R}^m$, $Y : L^2[a, a+b] \to \mathbf{R}^{m+1}$, $X_g : L^2[a, a+b] \to \mathbf{R}^{m+k}$, and $Y_g : L^2[a, a+b] \to \mathbf{R}^{m+k}$ be defined by the formulae

$$(X \cdot q)^* \cdot e_j = \int_a^{a+b} \sqrt{w_j} q(s) e^{(s-(a+b))t_j} ds, \tag{127}$$

$$(Y \cdot q)^* \cdot e_j = \begin{cases} \int_a^{a+b} \sqrt{w_j/t_j} q(s) e^{(s-(a+b))t_j} ds, & j = 1, \ldots, m \\ \int_a^{a+b} q(s) ds, & j = m+1, \end{cases} \tag{128}$$

$$(X_g \cdot q)^* \cdot e_j = \begin{cases} (X \cdot q)^* \cdot e_j, & j = 1, \ldots, m \\ \int_a^{a+b} q(s) P_{j-m-1}(s)/\sqrt{b}\ ds, & j = m+1, \ldots, m+k, \end{cases} \tag{129}$$

$$(Y_g \cdot q)^* \cdot e_j = \begin{cases} (Y \cdot q)^* \cdot e_j, & j = 1, \ldots, m+1 \\ \int_a^{a+b} q(s) P_{j-m-1}(s)\ ds, & j = m+2, \ldots, m+k, \end{cases} \tag{130}$$

where $e_1, \ldots, e_m \in \mathbf{R}^m$ are the standard basis vectors in $\mathbf{R}^m$, and where the polynomial P_j is defined by the formula $P_j(s) = ((s - a - b/2)/b)^j$. Let the operators $\tilde{X} : L^2[a, a+b] \to \mathbf{R}^m$, $\tilde{X}_g : L^2[a, b] \to \mathbf{R}^{m+k}$, $X_r : L^2[a, a+b-r] \to \mathbf{R}^m$, $\tilde{X}_r : L^2[a+r, a+b] \to \mathbf{R}^m$, $\tilde{Y} : L^2[a, a+b] \to \mathbf{R}^{m+1}$, $\tilde{Y}_g : L^2[a, b] \to \mathbf{R}^{m+k}$, $Y_r : L^2[a, a+b-r] \to \mathbf{R}^{m+1}$, and $\tilde{Y}_r : L^2[a+r, a+b] \to \mathbf{R}^{m+1}$ be the operators such that for any function $q : [a, a+b] \to \mathbf{R}$,

$$\tilde{X} \cdot \tilde{q} = X \cdot q, \tag{131}$$

$$\tilde{Y} \cdot \tilde{q} = Y \cdot q, \tag{132}$$

$$\tilde{X}_g \cdot \tilde{q} = X_g \cdot q, \tag{133}$$

$$\tilde{Y}_g \cdot \tilde{q} = Y_g \cdot q, \tag{134}$$

$$X_r \cdot q = X \cdot q_1, \tag{135}$$

$$Y_r \cdot q = Y \cdot q_1, \tag{136}$$

$$\tilde{X}_r \cdot \tilde{q} = X \cdot q_2, \tag{137}$$

$$\tilde{Y}_r \cdot \tilde{q} = Y \cdot q_2, \tag{138}$$

where the functions $\tilde{q}, q_1, q_2 : [a, a+b] \to \mathbf{R}$ are defined by the equations

$$\tilde{q}(x) = q(2a + b - x), \tag{139}$$

$$q_1(x) = \begin{cases} q(x), & a \le x \le a+b-r \\ 0, & a+b-r \le x \le a+b, \end{cases} \tag{140}$$

$$q_2(x) = \begin{cases} 0, & a \le x \le a+r \\ q(x), & a+r \le x \le a+b, \end{cases} \tag{141}$$

The terminology which will be used for the above operators and their adjoints is as follows. The operators X, X_g, X_r, $\tilde{X}$, $\tilde{X}_g$, $\tilde{X}_r$, and their adjoints, will be referred to as operators for the $1/x$ kernel. The operators Y, Y_g, Y_r, $\tilde{Y}$, $\tilde{Y}_g$, $\tilde{Y}_r$, and their adjoints will be referred to as operators for the logarithmic kernel. The operators X, Y, X_g, Y_g, X_r, and Y_r, will be referred to as rightgoing exponential expansion creation operators. Their adjoints will be referred to as leftgoing exponential expansion evaluation operators. The operators $\tilde{X}$, $\tilde{Y}$, $\tilde{X}_g$, $\tilde{Y}_g$, $\tilde{X}_r$, and $\tilde{Y}_r$, will be referred to as leftgoing exponential expansion creation operators. Their adjoints will be referred to as rightgoing exponential expansion evaluation operators. The operators X_r, Y_r, $\tilde{X}_r$, $\tilde{Y}_r$, and their adjoints, will be referred to as "restricted". The operators X_g, Y_g, $\tilde{X}_g$, $\tilde{Y}_g$,

Table 1: Operators used in the construction of translation and conversion matrices

Operator	Definition
$V_{02}^* : L^2[a, a+2b] \to \mathbf{R}^p$	rightgoing far field expansion creation operators on the intervals $[a, a+2b]$, $[a, a+b]$, and $[a+b, a+2b]$ respectively (see (104)).
$V_{01}^* : L^2[a, a+b] \to \mathbf{R}^p$	
$V_{10}^* : L^2[a+b, a+2b] \to \mathbf{R}^p$	
$V_{02a}^* : L^2[a, a+b] \to \mathbf{R}^p$	identical to V_{02}^*, but with inputs restricted to the intervals $[a, a+b]$ and $[a+b, a+2b]$ respectively.
$V_{02b}^* : L^2[a+b, a+2b] \to \mathbf{R}^p$	
$\tilde{V}_{34} : \mathbf{R}^p \to L^2[a+3b, a+4b]$	rightgoing local expansion evaluation operator on $[a+3b, a+4b]$. (see (105))
$A_{03} : L^2[a, a+b] \to L^2[a+3b, a+4b]$	$1/x$ potential operators on the specified intervals (see (103))
$A_{13} : L^2[a+b, a+2b] \to L^2[a+3b, a+4b]$	
$A_{01r} : L^2[a, a+b] \to L^2[a+b+r, a+3b]$	
$A_{02} : L^2[a, a+b] \to L^2[a+2b, a+4b]$	
$X_g : L^2[a, a+b] \to \mathbf{R}^{m+k}$	k-term augmented exponential expansion creation operator on $[a, a+b]$ (see (129))
$U_{X_g} \cdot S_{X_g} \cdot V_{X_g}^* = X_g$	singular value decomposition of X_g
$\tilde{X}_r^* : \mathbf{R}^m \to L^2[a+b+r, a+3b]$	restricted exponential expansion evaluation operator on $[a+b, a+3b]$ (see (137)).
$U_{X_r} \cdot S_{X_r} \cdot V_{X_r}^* = \tilde{X}_r$	singular value decomposition of $\tilde{X}_r$
$\tilde{X}^* : \mathbf{R}^m \to L^2[a+2b, a+4b]$	exponential expansion evaluation operator on $[a+2b, a+4b]$ (see (131)).
$U_X \cdot S_X \cdot V_X^* = \tilde{X}$	singular value decomposition of $\tilde{X}$

and their adjoints will be referred to as "augmented". (Thus, for example, $\tilde{Y}_g^*$ will be referred to as the augmented rightgoing exponential expansion evaluation operator for the logarithmic kernel.)

