A Fast Algorithm for the Discrete Laplace Transformation

V. Rokhlin
Research Report YALEU/DCS/RR-509
January 1987

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE
An algorithm is presented for the rapid evaluation of expressions of the form

\[ \sum_{j=1}^{m} \alpha_j \cdot e^{-\beta_j x} \]

at multiple points \(x_1, x_2, \ldots, x_n\). In order to evaluate the above sum at \(n\) points, the algorithm requires order \(O(n + m)\) operations, and a simple modification of the scheme provides an order \(O(n)\) procedure for the evaluation of an order \(n\) polynomial at \(n\) arbitrary real points. The algorithm is numerically stable, and its practical usefulness is demonstrated by numerical examples.

A Fast Algorithm for the Discrete Laplace Transformation

V. Rokhlin
Research Report YALEU/DCS/RR-509
January 1987

The author was supported in part by the Office of Naval Research under Grant N00014-86-K-0310
1. Introduction

In this paper, we present an algorithm for the rapid evaluation of expressions of the form

\[ \sum_{j=1}^{m} \alpha_j \cdot e^{-\beta_j \epsilon}, \tag{1} \]

where \( x \geq 0, \mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}, \mathcal{B} = \{\beta_1, \beta_2, \ldots, \beta_m\} \) are two finite sequences of real numbers and \( \beta_j \geq 0 \) for all \( 1 \leq j \leq m \). To evaluate the sum (1) at \( n \) arbitrary points on the real axis, the algorithm requires a number of arithmetic operations proportional to

\[ (n + m \cdot \log_2(\frac{1}{\epsilon})) \cdot (\log_2(\frac{1}{\epsilon}))^2, \tag{2} \]

where \( \epsilon \) is the precision with which the calculations are being performed, and in most cases likely to be encountered in practice, the estimate (2) can be reduced to

\[ (n + m) \cdot \log_2(\frac{1}{\epsilon}) \tag{3} \]

(see Observations 7.1, 7.2 below).

The evaluation of expressions of the form (1) is closely related to several classical problems in the theory of computation. For example, the problem of rapidly evaluating a polynomial

\[ P(t) = \sum_{j=1}^{m} P_j \cdot t^j \tag{4} \]

at \( m \) different points is readily reduced to the form (1) by the obvious substitution \( x = \log(t) \). The classical algorithm for evaluating (4) at \( m \) points has an asymptotic complexity \( O(m \cdot \log^2(m)) \) (see, for example, [1, 2]), making (3) a moderate improvement over previously available results, so far as the asymptotic CPU time estimate is concerned. On the other hand, the algorithm of the present paper is numerically stable, and our numerical experiments (see Section 8) indicate that in practical calculations, it is extremely efficient, making it a method of choice whenever expressions of the form (1) have to be evaluated at large numbers of points.

**Remark 1.1.** Classical algorithms for the rapid manipulation of polynomials are purely algebraic, and are applicable to polynomials over a wide class of fields. On the other hand, the algorithm presented here is based on approximation theory (i.e. it relies on certain facts from real analysis) and is restricted to polynomials over the field of real numbers. While it can be generalized to certain other fields, detailed investigation of such generalizations is outside the scope of this paper, and will be reported at a later date.

2. Relevant Facts From Approximation Theory

Suppose that \( a, b \) are a pair of real numbers such that \( a < b \), and that \( k \geq 2 \) is an integer. Chebyshev nodes \( t_1, t_2, \ldots, t_k \) on the interval \([a, b]\) are defined by the formula

\[ t_i = \frac{a + b}{2} + \frac{a - b}{2} \cdot \cos\left(\frac{2 \cdot i + 1}{k} \cdot \frac{\pi}{2}\right). \tag{5} \]
For a function \( f : [a, b] \rightarrow \mathbb{R} \), we will denote by \( P_{a,b}^{k} \) the order \( k - 1 \) Chebychev approximation to the function \( f \) on the interval \([a, b] \), i.e. the (unique) polynomial of order \( k - 1 \) such that \( P_{a,b}^{k}(t_i) = f(t_i) \) for all \( i = 1, 2, \ldots, k \). There exist several expressions for the polynomial \( P_{a,b}^{k} \), and the one we will use in this paper is

\[
P_{a,b}^{k}(t) = \sum_{j=1}^{k} u_{j}(t) \cdot f(t_j)
\]

with

\[
u_{j}(t) = \frac{\prod_{i=1, i \neq j}^{k} (t - t_i)}{\prod_{i=1, i \neq j}^{k} (t_j - t_i)}.
\]

The following well-known lemma provides an error estimate for Chebychev approximations. It is the principal analytical tool of this paper, and can be found, in a somewhat different form, in [3].