4.4 Translation and conversion matrices

This section lists (in Table 2) translation and conversion matrices used by the algorithm. These matrices are derived along the lines mentioned in Remarks 2.1 and 2.2: each matrix is chosen to minimize the norm of an appropriate linear operator, and the formula for the matrix is then a consequence of Lemma 2.7. Table 2 lists, for each matrix, the norm to be minimized and the resulting formula. Only matrices for the $1/x$ kernel are listed; the formulae for the matrices for the logarithmic kernel are identical, except in that they use the corresponding operators for

Table 2: Translation and conversion matrices

Matrix	Input Expansion	Output Expansion	Quantity to minimize	Formula
M_{ff1}	far field on $[a, a+b]$	far field on $[a, a+2b]$	$\|\|V^*_{02a} - M_{ff1} \cdot V^*_{01}\|\|$	$M_{ff1} = V^*_{02a} \cdot V_{01}$
M_{ff2}	far field on $[a+b, a+2b]$	far field on $[a, a+2b]$	$\|\|V^*_{02b} - M_{ff2} \cdot V^*_{12}\|\|$	$M_{ff2} = V^*_{02b} \cdot V_{12}$
M_{ef}	augmented exponential on $[a, a+b]$	far field on $[a, a+b]$	$\|\|V^*_{01} - M_{ef} \cdot X_g\|\|$	$M_{ef} = V^*_{01} \cdot V_{X_g} \cdot S^{-1}_{X_g} \cdot U^*_{X_g}$
M_{fl1}	far field on $[a, a+b]$	local on $[a+3b, a+4b]$	$\|\|A_{03} - \tilde{V}_{34} \cdot M_{fl1} \cdot V^*_{01}\|\|$	$M_{fl1} = \tilde{V}^*_{34} \cdot A_{03} \cdot V_{01}$
M_{fl2}	far field on $[a+b, a+2b]$	local on $[a+3b, a+4b]$	$\|\|A_{13} - \tilde{V}_{34} \cdot M_{fl2} \cdot V^*_{12}\|\|$	$M_{fl2} = \tilde{V}^*_{34} \cdot A_{13} \cdot V_{12}$
M_{ee}	augmented exponential on $[a, a+b]$	restricted exponential on $[a+b, a+3b]$	$\|\|A_{01r} - \tilde{X}^*_r \cdot M_{ee} \cdot X_g\|\|$	$M_{ee} = V_{X_r} \cdot S^{-1}_{X_r} \cdot U^*_{X_r} \cdot A_{01r} \cdot V_g \cdot S^{-1}_g \cdot U_g$
M_{de}	far field on $[a, a+b]$	exponential on $[a+2b, a+4b]$	$\|\|A_{02} - \tilde{X}^* \cdot M_{de} \cdot V^*_{01}\|\|$	$M_{de} = V_X \cdot S^{-1}_X \cdot U^*_X \cdot A_{02} \cdot V_{01}$

Table 3: Uses of adjoints of translation and conversion matrices

Matrix	Input Expansion	Output Expansion
M^*_{ff1}	local on $[a, a+2b]$	local on $[a, a+b]$
M^*_{ff2}	local on $[a, a+2b]$	local on $[a+b, a+2b]$
M^*_{ef}	local on $[a, a+b]$	augmented exponential on $[a, a+b]$
M^*_{ee}	restricted exponential on $[a+b, a+3b]$	augmented exponential on $[a, a+b]$
M^*_{de}	restricted exponential on $[a+b, a+3b]$	augmented exponential on $[a, a+b]$

that kernel. Only matrices for rightgoing expansions are listed; by symmetry, the matrices for leftgoing expansions are identical. The adjoints of all but two of the matrices in Table 2 also are used by the algorithm; these uses are listed in Table 3.

The operators used in Table 2 are listed in Table 1. Each operator's input and/or output lies on an interval which bears a fixed relation to a reference interval $[a, a+b]$. Each of the translation and conversion matrices is thus a function of a and b. A rather tedious analysis, which is omitted here, and which would be based largely on Theorems 4.5 and 4.6, shows that for the $1/x$ kernel, none of the matrices varies with a or b, and that for the logarithmic kernel, none of the matrices vary with a or b, except that the last four matrices in Table 2 each contain one element which varies with b. This element (the (p_c+1, p_c+1)'th element of M_{fl1} and M_{fl2}, the $(m+1, p_c+1)$'th element of M_{de}, and the $(m+1, m+1)$'th element of M_{ee}, where $m+1$ is the number of terms in an exponential expansion for the logarithmic kernel, and p_c+1 is the number of terms in a far field expansion for the logarithmic kernel) is the element which is the coefficient, on the input side, of the integral of the charge, and on the output side, of the constant potential, and, in all cases, varies as follows: if b is multiplied by a number f, then the quantity $(\log f)$ is added to that element of the matrix.

Bounds for the L^2 error of these matrices are given by Lemma 2.9, for the first three matrices, and by Lemma 2.8 for the remaining four matrices (see also Remarks 2.3 and 2.4.) In most cases the conditions of the lemmas are trivially satisfied, with the numbers ε_1 and ε_2 of the lemmas being the L^2 errors of the expansions between which the matrices convert. The only exception is in the case of the matrix M_{ef}, which converts an augmented exponential expansion on an interval $[a, a+b]$ into a far field expansion on the same interval. In this case, Lemma 2.9 requires that the augmented exponential expansion be sufficient to accurately compute the potential on the interval $[a+2b, \infty)$ (on which the far field expansion is to be used) which is due to a set of charges on $[a, a+b]$. As is shown in [2], the multipole expansion which comprises the augmentation to the exponential expansion is sufficient by itself for computing the potential on $[a+2b, \infty)$; any desired accuracy can be achieved by taking a sufficiently large number k of terms. Thus for some k, this condition is satisfied.

Remark 4.1 *The authors know of no analytic formula for the minimum value of k which is necessary to achieve a given precision. Thus, in the authors' implementation, the necessary value of k was determined by numerical experimentation, using the formula (25) to compute the error for each k.)*

5 Simple Exponential Expansion FMM Algorithm

A simple algorithm for (1), which is suitable for moderate numbers of nearly equispaced points, is as follows. The input to the algorithm is a sequence of points in sorted order $x_1 < x_2 < \ldots < x_n$ and a sequence of charges $q_1, q_2, \ldots, q_n \in \mathbf{R}$. The algorithm consists of two passes. The first pass computes the portion of the potential at each point which is due to the charges to the left of it, in other words, the sums

$$\phi_j = \sum_{i=1}^{j-1} \frac{q_i}{x_j - x_i}. \tag{142}$$

The second pass computes the portion of the potential at each point which is due to charges to the right of it; this pass is almost identical to the first pass, and thus will not be discussed further.

The algorithm uses a quadrature which satisfies the conditions of Theorem 3.1. The first step of the algorithm is to rescale the quadrature (using (61),(62) so that the maximum range of the exponential expansions is equal to $x_n - x_1$. Let the number r denote the minimum range of the expansions based on this rescaled quadrature, and let its weights and nodes be denoted by $w_1, \ldots, w_m \in \mathbf{R}$ and $t_1, \ldots, t_m \in \mathbf{R}$ respectively.

The algorithm makes use of the observation that evaluation of an exponential expansion is less computationally expensive when the expansion is already located at the point at which it is to be evaluated, since the factor $e^{(y-x_i)t_j}$ in (52) is equal to one when $y = x_i$. For the same reason, the creation of an exponential expansion for a single charge q_i at a point x_i is, less computationally expensive when the expansion is located at that point. Thus the algorithm maintains a single exponential expansion, which it translates to each point x_i in turn, moving from left to right. This expansion represents the potential due to the charges to the left of the current point, and is accurate on the region to the right of the current point.

Algorithm 5.1

Comment { Make an exponential expansion for the charge at the first point}
do $j = 1, \ldots, m$ { $\alpha_j := q_1$ }

do $i = 2, \ldots, n$ {
    Comment { translate the expansion from point $i-1$ to point i }
    do $j = 1, \ldots, m$ { $\alpha_j := \alpha_j e^{(x_{i-1}-x_i)t_j}$ }

    Comment { evaluate the potential at point i }
    $\phi_i := \sum_{j=1}^{m} w_j \alpha_j$

    Comment { add charge i to the expansion }
    do $j = 1, \ldots, m$ { $\alpha_j := \alpha_j + q_i$ }
}

The execution time for Algorithm 5.1 is clearly $O(mn)$. One of the most computationally expensive portions of the algorithm is the evaluation of exponentials for use in the translation operator. These exponentials do not depend on the sequence $\{q_i\}$, but rather only on the sequence $\{x_i\}$. Thus, if (1) is to be computed for many sequences $\{q_i\}$ and a single sequence $\{x_i\}$, it may be most efficient to precompute these exponentials.

The error which Algorithm 5.1 makes in the portion of the potential at a point x_i which is due to a charge q_k is given by the obvious formula $q_k c_{ik}$, where

$$c_{ik} = \left(\frac{1}{x_i - x_k} - \sum_{j=1}^{m} w_j e^{(x_i - x_k)t_j} \right). \tag{143}$$

By Theorem 3.1, if $|x_k - x_i| \geq r$, then $c_{ik} < \varepsilon$. The remaining interactions, between pairs of points which are too close together, can obviously be corrected by performing the operation

$$\phi_k := \phi_k + c_{ik} q_i, \tag{144}$$

for each of the p pairs of points (x_i, x_k) such that $|x_k - x_i| < r$, after the completion of Algorithm 5.1. The CPU time required for this process is clearly $O(p)$; in addition, the computation of the coefficients c_{ik} takes $O(mp)$ CPU time, but only needs to be performed once for any given sequence of points $\{x_i\}$. For large numbers of points n, p varies as n^2; thus, while Algorithm 5.1 combined with this process of "local corrections" produces accurate results for any n and any distribution of points, and has $O(n)$ behavior for small numbers of points which are close to uniformly distributed, it is asymptotically an $O(n^2)$ algorithm.

The quadratures used in the authors' implementation were generalized Gaussian quadratures [9] specifically tailored to this problem. In particular, quadratures were generated for integrals of the form (51), with the interval of accuracy being $[1, 500]$.

5.1 Logarithmic kernel

The algorithm for the logarithmic kernel (2) is nearly identical. The required changes to the algorithm follow from the differences between Theorem 3.1 and Theorem 3.2, and are as follows:

- 1. In place of the quadrature weights w_j, the numbers $(-w_j/t_j)$ are used.
- 2. The numbers $p_1, \ldots, p_{n-1}$ defined by (69) are computed in $O(n)$ time by taking a running sum.

5.2 Square root kernel

The algorithm for the square root kernel (77) is again nearly identical. The required changes follow from the differences between Theorem 3.1 and Theorem 3.3, and are as follows:

- 1. The input points are further restricted as follows: the points x_i must lie on the interval $[0, b]$, where b is determined by the quadrature.
- 2. The generation of the exponential expansion for each point is done using (81) rather than (53); thus $O(nm)$ evaluations of the function I_0 are required.
- 3. A different quadrature is required. The authors' implementation of this algorithm used generalized Gaussian quadratures [9] tailored to the integral (78) under the condition that $x \in [1, 500]$ and $y \in [0, x - 1]$. These quadratures are similar to those used for the $1/x$ kernel, both in the number of nodes required to achieve a given accuracy and in the values of the weights and nodes. They are tabulated in [9] for various precisions, and are partially repeated in this paper as Tables 16 and 17.

6 Adaptive FMM in One Dimension

This section describes an algorithm for the computation of (1) or (2) which uses an adaptive partitioning of the interval on which the points $\{x_i\}$ lie, and which consumes $O(n)$ CPU time for arbitrary point distributions.

The algorithm uses the translation and conversion matrices listed in Table 2. Due to the simple manner in which these matrices scale to different size intervals, their computation only

needs to be performed once (ever) for any given precision of the algorithm. In this computation, the singular value decompositions of linear operators used in the formulae for these matrices can be computed by the algorithms of Sections 4.1.2 and 4.1.3, with the combinations of quadrature and interpolation scheme used for those algorithms being Gaussian quadratures at Legendre nodes, and polynomial interpolation.

The algorithm partitions the interval $[x_1, x_n]$ on which the points lie in the following manner. (The points are assumed to be in sorted order.) Each subinterval produced is examined to determine the number of local corrections which would be used on that interval if it were left intact; this is equal to the number of pairs of points which are closer together than a threshold which depends on the quadrature used for exponential expansions, and which is proportional to the size of the interval. If this exceeds a fixed number n_c, the interval is split; the split is always exactly in half. In addition, the interval is split if the adjacent interval on either side is less than half its size. Otherwise, the interval is left intact. (The number n_c is chosen by numerical experiment, so as to roughly minimize the CPU time consumed by the algorithm; the tradeoff it controls is between CPU time necessary to perform multiplications of vectors by translation and conversion matrices, and CPU time necessary to process local corrections.)

The main computation is done in two passes. The first pass computes potentials due to charges to the left of each point (142), and the second pass computes potentials due to charges to the right. These two computations are essentially identical, and thus only the first pass will be described. Only rightgoing expansions are used in this pass; thus the word "rightgoing" will be omitted throughout the description. We will refer to an exponential expansion created from charges on an interval as an outgoing expansion on that interval, and to an exponential expansion from which the potential on some interval can be calculated as an incoming expansion on that interval.

The program maintains the following variables. Each interval is assigned a number, which is used to index two arrays: the array γ_i contains the far field expansion for interval i, and the array β_i contains the local expansion for interval i. The latter array is initialized to zero. In addition, six exponential expansion variables are used, three for incoming expansions (α^{in}, α^{in2}, and α^{in3}), and three for outgoing expansions (α^{out}, α^{out2}, and α^{out3}). We will designate the j'th element of each of these expansions by appending the subscript j.

The first pass of the algorithm is done by the following recursive subroutine. The routine takes one parameter, the number of the interval on which to operate; to do the entire first pass, the routine is called on the top-level interval. The routine traverses the tree of subintervals in such a manner that the lowest-level subintervals are visited in order from left to right. The subroutines which it calls are presented subsequently.

Algorithm 6.1

```
subroutine process ( i )

if ( interval i is subdivided ) {
        i1 := interval number of left half of i
        i2 := interval number of right half of i
```

Comment { Use the local expansion for the interval i to produce local expansions for its two halves }
$\beta_{i_1} := M^*_{ff1} \cdot \beta_i$
$\beta_{i_2} := M^*_{ff2} \cdot \beta_i$

Comment { Process the left half }
call process (i_1)

Comment { Compute interactions from the left half to the right half }
call interact (i)

Comment { Process the right half }
call process (i_2)

Comment { Use the far field expansions on the two halves to produce far field expansions on the interval i. }
$\gamma_i := M_{ff1} \cdot \gamma_{i_1} + M_{ff2} \cdot \gamma_{i_2}$

} else {

Comment { Transform the local expansion for this interval into an augmented incoming exponential expansion, and add it to the already-produced incoming expansion resulting from the adjacent segment to the left. }
$\alpha^{\text{in}} := \alpha^{\text{in}} + M^*_{ef} \cdot \beta_i$

Comment { Perform the evaluation of potentials and creation of outgoing exponential expansion(s) on this interval }
call lowlev(i)

Comment { Transform the outgoing augmented exponential expansion into the far field expansion for this interval }
$\gamma_i := M_{ef} \cdot \alpha^{\text{out}}$

}

The routine *lowlev* is a version of Algorithm 5.1, which it differs from in the following ways.