**Lemma 2.1.** If \( f \in \mathcal{C}^{k}[a,b] \) (i.e. \( f \) has \( k \) continuous derivatives on the interval \([a,b] \)), then for any \( t \in [a,b] \),

\[
|P_{a,b}^{k}(t) - f(t)| \leq \frac{M}{k!} \cdot \frac{(b-a)^k}{4^k}
\]

with

\[
M = \max_{t \in [a,b]} |f^{(k)}(t)|.
\]

Furthermore, for any \( k \geq 2 \) and \( t \in [a,b] \),

\[
\sum_{j=1}^{k} u_{j}^{2}(t) \leq 2.
\]

and

\[
\sum_{j=1}^{k} |u_{j}(t)| \leq 2 + \frac{2}{\pi} \cdot log(k).
\]

In the present paper, the above lemma will be used in the special case where \( 0 < a < b \), and \( f(t) = e^{-\gamma t} \), with \( \gamma > 0 \). Under these conditions, the expression \((8)\) assumes the form

\[
|P_{a,b}^{k}(t) - f(t)| \leq \frac{\gamma^k}{k!} \cdot \frac{(b-a)^k}{4^k} \cdot e^{-\gamma t}.
\]

and the following lemma provides a form of the estimate \((12)\) independent of \( \gamma \).

**Lemma 2.2.** If under the conditions of Lemma 2.1, \( f(t) = e^{-\gamma t} \), \( b = 2a \), and \( \alpha > 0 \), then

\[
|P_{a,b}^{k}(t) - f(t)| \leq \frac{1}{4^k}
\]
for all \( k \geq 2 \) and \( t \in [a, b] \).

**Proof.** Obviously, for \( t \in [a, 2a] \), the estimate (12) can be rewritten in the form

\[
| P_{a,b,f}^k(t) - f(t) | \leq \frac{\gamma^k}{k!} \cdot \frac{a^k}{4^k} \cdot e^{-\gamma a}.
\]

Differentiating the latter expression with respect to \( \gamma \), we find that its maximum is achieved at

\[
\gamma = \frac{k}{a}.
\]

Now, substituting (15) into (14) and using Stirling's formula, we obtain

\[
| P_{a,b,f}^k(t) - f(t) | \leq \frac{1}{k!} \cdot \left( \frac{k}{a} \right)^k \cdot \frac{a^k}{4^k} \cdot e^{-\frac{1}{2} \frac{a}{k}} \leq \frac{1}{4^k} \cdot \frac{k^k}{k!} \cdot e^{-k} \leq \frac{1}{4^k}.
\]

3. **Exact Statement of the Problem**

In the description of the algorithm below, we will assume that:

a) \( \hat{a} = \{a_1, a_2, \ldots, a_m\} \), \( \hat{b} = \{b_1, b_2, \ldots, b_m\} \), \( \hat{x} = \{x_1, x_2, \ldots, x_n\} \) are three finite sequences of real numbers.

b) The sequences \( \hat{a} \) and \( \hat{b} \) are monotonically increasing.

c) \( a_i \geq 0 \).

d) \( b_i \geq 0 \).

e) We would like to evaluate the sums

\[
S_{n, \hat{a}}(x_k) = \sum_{j=1}^m a_j \cdot e^{-b_j} x_k
\]

for all \( k = 1, 2, \ldots, n \) with a relative accuracy \( \epsilon > 0 \), i.e. we would like to find a number \( \hat{S}_{n, \hat{a}}(x_k) \) such that

\[
\frac{| \hat{S}_{n, \hat{a}}(x_k) - S_{n, \hat{a}}(x_k) |}{\sum_{j=1}^m | r_j |} \leq \epsilon
\]

for each \( k \in [1, n] \).