- 1. It takes as a parameter the interval number i of the interval $[a, a+b]$ to process. The quadrature used in exponential expansions is rescaled (using (61),(62)) so that the maximum range of the resulting exponential expansions is equal to $2b$; we will denote the corresponding minimum range by r.

- 2. The incoming augmented exponential expansion α^{in} on $[a, a+b]$ is evaluated at each point. The Taylor series portion of the expansion is evaluated directly; the exponential expansion portion is computed by initializing the exponential expansion α to equal α^{in}, rather than initializing it to zero.

- 3. The outgoing augmented exponential expansion α^{out} on $[a, a+b]$ is computed from the input charges on $[a, a+b]$. The multipole expansion portion is computed directly; the

exponential expansion portion is computed from the value of the exponential expansion α at the end of the routine (after all the charges on the interval have been added to it).

- 4. If the adjacent interval to the left is smaller than the current interval, two additional incoming exponential expansions are evaluated. The first expansion, $\alpha^{\text{in}3}$, is the restricted incoming exponential expansion on $[a, a+b]$, and is evaluated by adding it to the expansion α when that expansion has just been translated past the point $a+r$. The second expansion, $\alpha^{\text{in}2}$, uses a quadrature scaled appropriately for the interval to the left, and is only evaluated on the tiny interval $[a, a+r)$.

- 5. If the adjacent interval to the right is smaller than the current interval, two additional outgoing expansions are created. The first expansion, $\alpha^{\text{out}3}$, is the restricted outgoing exponential expansion on $[a, a+b]$, and is computed from the expansion α when that expansion is just about to be translated past the point $a+b-r$. The second additional outgoing expansion, $\alpha^{\text{out}2}$, uses a quadrature scaled appropriately for the interval to the right, and is computed so as to represent only the charges on the tiny interval $(a+b-r, a+b]$.

The routine *interact* takes as a parameter an interval number i. It computes the portion of the potential on the right half of the interval which is due to the charges on its left half, with both the potential and the charges being represented by expansions. The routine is as follows; Figure 4 contains a map of the expansions which it uses in a case of equally sized intervals, and Figure 5 contains a map of the expansions which it uses in a case of adjacent unequally sized intervals. (In places in the following pseudocode, conversion matrices which expect non-augmented exponential expansions are multiplied by augmented exponential expansions; this normally undefined operation is, here, used to mean that the additional terms which comprise the augmentation are ignored in that multiplication. Similarly, in places, a non-augmented expansion is added to an augmented expansion; the meaning is that the terms which the non-augmented expansion lacks are taken to be zero.)

```
subroutine interact (i)

i_1 := interval number of left half of i
i_2 := interval number of right half of i

α^in2 := 0
α^in3 := 0

while both i_1 and i_2 are higher level intervals {
        i_11 := interval number of left half of i_1
        i_12 := interval number of right half of i_1
        i_21 := interval number of left half of i_2
        i_22 := interval number of right half of i_2

        Comment { Compute three out of the four interactions, using far field to local
                  expansion potential operators }
```

$\beta_{i_{21}} := \beta_{i_{21}} + M_{fl1} \cdot \gamma_{i_{11}}$
$\beta_{i_{22}} := \beta_{i_{22}} + M_{fl2} \cdot \gamma_{i_{11}}$
$\beta_{i_{22}} := \beta_{i_{22}} + M_{fl1} \cdot \gamma_{i_{12}}$

Comment { Move down one level to examine the remaining interaction }
$i_1 := i_{12}$
$i_2 := i_{21}$
}

Comment { At this point at least one of i_1 or i_2 is a lowest level interval. }

If i_1 and i_2 are both lowest level intervals, then {
Comment { Use the outgoing exponential expansion from i_1 as the incoming exponential expansion to i_2, but delete the augmentation }
$\alpha^{\text{in}} := \alpha^{\text{out}}$
do $j = m+1, \ldots, m+k$ { $\alpha_j^{\text{in}} := 0$ }
}

If i_1 is a lowest level interval and i_2 is subdivided, then {
i_{21} := interval number of left half of i_2
i_{22} := interval number of right half of i_2

Comment { Use the far field to local expansion rescaling matrix to compute the local expansion on i_{22} }
$\beta_{i_{22}} := \beta_{i_{22}} + M_{de}^{*} \cdot \alpha^{\text{out}}$

Comment { Use the exponential expansion scaling-down matrix, plus the stub exponential expansion, to compute the incoming exponential expansion on i_{21} }
$\alpha^{\text{in}} := M_{ee}^{*} \cdot \alpha^{\text{out3}} + \alpha^{\text{out2}}$
}

If i_1 is subdivided and i_2 is a lowest level interval, then {
i_{11} := interval number of left half of i_1
i_{12} := interval number of right half of i_1

Comment { Use the exponential expansion scaling-up matrix to compute the incoming exponential expansion on i_{21}. }
$\alpha^{\text{in3}} := M_{ee} \cdot \alpha^{\text{out}}$

Comment { Use the far field to local expansion rescaling matrix to compute the local expansion on i_{22} }
$\alpha^{\text{in}} := M_{de} \cdot \gamma_{i_{11}}$

Comment { Use the old exponential expansion as the stub expansion }
$\alpha^{\text{in2}} := \alpha^{\text{out}}$
}

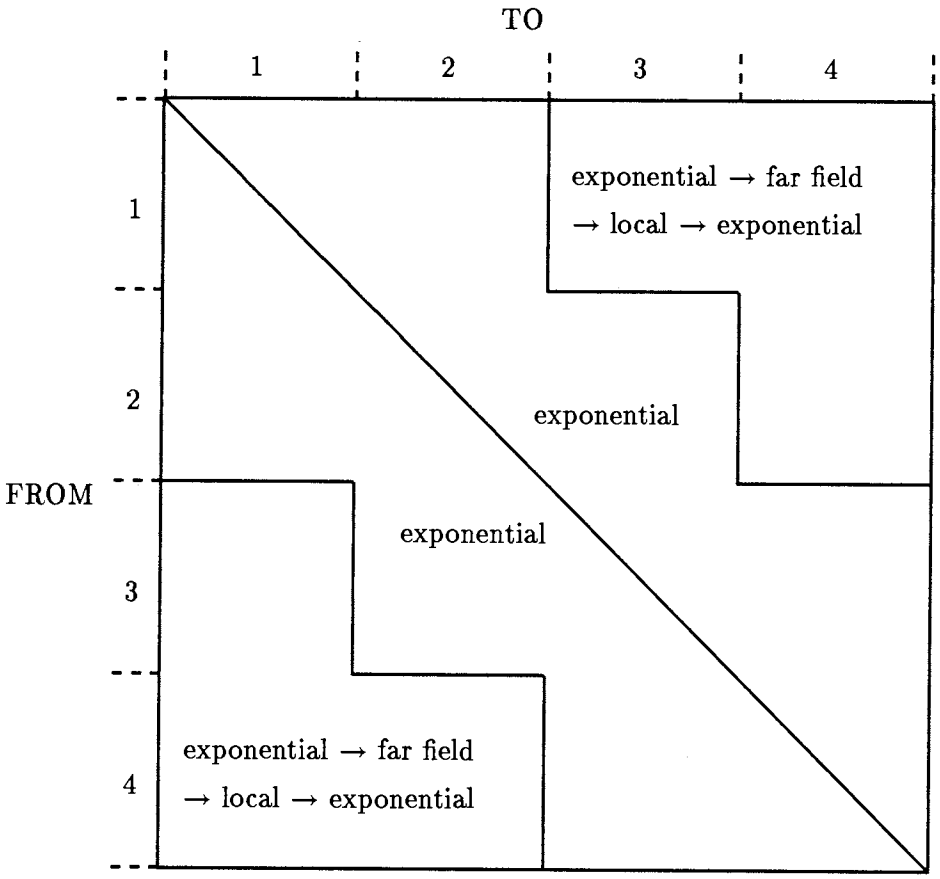


Figure 4: Map of expansions used for uniform subdivision.

The text at a vertical location x_1 and horizontal location x_2 lists the expansions used to compute the potential at x_2 from a charge at x_1. The rightwards arrow ($\rightarrow$) indicates a conversion from one expansion type to another.

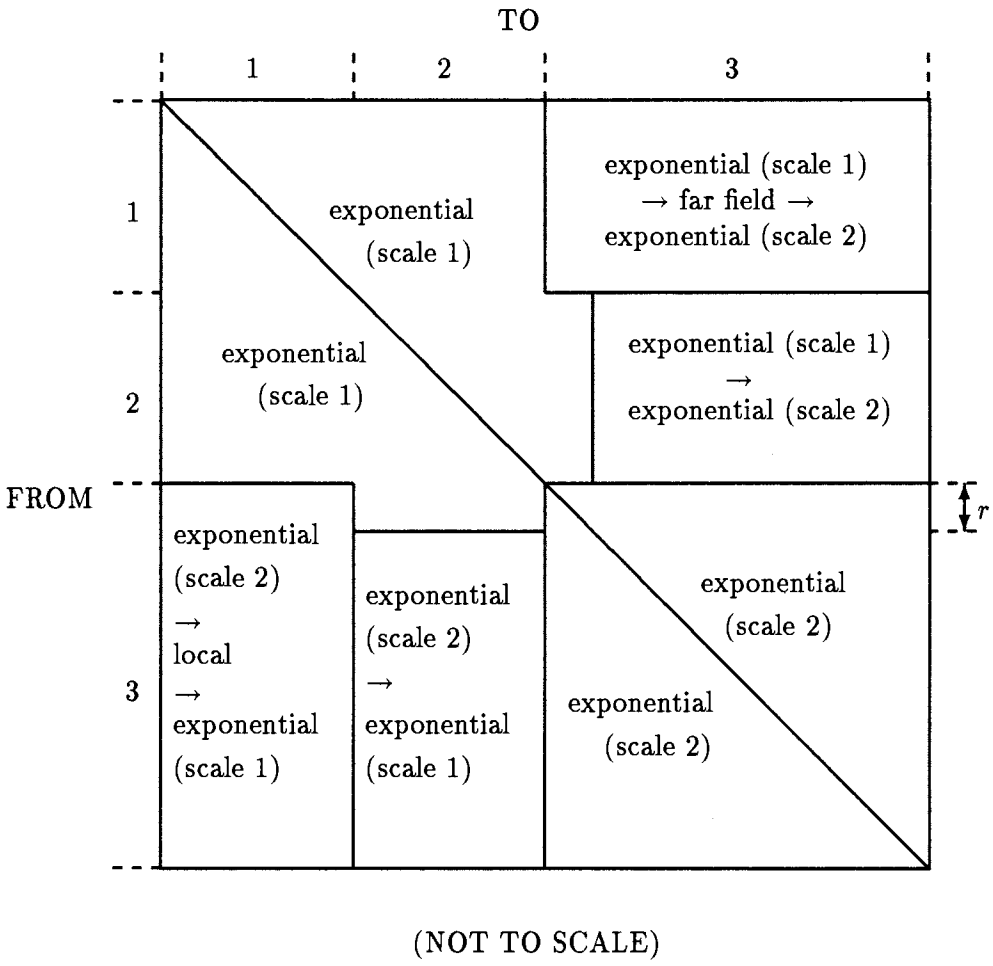


Figure 5: Map of expansions used for non-uniform subdivision

The text at a vertical location x_1 and horizontal location x_2 lists the expansions used to compute the potential at x_2 from a charge at x_1. The rightwards arrow ($\rightarrow$) indicates a conversion from one expansion type to another.

6.1 Execution time

The CPU time consumed by Algorithm 6.1 can be bounded as follows. The subroutine *lowlev* is called once on each lowest-level subinterval, and takes $O(m+k)$ time for each point therein, giving a total execution time of $O((m+k)n)$ (where m is the number of nodes in the quadrature on which the exponential expansions are based, k is the number of terms used in the augmentation of exponential expansions, and n is the number of input points). The routines *process* and *interact* spend their time in matrix multiplications, each of which is performed a number of times proportional to the number M of subintervals which are produced by the partitioning routine, and each of which takes either $O(p^2)$, $O(p(m+k))$, or $O(m(m+k))$ time (where p is the number of terms in a far field expansion). Since M is clearly $O(n)$, the algorithm as a whole takes $O(n(m+p^2+(p+m)(m+k)))$ time.

7 Numerical Results

Algorithm 6.1 has been implemented in double precision (Fortran `REAL*8`) arithmetic. The implementation used generalized Gaussian quadratures [9] as the quadratures for the exponential expansions. To achieve roughly double precision accuracy, a generalized Gaussian quadrature using 27 weights and nodes was used. To achieve roughly single precision accuracy, a generalized Gaussian quadrature using 14 weights and nodes was used. These quadratures are tabulated in [9], and are repeated here in Tables 14 and 15.

The sizes of certain expansions are determined numerically by the stage of the algorithm which computes the conversion and translation matrices; the values which were produced are as follows. For the $1/x$ kernel, the number p of terms in a far field or local expansion was 12 for double precision, and 6 for single precision, and the number k of multipole terms used to augment exponential expansions was 3 for double precision, and 2 for single precision. The sizes of expansions for the logarithmic kernel were identical, except that for single precision, the number of terms in a far field expansion was 7.

Algorithm 6.1 was run for two types of distributions of the points $\{x_i\}$, namely equispaced points and Chebyshev nodes; for both single and double precision expansions (but both in double precision arithmetic); and for both the logarithmic and $1/x$ kernels. The resulting timings (for a Sun Sparcstation 2) and accuracies are included as Tables 4–11. The execution times are listed in two ways: first as an absolute time, then as the ratio of the time taken by the algorithm to the time taken by an FFT of the same size. The execution times of the initialization and evaluation stages of Algorithm 6.1 are listed separately, in the columns labeled "Init" and "Eval" respectively. The column of the tables which is labeled "Error" contains the normalized L^2 error (the L^2 norm of the error, divided by the L^2 norm of the correct result) of Algorithm 6.1; the correct result was determined by the direct computation of (1) or (2) in extended precision (Fortran `REAL*16`) arithmetic. Also listed in Tables 4–11 are timings for the direct computation of (1), performed in double precision arithmetic. For large n, the computation becomes expensive, and was not performed; for these entries the tables contain extrapolations, calculated by assuming $O(n^2)$ behavior, which are indicated by parentheses.