**Remark 3.1.** As has been mentioned in the Introduction, the problem of evaluating a polynomial of order \( m \) at \( n \) points is easily reduced to the form (17). Indeed, suppose that a polynomial of the form (4) has to be evaluated at a monotonically increasing finite sequence of points \( t_1, t_2, \ldots, t_m \). It can be assumed without a loss of generality that \( 0 \leq t_k \leq 1 \) for all \( k = 1, 2, \ldots, m \), and we will introduce a new variable \( x = -\log(t) \), and denote \(-\log(t_k)\) by \( x_k \). Thus, evaluation of the polynomial (4) at a monotonically increasing finite sequence of points has been reduced to evaluating the expression

\[
\sum_{j=1}^m P_j \cdot e^{-x_j}
\]
at a monotonically decreasing finite sequence of points \( \tilde{x} = \{x_1, x_2, \ldots, x_n\} \). Finally, by reversing the order of the sequence \( \tilde{x} \), we reduce the evaluation of the polynomial \( (4) \) at the points \( t_1, t_2, \ldots, t_n \) to the standard problem formulated above.

4. Notation

In this section, we will introduce several definitions to be used in the description of the algorithm in Sections 5, 6 below. Throughout this section, we will assume that we are dealing with the problem described in Section 3, and that \( q \) is an integer whose particular value is to be determined later.

We will denote by \( M \) the smallest integer number such that

\[
\frac{\beta_m x_n}{2^{M-1}} < 1. \tag{20}
\]

We will define a finite sequence \( \{U_i\}, i = 1, 2, \ldots, M \) of intervals on the real axis by the formulae

\[
U_i = (\frac{\beta_m}{2^i}, \frac{\beta_m}{2^{i-1}}) \text{ for } 1 \leq i \leq M - 1.
\]

\[
U_M = [0, \frac{\beta_m}{2^{M-1}}]. \tag{21}
\]

Similarly, we will define a finite sequence \( \{V_i\}, i = 1, 2, \ldots, M \) of intervals by the formulae

\[
V_i = (\frac{x_n}{2^i}, \frac{x_n}{2^{i-1}}) \text{ for } 1 \leq i \leq M - 1.
\]

\[
V_M = [0, \frac{x_n}{2^{M-1}}]. \tag{22}
\]

For any \( i = 1, 2, \ldots, M \), we will denote by \( \tilde{x}_i \) the subset of \( \tilde{x} \) consisting of all points \( x \) such that \( x \in U_i \), and for any \( i = 1, 2, \ldots, M \), we will denote by \( x_j \) the subset of \( \tilde{x} \) consisting of all points \( x_j \) such that \( x_j \in V_i \).

For each \( i = 1, 2, \ldots, M, m_i \) will denote the number of elements in \( \tilde{x}_i \). Similarly, for each \( i = 1, 2, \ldots, M, n_i \) will denote the number of elements in \( x_j \).

Remark 4.1. Obviously, depending on the distributions of the points \( \tilde{x} \) and \( x_j \), the \( M \) can be fairly large. However, the total number \( \sum \) of such \( i \) that \( m_i \neq 0 \) is bounded by \( m \), and the total number \( \sum \) of such \( i \) that \( n_i \neq 0 \) is bounded by \( n \). For obvious reasons, we will refer to empty intervals \( U_i, V_i \) such that \( m_i = 0 \) and \( n_i = 0 \). In the opposite case, the intervals will be referred to as non-empty.

For each \( i = 1, 2, \ldots, M \) and \( j = 1, 2, \ldots, q \), we will denote by \( \tilde{x}_j^{(i)} \) the \( j \)-th Chebyshev node on the interval \( U_i \).

Similarly, for each \( i = 1, 2, \ldots, M \) and \( j = 1, 2, \ldots, q \), we will denote by \( x_j^{(i)} \) the \( j \)-th Chebyshev node on the interval \( V_i \).

For each \( k = 1, 2, \ldots, M \) and \( i \) such that \( \tilde{x}_i \in U_k \), we will define the finite sequence \( \{\tilde{x}_j^{(k,i)}\}, j = 1, 2, \ldots, q \) by the formula

\[
\tilde{x}_j^{(k,i)} = \begin{cases} x_j^{(i)} & \text{if } j \leq q \text{ and } x_j^{(i)} \in U_k, \\ \tilde{x}_j^{(k,i)} & \text{if } j > q \text{ and } x_j^{(i)} \in U_k. \end{cases}
\]
\[ a_{i,j}^k = \prod_{t=1}^{q} \frac{\lambda_t - \lambda_j^k}{\lambda_t^j - \lambda_i^k}. \]  

(23)

For each \( k = 1, 2, \ldots, M \) and \( i = 1, 2, \ldots, q \), we will define a real number \( u_i^k \) by the formula

\[ u_i^k = \sum_{j \in \mathcal{E}_i^k} \alpha_j \cdot u_{i,j}^k. \]