Remark 7.1 *For the* $1/x$ *kernel, comparing against an extended precision calculation produces*

error figures which are considerably larger than those found if one compares against the direct computation done in double precision arithmetic. The errors seem to arise in the subtraction $x_i - x_k$; in the case that x_i and x_k are adjacent points (that is, $k = i - 1$ or $k = i + 1$), this subtraction has an $O(n)$ condition number for equispaced points and an $O(n^2)$ condition number for Chebyshev nodes. Since the interactions between adjacent points are much larger than the remaining interactions, their $O(n)$ or $O(n^2)$ error plays a large part in the total error. If both the test computation and the reference computation are done in double precision arithmetic, the error in the computation of $x_i - x_k$ is made exactly the same way in both, and thus is not observed. For the logarithmic kernel, this problem does not arise, since the interactions between adjacent points are, in general, little larger than the remaining interactions.

In Table 8, timings from [2] for the one-dimensional FMM of that paper are also presented, for comparison. (Of the two versions of the FMM in that paper, the one picked for comparison here, Algorithm 3.2, is the one with the faster evaluation times and slower precomputation times.) However, the timings for the algorithms are not directly comparable; the present algorithm calculates the potentials at the same points where the charges are located, but in the algorithm of [2] the two sets of locations need not be the same. Thus, when the points where the potentials are to be evaluated coincide with the locations of the charges, our scheme displays an improvement of roughly a factor of two over the method of [2] (in terms of CPU time requirements). On the other hand, when the charge locations are distinct from the points where the potentials are to be evaluated, most of the advantage of our algorithm is lost.

The column labeled N_{div} in Tables 4–11 contains the number of lowest-level subdivisions which the algorithm used. In the case where this number is equal to two, the timings printed serve also as the timings for Algorithm 5.1, since the implementation of Algorithm 6.1 avoids in this case any unnecessary computations of far field expansions, augmentations to exponential expansions, and so forth.

Table 13 contains execution times and accuracies for the exponential-expansion-only algorithm for the square root kernel, as described in Section 5.2. The input consisted of equispaced points. The unusually high initialization times (as compared to the other tables) are due to the large number of evaluations of the function I_0.

Table 12 contains timings and errors for a version of the algorithm which is optimized for equispaced points. With equispaced points, the precomputed translation operators and local correction coefficients are identical for all points $\{x_i\}$; thus the precomputation time is quite small. In addition, each of the subtractions $x_i - x_k$ can be calculated more accurately, using the formula $(x_n - x_1)(i - k)/n$; the tabulated accuracy is thus much better than the accuracy for the unoptimized algorithm (Table 8). However, this calculation is a convolution, and thus can be performed efficiently, and much more simply, using the FFT.

8 Conclusions and Generalizations

We have presented a version of the one-dimension fast multipole method which, by incorporating expansions based on generalized Gaussian quadratures, is in the best case roughly twice as fast as its predecessors, and in the worst case roughly breaks even with them. This algorithm exists

in versions for both the logarithmic and $1/x$ kernels. Generalizations to other kernels are somewhat problematic; unless those kernels possess benign scaling properties, as do the two kernels treated in this paper, the translation and conversion matrices would have to be recomputed for each scale, and perhaps for each location. Generalizations of the simple exponential-expansion-only algorithm are somewhat easier, and one such generalization (to the square root kernel) has been presented. However, the exponential-expansion-only algorithms are only efficient for small to medium numbers of points which are close to uniformly distributed.

References

[1] M. ABRAMOWITZ, I. STEGUN, *Handbook of Mathematical Functions*, Applied Mathematics Series, National Bureau of Standards, DC, 1964

[2] A. DUTT, M. GU, AND V. ROKHLIN, *Fast Algorithms for Polynomial Interpolation, Integration, and Differentiation*, SIAM Journal on Numerical Analysis, Vol. 33, No. 5, Oct 1996

[3] L. GREENGARD AND V. ROKHLIN, *A Fast Algorithm for Particle Simulations*, Journal Of Computational Physics, Vol. 73, No.2, Dec 1987

[4] T. HRYCAK, V. ROKHLIN, *An Improved Fast Multipole Algorithm for Potential Fields*, Research Report 1089, Yale Computer Science Department, 1995

[5] S. KAPUR, V. ROKHLIN, *An Algorithm for the fast Hankel transform*, Technical Report 1045, Yale Computer Science Department, 1995

[6] J. MA, V. ROKHLIN, AND S. WANDZURA, *Generalized Gaussian Quadratures For Systems of Arbitrary Functions*, SIAM Journal of Numerical Analysis, June 1996

[7] M. REED, B. SIMON, *Methods of modern mathematical physics, Vol. 1*, Academic Press, 1980

[8] J. STOER, R. BULIRSCH, *Introduction to Numerical Analysis, Second Edition*, Springer-Verlag, 1993

[9] N. YARVIN, V. ROKHLIN, *Generalized Gaussian Quadratures and Singular Value Decompositions of Integral Operators*, Research Report 1109, Yale Computer Science Department, 1996

Table 4: Single precision timings for the $1/x$ kernel, for equispaced points

N	N_{div}	Memory	Error	Times (seconds)			Timing Ratios	
		(REAL*8 spaces)		Init	Eval	Direct	Eval/ FFT	Init/ FFT
64	3	2410	0.64E-07	0.008	0.003	0.004	5.60	16.77
128	3	3338	0.49E-07	0.013	0.004	0.017	4.65	14.58
256	3	5194	0.55E-07	0.024	0.008	0.071	3.99	12.52
512	3	9417	0.36E-07	0.057	0.016	0.273	2.89	10.43
1024	3	18375	0.37E-07	0.137	0.034	1.195	2.51	10.15
2048	3	39360	0.32E-07	0.367	0.074	5.270	2.61	12.85
4096	15	71049	0.39E-07	0.559	0.168	22.250	2.42	8.04
8192	31	141281	0.39E-07	1.150	0.327	92.800	1.77	6.23
16384	63	281745		2.320	0.660	(3.7E+02)	1.67	5.87
32768	127	562673		4.830	1.332	(1.5E+03)	1.62	5.89
65536	255	1124529		9.690	2.685	(5.9E+03)	1.51	5.46

Table 5: Single precision timings for the $1/x$ kernel, for Chebyshev nodes

N	N_{div}	Memory	Error	Times (seconds)			Timing Ratios	
		(REAL*8 spaces)		Init	Eval	Direct	Eval/ FFT	Init/ FFT
64	3	2412	0.53E-07	0.008	0.003	0.004	5.67	16.97
128	3	3354	0.30E-07	0.014	0.004	0.017	4.83	15.23
256	3	5300	0.22E-07	0.026	0.015	0.068	7.78	13.57
512	3	9506	0.19E-07	0.059	0.016	0.274	2.94	10.93
1024	3	19477	0.19E-07	0.169	0.036	1.200	2.69	12.49
2048	7	38737	0.75E-08	0.344	0.086	5.290	2.98	11.97
4096	15	73061	0.80E-08	0.611	0.171	22.350	2.44	8.73
8192	27	144257	0.77E-08	1.212	0.338	93.000	1.84	6.58

Table 6: Single precision timings for the logarithmic kernel, for equispaced points

N	N_{div}	Memory	Error	Times (seconds)			Timing Ratios	
		(REAL*8 spaces)		Init	Eval	Direct	Eval/ FFT	Init/ FFT
64	3	2547	0.63E-08	0.008	0.003	0.018	5.05	16.55
128	3	3475	0.43E-08	0.013	0.004	0.071	4.88	15.09
256	3	5331	0.53E-08	0.024	0.008	0.286	4.12	12.52
512	3	9554	0.53E-08	0.061	0.016	1.151	3.05	11.36
1024	3	18512	0.44E-08	0.151	0.037	4.707	2.72	11.13
2048	3	39497	0.55E-08	0.422	0.076	19.340	2.65	14.68
4096	15	71210	0.92E-08	0.613	0.161	78.590	2.28	8.70
8192	31	141474	0.43E-08	1.268	0.321	320.510	1.65	6.54
16384	63	282002		2.455	0.646	(1.3E+03)	1.65	6.26
32768	127	563058		4.910	1.300	(5.1E+03)	1.59	6.01
65536	255	1125170		9.780	2.605	(2.1E+04)	1.49	5.59

Table 7: Single precision timings for the logarithmic kernel, for Chebyshev nodes

N	N_{div}	Memory	Error	Times (seconds)			Timing Ratios	
		(REAL*8 spaces)		Init	Eval	Direct	Eval/ FFT	Init/ FFT
64	3	2549	0.56E-08	0.008	0.003	0.018	5.09	16.49
128	3	3491	0.54E-08	0.015	0.004	0.072	4.92	16.49
256	3	5437	0.56E-08	0.027	0.008	0.288	4.24	14.04
512	3	9643	0.53E-08	0.068	0.016	1.162	3.03	12.57
1024	3	19614	0.55E-08	0.183	0.037	4.785	2.71	13.60
2048	7	38882	0.79E-08	0.386	0.086	19.610	3.00	13.51
4096	15	73222	0.33E-08	0.671	0.166	79.310	2.36	9.55
8192	27	144442	0.26E-08	1.335	0.332	323.490	1.81	7.29

Table 8: Double precision timings for the $1/x$ kernel, for equispaced points

N	N_{div}	Error	Times (seconds)			Timing Ratios		Old ratios	
			Init	Eval	Direct	Eval/ FFT	Init/ FFT	Eval/ FFT	Init/ FFT
64	3	0.52E-14	0.017	0.004	0.004	8.41	34.53		
128	3	0.65E-14	0.026	0.007	0.017	8.04	29.30		
256	3	0.11E-13	0.046	0.014	0.068	7.47	24.45		
512	3	0.27E-13	0.106	0.029	0.273	5.36	19.73	12.2	78.3
1024	3	0.25E-13	0.253	0.059	1.228	4.39	18.69	11.7	65.7
2048	3	0.11E-12	0.596	0.124	5.280	4.35	20.84	10.6	57.6
4096	15	0.10E-12	0.956	0.277	22.240	3.98	13.75	9.7	49.8
8192	31	0.17E-12	2.017	0.553	92.400	3.02	11.02		
16384	63		3.795	1.118	(3.7E+02)	2.86	9.70		
32768	127		8.270	2.255	(1.5E+03)	2.78	10.18		
65536	255		16.121	4.900	(5.9E+03)	2.68	8.83		

Table 9: Double precision timings for the $1/x$ kernel, for Chebyshev nodes

N	N_{div}	Memory	Error	Times (seconds)			Timing Ratios	
		(REAL*8 spaces)		Init	Eval	Direct	Eval/ FFT	Init/ FFT
64	3	5833	0.13E-13	0.017	0.004	0.004	8.17	34.58
128	3	7607	0.49E-13	0.026	0.007	0.017	8.14	29.29
256	3	11217	0.77E-13	0.049	0.014	0.070	7.20	24.78
512	3	18751	0.13E-11	0.108	0.039	0.279	7.23	19.95
1024	3	35378	0.21E-11	0.272	0.061	1.207	4.55	20.12
2048	3	73866	0.32E-11	0.782	0.135	5.570	4.69	27.24
4096	15	130626	0.13E-10	1.020	0.287	22.380	4.07	14.46
8192	27	256798	0.15E-09	2.015	0.572	92.950	3.10	10.91

Table 10: Double precision timings for the logarithmic kernel, for equispaced points

N	N_{div}	Memory	Error	Times (seconds)			Timing Ratios	
		(REAL*8 spaces)		Init	Eval	Direct	Eval/ FFT	Init/ FFT
64	3	5929	0.20E-14	0.017	0.004	0.018	8.28	35.58
128	3	7689	0.19E-14	0.026	0.007	0.071	8.26	29.36
256	3	11209	0.48E-14	0.046	0.014	0.284	7.30	23.47
512	3	18760	0.49E-14	0.111	0.033	1.151	6.05	20.53
1024	3	34374	0.31E-14	0.249	0.061	4.767	4.48	18.39
2048	3	68671	0.78E-14	0.652	0.127	19.330	4.42	22.69
4096	15	128712	0.69E-14	1.038	0.268	78.789	3.82	14.77
8192	31	254496	0.62E-14	2.025	0.546	320.520	2.97	11.02
16384	63	506064		4.090	1.097	(1.3E+03)	2.80	10.43
32768	127	1009200		8.221	2.247	(5.1E+03)	2.75	10.06
65536	255	2015472		16.790	4.600	(2.1E+04)	2.49	9.10

Table 11: Double precision timings for the logarithmic kernel, for Chebyshev nodes

N	N_{div}	Memory	Error	Times (seconds)			Timing Ratios	
		(REAL*8 spaces)		Init	Eval	Direct	Eval/ FFT	Init/ FFT
64	3	5931	0.73E-15	0.017	0.004	0.018	8.30	35.32
128	3	7705	0.60E-15	0.026	0.007	0.072	7.07	25.45
256	3	11315	0.86E-15	0.050	0.015	0.288	7.59	25.92
512	3	18849	0.11E-14	0.108	0.030	1.166	5.59	19.99
1024	3	35476	0.17E-14	0.291	0.102	4.757	7.57	21.57
2048	3	73964	0.21E-14	0.857	0.138	19.580	4.83	30.00
4096	15	130724	0.33E-14	1.077	0.282	79.180	4.04	15.44
8192	27	256896		2.143	0.560	322.230	3.03	11.60

Table 12: Double precision timings for the $1/x$ kernel, optimized for equispaced points

N	N_{div}	Memory	Error	Times (seconds)			Timing Ratios	
		(REAL*8 spaces)		Init	Eval	Direct	Eval/ FFT	Init/ FFT
64	3	4130	0.43E-14	0.006	0.004	0.004	7.43	13.10
128	3	4162	0.28E-14	0.006	0.006	0.017	6.76	6.89
256	3	4226	0.19E-14	0.006	0.011	0.069	5.84	3.31
512	3	4865	0.46E-14	0.007	0.022	0.273	4.15	1.30
1024	3	6655	0.35E-14	0.008	0.046	1.190	3.36	0.58
2048	3	13304	0.55E-14	0.010	0.098	5.270	3.21	0.33
4096	15	18049	0.18E-13	0.031	0.212	22.740	3.01	0.44
8192	31	33241	0.70E-13	0.063	0.422	92.630	2.29	0.34
16384	63	63625		0.134	0.859	(3.7E+02)	2.17	0.34
32768	127	124393		0.293	1.722	(1.5E+03)	2.09	0.36
65536	255	245929		0.578	3.472	(5.9E+03)	1.96	0.33