(24)

**Observation 4.1.** Due to Lemma 2.2, the expression

\[ \phi_i^k(t) = \sum_{j=1}^{q} a_{i,j}^k \cdot e^{-\lambda_j^k t} \]

(25)

can be viewed as an approximation to the function \( e^{-\lambda_i^k t} \). Furthermore, for any \( t \in [0, \infty] \),

\[ |\phi_i^k(t) - e^{-\lambda_i^k t}| \leq \frac{1}{4q}. \]

(26)

Combining (24), (25), (26) with the triangle inequality, we easily see that the sum

\[ \phi_i(t) = \sum_{j=1}^{q} u_i^k \cdot e^{-\lambda_j^k t} = \sum_{j=1}^{q} \sum_{j \in \mathcal{E}_i^k} \alpha_j \cdot u_{i,j}^k \cdot e^{-\lambda_j^k t} = \sum_{j \in \mathcal{E}_i^k} \alpha_j \cdot \sum_{i=1}^{q} a_{i,j}^k \cdot e^{-\lambda_j^k t} \]

(27)

can be viewed as an approximation to

\[ \sum_{j \in \mathcal{E}_i^k} \alpha_j \cdot e^{-\lambda_j^k t}, \]

(28)

and that

\[ |\phi_i(t) - \sum_{j \in \mathcal{E}_i^k} \alpha_j \cdot e^{-\lambda_j^k t}| \leq \frac{1}{4q} \cdot \sum_{j \in \mathcal{E}_i^k} |\alpha_j|. \]

(29)

Furthermore, combining (11) with (24) and using the triangle inequality, we obtain

\[ \sum_{i=1}^{q} |a_i^k| \leq (|\alpha_j| \sum_{i=1}^{q} |u_{i,j}^k|) \leq (2 + \frac{2}{\pi} \cdot \log(q)) \cdot \sum_{j \in \mathcal{E}_i^k} |\alpha_j|. \]

(30)

Given \( k = 1, 2, \ldots, M \) and \( i = 1, 2, \ldots, q \), we will define a real number \( f_i^k \) by the expression

\[ f_i^k = \sum_{j=1}^{q} \sum_{i=1}^{\alpha_i - 1} u_{i,j}^k \cdot e^{-\lambda_i^k t}. \]

(31)
For each $k = 1, 2, \ldots, M$, and $1 \leq j \leq n$ such that $x_j \in V_k$, we will define $f_j$ by the formula

$$f_j = \sum_{i=1}^{q} r_{i,j} \cdot f_i^k,$$

with the coefficients $r_{i,j}^k$ defined by the formula

$$r_{i,j}^k = \prod_{i=1, i \neq j}^{q} \frac{x_j - x_i^k}{x_i^k - x_i^k}.$$

**Observation 4.2.** Due to Lemma 2.2, for any $j = 1, 2, \ldots, n$ and $k$ such that $x_j \in V_k$, $f_j$ can be viewed as an approximation to the expression

$$\tilde{f}_j = \sum_{i=\nu_k+1}^{\mu_k-1} \sum_{l=1}^{q} u_l^{i} \cdot e^{-\frac{1}{l} \cdot x_j},$$

and

$$|f_j - \tilde{f}_j| \leq \frac{1}{4^q} \cdot \sum_{i=\nu_k+1}^{\mu_k-1} \sum_{l=1}^{q} |u_l^i|.$$  

Combining (35), (20) (30), and using the triangle inequality, we conclude that

$$|f_j - \sum_{i=1}^{m} \alpha_i \cdot e^{-\frac{1}{i} \cdot x_j}| \leq \frac{1}{4^q} \cdot (3 + \frac{2}{\pi} \cdot \log(q)) \cdot \sum_{i=1}^{m} |\alpha_i|,$$

for any $j = 1, 2, \ldots, n$. Now, for any given $\epsilon$ and $q > 2 / \log_2(\epsilon)$,

$$|f_j - \sum_{i=\nu_k+1}^{\mu_k-1} \alpha_i \cdot e^{-\frac{1}{i} \cdot x_j}| \leq \epsilon \cdot \sum_{i=1}^{m} |\alpha_i|.$$  

For any $i = 1, 2, \ldots, M - 1$, we will denote by $\nu_i$ the largest integer such that

$$\nu_i < \log_2(\beta_m - x_n) - i - \log_2(\log_2(\frac{1}{\epsilon})).$$

Similarly, for any $i = 1, 2, \ldots, M - 1$, we will denote by $\mu_i$ the smallest integer such that

$$\mu_i > \log_2(\beta_m - x_n) - i - \log_2(\frac{1}{\epsilon}).$$

For any $k = 1, 2, \ldots, M$, we will define the subset $W_k$ of the interval $[0, \beta_n]$ by the formula

$$W_k = \bigcup_{i \geq \mu_k} V_i,$$

and denote by $S_k$ the sum.
\[ S_k = \sum_{i,j \in A_k} \alpha_{ij}. \]  

**Observation 4.3.** It is easy to see that if \( x \in U_i \) and \( \beta \in V_j \) with \( j \leq \nu_i \), then
\[ e^{-r \beta} \leq \epsilon. \]  

Similarly, if \( x \in U_i \) and \( \beta \in V_j \) with \( j \geq \mu_i \), then
\[ |e^{-r \beta} - 1| \leq \epsilon. \]  

Furthermore, for any \( i = 1, 2, \ldots, M - 1 \),
\[ \mu_i - \nu_i \leq 2 \cdot \log_2 \left( \frac{1}{\epsilon} \right). \]  

In other words, given \( x \in U_i \) and \( \beta \in V_j \), one of three possible situations obtains:

a) \( j \leq \nu_i \). In this case, \( e^{-r \beta} \) can be approximated by 0 with a precision \( \epsilon \).

b) \( j \geq \mu_i \). In this case, \( e^{-r \beta} \) can be approximated by 1 with a precision \( \epsilon \).

c) \( \nu_i \leq j \leq \mu_i \). In this case, \( e^{-r \beta} \) can not be approximated by either 0 or 1. However, the total number of indices \( j \) for which this situation obtains is bounded by \( 2 \cdot \log_2 \left( \frac{1}{\epsilon} \right) \), independently of \( x, \beta, \) or \( \epsilon \).

5. Informal Description of the Algorithm.

We will illustrate the idea of the algorithm on a simplified example. Namely, we will assume that \( \beta_i \in U_1 \), i.e.
\[ \frac{x_m}{2} \leq \beta_i \leq \frac{x_m}{2} \]  

for all \( i = 1, 2, \ldots, m \), and \( x_j \in V_1 \), i.e.
\[ \frac{x_n}{2} \leq x_j \leq \frac{x_n}{2} \]  

for all \( j = 1, 2, \ldots, n \).

Consider the function \( e^{-r \beta} \) with \( \beta \in U_1, x \in V_1 \). Fixing \( x \) and viewing \( e^{-r \beta} \) as a function of \( \beta \), we construct its \( q \)-point Chebyshev approximation \( \phi_q(\beta) \) on the interval \( U_1 \). Due to (6),
\[ \phi_q(\beta) = \sum_{j=1}^{q} u_j(x) \cdot e^{-r \beta_j}. \]  

with the functions \( u_j \) defined by (7), and the coefficients \( \beta_j \) defined in Section 4. According to Lemma 2.2,
\[ \left| \phi_q(\beta) - e^{-r \beta} \right| \leq \frac{1}{4^q}. \]
and, given a fixed precision \( \varepsilon \), we can choose \( q = 2 \cdot \log_4 \left( \frac{1}{\varepsilon} \right) \), and in all subsequent calculations replace \( e^{-x^2} \) with \( \psi_k^2(x) \). Combining (48) with the triangle inequality, we obtain the estimate

\[
| \sum_{j=1}^{m} \alpha_j \cdot \psi_k^2(x_j) - \sum_{j=1}^{m} \alpha_j \cdot e^{-x_j^2} | \leq \frac{1}{4q} \cdot \sum_{j=1}^{m} | \alpha_j |	ag{49}
\]

for any \( x \in [0, +\infty] \), and due to (45), the latter can be rewritten in the form

\[
| \sum_{i=1}^{l} c_i \cdot e^{-x_i^2} - \sum_{j=1}^{m} \alpha_j \cdot e^{-x_j^2} | \leq \frac{1}{4q} \cdot \sum_{j=1}^{m} | \alpha_j |,
\]

with the coefficients \( c_1, c_2, \ldots, c_l \) defined by the formula

\[
c_i = \sum_{j=1}^{m} \alpha_j \cdot u_i(x_j).	ag{51}
\]