Table 13: Timings for the square root kernel, for equispaced points

N	Error	Times (seconds)			Timing Ratios	
		Init	Eval	Direct	Eval/ FFT	Init/ FFT
64	0.30E-14	0.041	0.002	0.018	3.70	82.64
128	0.23E-14	0.083	0.004	0.075	4.06	83.79
256	0.21E-14	0.167	0.010	0.308	4.75	80.80
512	0.32E-14	0.352	0.020	1.237	3.78	64.97
1024	0.19E-14	0.733	0.041	4.920	3.26	57.78
2048	0.29E-14	1.591	0.087	19.430	3.16	57.47
4096	0.34E-14	3.733	0.182	78.140	2.61	53.53
64	0.54E-07	0.021	0.001	0.018	2.19	42.98
128	0.51E-07	0.043	0.002	0.074	2.30	43.36
256	0.48E-07	0.086	0.005	0.307	2.44	41.89
512	0.48E-07	0.184	0.013	1.238	2.34	33.94
1024	0.45E-07	0.392	0.025	4.890	1.95	30.58
2048	0.44E-07	0.879	0.053	19.500	1.90	31.73
4096	0.44E-07	2.140	0.115	77.940	1.65	30.68

Table 14: Quadratures for exponentials

Quadratures for the integral

$$\int_0^\infty e^{-xt}dx,$$

under the condition that $1 \leq t \leq 500$.

N	Nodes (x_i)	Weights (w_i)	Error
14	0.1075073588251350E-02	0.2783455121689438E-02	0.366E-07
	0.5889243490962496E-02	0.7006395914900820E-02	
	0.1560078432135377E-01	0.1279502133157069E-01	
	0.3258052212086110E-01	0.2192733340131016E-01	
	0.6154351752779967E-01	0.3737740049082059E-01	
	0.1109619891032348E+00	0.6379243969367225E-01	
	0.1951651530857407E+00	0.1084594588227473E+00	
	0.3377699882687942E+00	0.1830223278438481E+00	
	0.5772805419211481E+00	0.3061647832783700E+00	
	0.9761165652290038E+00	0.5079755103629931E+00	
	0.1635615445691163E+01	0.8381174751258640E+00	
	0.2723809484786727E+01	0.1385562498413431E+01	
	0.4541163041303490E+01	0.2347348786059432E+01	
	0.7767616655342678E+01	0.4444622409829190E+01	

Table 15: Quadratures for exponentials (continued)

Quadratures for the integral

$$\int_0^\infty e^{-xt} dx,$$

under the condition that $1 \leq t \leq 500$.

N	Nodes (x_i)	Weights (w_i)	Error
27	0.5378759010624780E-03	0.1383311204046008E-02	0.323E-14
	0.2860176825815242E-02	0.3279869733166365E-02	
	0.7148658617716300E-02	0.5330932895600203E-02	
	0.1360965515937845E-01	0.7646093110803760E-02	
	0.2257800188133212E-01	0.1037458793227033E-01	
	0.3456421989535069E-01	0.1372178039022047E-01	
	0.5032042618508775E-01	0.1796868836009351E-01	
	0.7092509447124836E-01	0.2348971809947674E-01	
	0.9788439120828463E-01	0.3076860552710760E-01	
	0.1332509921950535E+00	0.4041894092839717E-01	
	0.1797695570864978E+00	0.5321827718681367E-01	
	0.2410654714132133E+00	0.7016094768858448E-01	
	0.3218961915636380E+00	0.9253048536912244E-01	
	0.4284852078938826E+00	0.1219928996130354E+00	
	0.5689615509235298E+00	0.1607156476580828E+00	
	0.7539347736933301E+00	0.2115215602167892E+00	
	0.9972472224438443E+00	0.2780925850550500E+00	
	0.1316964566299846E+01	0.3652478333806065E+00	
	0.1736698582009859E+01	0.4793398853949993E+00	
	0.2287418444638146E+01	0.6288554258416082E+00	
	0.3010034073439038E+01	0.8254021100491956E+00	
	0.3959315495048493E+01	0.1085495633209734E+01	
	0.5210381702393131E+01	0.1434174907278760E+01	
	0.6870768194824406E+01	0.1913323186889750E+01	
	0.9106577764323245E+01	0.2604342790201154E+01	
	0.1221294512896673E+02	0.3708436699287805E+01	
	0.1689348652665484E+02	0.6023086156615004E+01	

Table 16: Quadratures for exponentials multiplied by I_0

Quadratures for the integral

$$\int_0^\infty I_0(xy)e^{-xt}dx,$$

under the condition that $t \in [1, 500]$ and $y \in [0, t-1]$.

N	Nodes (x_i)	Weights (w_i)	Error
14	0.642288534795956E-03	0.166796436786039 5E-02	0.900E-07
	0.3562319666990144E-02	0.4298903067080389E-02	
	0.9643424057074440E-02	0.8159545461265918E-02	
	0.2074298599770349E-01	0.1463864640961027E-01	
	0.4057928260022333E-01	0.2614391453322226E-01	
	0.7600572280169251E-01	0.4665755537868725E-01	
	0.1390443053485344E+00	0.8276361628521883E-01	
	0.2503136051566992E+00	0.1454478222995341E+00	
	0.4447622918282108E+00	0.2529871458046016E+00	
	0.7811276346003586E+00	0.4357009925372973E+00	
	0.1357818162257100E+01	0.7446059596729966E+00	
	0.2341992534236977E+01	0.1271434906786924E+01	
	0.4036529413075654E+01	0.2216807353831690E+01	
	0.7126502974662635E+01	0.4302367103374836E+01	

Table 17: Quadratures for exponentials multiplied by I_0 (continued)

Quadratures for the integral

$$\int_0^\infty I_0(xy)e^{-xt}dx,$$

under the condition that $t \in [1, 500]$ and $y \in [0, t-1]$.

N	Nodes (x_i)	Weights (w_i)	Error
29	0.2855179413353365E-03	0.7344503079351386E-03	0.299E-14
	0.1519624696728258E-02	0.1744538390662211E-02	
	0.3804359141657344E-02	0.2844687196642974E-02	
	0.7260138000706486E-02	0.4098961298933580E-02	
	0.1208205371062810E-01	0.5593550200298448E-02	
	0.1856564543199398E-01	0.7444670271885530E-02	
	0.2714156753309568E-01	0.9807968524698940E-02	
	0.3842017800878239E-01	0.1288914176031762E-01	
	0.5324783256625659E-01	0.1695687345717790E-01	
	0.7277755829761968E-01	0.2235879759838917E-01	
	0.9855788611173273E-01	0.2954235698585380E-01	
	0.1326465035778468E+00	0.3908423859367898E-01	
	0.1777590387840778E+00	0.5173159700695577E-01	
	0.2374657658898870E+00	0.6845695550893067E-01	
	0.3164509240422835E+00	0.9052903520482935E-01	
	0.4208524457939620E+00	0.1196036182345700E+00	
	0.5587051648321601E+00	0.1578409524693449E+00	
	0.7405185479404663E+00	0.2080593451794129E+00	
	0.9800319873390735E+00	0.2739397750144418E+00	
	0.1295209795621391E+01	0.3603059290242059E+00	
	0.1709570851677607E+01	0.4735231867763476E+00	
	0.2254009385987865E+01	0.6221016600956893E+00	
	0.2969389638669206E+01	0.8176841100086656E+00	
	0.3910476327629495E+01	0.1076831175000068E+01	
	0.5152430007642100E+01	0.1424628439002124E+01	
	0.6802867813529709E+01	0.1902988149814232E+01	
	0.9027979519502084E+01	0.2593285548365225E+01	
	0.1212289908066820E+02	0.3696550722303479E+01	
	0.1679085599535762E+02	0.6009492062220468E+01	