Now, instead of evaluating (17) at each of the points \( x_i \), we start with evaluating the coefficients \( c_i, i = 1, 2, \ldots, q \), which is, obviously, an order \( O(m \cdot q) \) procedure. After that, we evaluate the expression

\[
\sum_{i=1}^{l} c_i \cdot e^{-x_i^2}
\]

for all \( k = 1, 2, \ldots, n \), which is an order \( O(n \cdot q) \) procedure (evaluating a \( q \)-term expansion at \( n \) points). Thus, the total operation count becomes \( O((n + m) \cdot q) \). Due to (42), in order to obtain a relative accuracy \( \varepsilon \), \( q \) has to be of the order \( \log_4 \left( \frac{1}{\varepsilon} \right) \), and we have reduced the computational complexity of evaluating (17) from \( O(n \cdot m) \) to

\[
O((n + m) \cdot \log_4 \left( \frac{1}{\varepsilon} \right)).	ag{53}
\]

An alternative approach would be to calculate the coefficients \( c_i \) for \( i = 1, 2, \ldots, q \) (order \( m \cdot q \) operations), evaluate the expression

\[
\sum_{i=1}^{q} c_i \cdot e^{-x_i^2}
\]

for all \( k = 1, 2, \ldots, q \) (order \( q^2 \) operations), and interpolate the expression (17) from the Chebyshev nodes \( x_1, x_2, \ldots, x_n \) to the points \( x_1, x_2, \ldots, x_n \) (order \( n \cdot q \) operations). The resulting CPU time estimate in this case is

\[
O((n + m) \cdot q + O(q^2)) = O((n + m) \cdot \log_4 \left( \frac{1}{\varepsilon} \right) + O((\log_4 \left( \frac{1}{\varepsilon} \right))^2)),	ag{55}
\]

which is not substantially different from (53).

When the points \( \beta_1, \beta_2, \ldots, \beta_m \) and \( x_1, x_2, \ldots, x_n \) do not satisfy the inequalities (45), (46), the above approach cannot be used in such straightforward manner. However, for any \( \varepsilon, \eta \in [1, M] \), Lemma 2.2 can be used separately on each of the intervals \( U_i, V_j \), with the results combined to obtain an approximation to (17). This is done in the following section, resulting in an order
\[ O(n + w) \cdot \log \left( \frac{1}{\varepsilon} \right) + O\left( n \cdot \log \left( \frac{1}{\varepsilon} \right)^3 \right) \]

algorithm for evaluating (17) at \( n \) points with a relative precision \( \varepsilon \).

6. Detailed Description of the Algorithm

Algorithm

Stage 1.

Comment (Choose parameters and perform geometrical preprocessing.)

Choose precision \( \varepsilon \) to be achieved. Set \( q = 2 \cdot \log \left( \frac{1}{\varepsilon} \right) \). Construct the intervals \( U_i, V_i \) and the sets \( \widehat{S}_i, \widetilde{S}_i \) with \( i = 1, 2, \ldots, M \).

Stage 2.

Comment (On each of the non-empty intervals \( U_i \), evaluate the coefficients \( a^k_i \) in the expansions (27).)

Step 1.
Comment (Set all coefficients \( a^k_i \) to zero.)

\[
\text{do } k = 1, M - 1, \text{ } \widehat{S}_k \neq \emptyset \\
\quad \text{do } i = 1, q \\
\quad \quad \text{set } a^k_i \text{ to zero.} \\
\quad \text{end do} \\
\text{end do}
\]

Step 2.
Comment (For each \( \zeta, \lambda \) on each of the non-empty intervals \( U_i \), evaluate \( a_i, a^k_{\lambda, \zeta} \) and add it to the \( a^k_{\lambda, \zeta} \).)

\[
\text{do } k = 1, M - 1, \text{ } \widehat{S}_k \neq \emptyset \\
\quad \text{do } i = 1, q \\
\quad \quad \text{do } \lambda \in U_i \\
\quad \quad \quad \text{Evaluate } a^k_{\lambda, \zeta} \text{ via formula (23)} \\
\quad \quad \quad \text{and add the product } \lambda, a^k_{\lambda, \zeta} \text{ to } a^k_{\lambda, \zeta} \\
\quad \quad \text{end do} \\
\quad \text{end do} \\
\text{end do}
\]

Stage 3.
Stage 4.

Comment [For each \( j = 1, 2, \ldots, n \), evaluate \( f_j \) via formula (32).]

\[
\text{do } k = 1, M - 1, \hat{x}_k \neq \emptyset \\
\quad \text{do } i = 1, q \\
\quad \quad \text{evaluate the expression } \hat{f}_i = \sum_{j=k+1}^{n-1} \sum_{l=1}^{q} a_{ij} e^{-\delta_i \epsilon_j} \\
\quad \text{end do} \\
\text{end do}
\]

Stage 5.

Comment [For each \( k = 1, 2, \ldots, M \) and each \( x_i \in V_k \), use Observation 4.3 to evaluate the sum \( \hat{S}_k = \sum_{i \in V_k} \hat{w}_i \hat{a}_j \). Add the result to \( f_i \), concluding the calculation.]

Step 1

Comment [Evaluate \( \hat{S}_1 \).]

\[
\text{set } \hat{S}_1 = \sum_{i \in V_1} \hat{w}_i \hat{a}_j
\]

Step 2

Comment [Evaluate \( \hat{S}_1 \) recursively for \( k = 2, 3, \ldots, M \).]

\[
\text{do } k = 2, M, \hat{x}_k \neq \emptyset \\
\quad \text{evaluate } \hat{S}_k \text{ via the formula } \hat{S}_k = \hat{S}_{k-1} + \sum_{i \in V_k} \hat{w}_i \hat{a}_j \\
\text{end do}
\]

Step 3

Comment [For all \( k = 1, 2, \ldots, M \) and all \( i \) such that \( x_i \in V_k \), add \( \hat{S}_k \) to \( f_i \), concluding the calculation.]
do $k = 1, M, \delta_k \neq \emptyset$
    do $x_i \in V_k$
        add $S_k$ to $f_i$
    end do
end do

7. Complexity analysis

<table>
<thead>
<tr>
<th>Stage number</th>
<th>Operation count</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage 1</td>
<td>$O(n + m)$</td>
<td>Each of the points $\beta_1, \beta_2, \ldots, \beta_m$ is assigned to a single interval $U_k$. Each of the points $x_1, x_2, \ldots, x_n$ is assigned to a single interval $V_i$.</td>
</tr>
<tr>
<td>Step 1</td>
<td>$O(M \cdot q)$</td>
<td>Each of the coefficients $u_i^k$, with $k = 1, 2, \ldots, M$, and $i = 1, 2, \ldots, q$ is set to zero.</td>
</tr>
<tr>
<td>Step 2</td>
<td>$O(m \cdot q^2)$</td>
<td>Each of the points $\beta_1, \beta_2, \ldots, \beta_m$ contributes to the coefficients $u_i^k$, with $j = 1, 2, \ldots, q$, and evaluating each of the coefficients $u_i^k$ requires order $q$ work (see (23)).</td>
</tr>
<tr>
<td>Stage 3</td>
<td>$O(M \cdot q^2 (\frac{1}{t}))$</td>
<td>The sum (31) has to be evaluated at $q$ nodes $x_1^k, x_2^k, \ldots, x_{\nu_k}^k$ on each of non-empty intervals $V_1, V_2, \ldots, V_M$, and on the $k-th$ interval, it contains $\mu_k - \nu_k$ terms. However, due to (44), $\mu_k - \nu_k \leq t$ for all $k = 1, 2, \ldots, M$.</td>
</tr>
<tr>
<td>Stage 4</td>
<td>$O(n \cdot q^2)$</td>
<td>The expression (32) has to be evaluated for each of the points $x_1, x_2, \ldots, x_n$, and evaluating each of the coefficients $v_{i,j}^k$ requires order $q$ work (see (33)).</td>
</tr>
<tr>
<td>Stage 5</td>
<td>$O(m)$</td>
<td>The sum $S_1 = \sum_{i \in T_1} \alpha_i$ contains no more than $m$ terms.</td>
</tr>
<tr>
<td>Step 1</td>
<td>$O(m)$</td>
<td>The total number of non-empty intervals $V_i$ is bounded by $n$, and the total number of coefficients $\alpha_i$ is bounded by $m$.</td>
</tr>
</tbody>
</table>
Step 3 $O(n)$ Each of the numbers $f_j$ is amended once.

Summing up the CPU times for all stages above, we obtain the following time estimate:

$$T_{load} = a + m = b \cdot n + c \cdot m \cdot q^2 = d \cdot n + q^2 + e \cdot \overline{M} \cdot q^2 \cdot \log(\frac{1}{\varepsilon}).$$  \hspace{1cm} (57)

where the coefficients $a, b, c, d, e$ depend on the computer system, language, implementation, etc. However, $\overline{M} \leq m$, and $\varepsilon \approx \log(\frac{1}{\varepsilon})$, and the estimate (57) assumes the form

$$T_{load} = a + m = \log(\frac{1}{\varepsilon}) \cdot b \cdot n + \log(\frac{1}{\varepsilon}) \cdot c.$$

The estimate (58) is independent of the locations of the points $\alpha_i, x_i$ in $R^d$, and does not depend on any precomputed data. The following two observations reduce it to

$$T_{load} = O(m + m \cdot \log(\frac{1}{\varepsilon}))$$

(59)

for many problems of practical interest.

**Observation 7.1.** The term $b \cdot m \cdot q^2$ in (58) is associated with the Stage 3 of the algorithm and the grossly pessimistic estimate

$$\overline{M} \leq M \leq m.$$  \hspace{1cm} (60)

According to (20),

$$M - 1 \leq \log_2(\frac{\ell_m}{x_m}) = \log_2(\ell_m) + \log_2(x_m) + \log_2(\frac{1}{\varepsilon}).$$

(61)

Normally, when calculations are performed on a physical computer, the exponential in the binary representation of a real number is bounded, and we will denote this bound by $A$. It immediately follows that in all cases,

$$\overline{M} \leq M \leq 3 \cdot \log_2[A] + 1.$$  \hspace{1cm} (62)

and the estimate (57) becomes

$$T_{load} = a + m + b \cdot n + c \cdot m \cdot q^2 + d \cdot n + q^2 + e \cdot \log(A) \cdot q^2 \cdot \log(\frac{1}{\varepsilon}).$$

(63)

**Observation 7.2.** The terms $c \cdot m \cdot q^2$ and $d \cdot n \cdot q^2$ in (57) are associated with the Stages 2 and 4 of the algorithm, and with the fact that in order to evaluate each of the coefficients $a_{ij}$ (or $b_{ij}$), a $q - 1$ term product of the form (24) (or (33)) has to be evaluated. Obviously, the coefficients $a_{ij}$ depend only on the distribution of the points $\alpha_i$ and not on that of $x_i$ or $\alpha_i$. Similarly, the coefficients $b_{ij}$ depend only on the distribution of the points $x_i$ and not on that of $\alpha_i$ or $\alpha_i$. Therefore, for fixed distributions of $\alpha_i$ and $x_i$, the coefficients $a_{ij}; b_{ij}$ can be precomputed and stored, reducing the total CPU time estimate to
\[ T_{\text{total}} \approx a \cdot m \cdot q + b \cdot n \cdot q + c \cdot \log(A) \cdot q^2 \cdot \log(\frac{1}{r}). \]  

(64)

However, \( q \approx \log(\frac{1}{2}) \), and \( \log(A) \) is fixed for given computer system and language. Thus, when \( m, n \rightarrow \infty \),

\[ T_{\text{total}} \approx (a \cdot m + b \cdot n) \cdot q \]  

(65)

8. Numerical Results

A computer program has been written implementing the algorithm of this paper. The calculation is performed in two stages, each implemented by a separate subroutine. During the initialization stage, the coefficients \( a_{i,j}^k, c_{i,j}^k \) are evaluated for given distributions of points \( x_1, x_2, \ldots, x_m, x_1, x_2, \ldots, x_n \) (see Observation 7.2). During the second stage, the sums (17) are evaluated for a given set of weights \( \omega_1, \omega_2, \ldots, \omega_m \).

**Remark 8.1.** It is clear from Tables 1, 3, 5 that the first stage (initialization) tends to be several times more expensive than the second (evaluation). However, in most applications the algorithm has to be initialized once, with subsequent repeated evaluation of the sums (17) for varying sets of weights \( \omega_1, \omega_2, \ldots, \omega_m \). This situation is similar to that encountered for the Fast Fourier Transformation.

The program has been applied to a variety of situations, and three such examples are presented in this section, with the computations performed on a VAX-6000 computer. In each case, we performed the calculations in three ways: via the algorithm of the present paper in single precision arithmetic, directly in single precision arithmetic, and directly in double precision arithmetic. The first two calculations were used to compare the speed and precision of the algorithm with that of the direct calculation. The direct evaluation of the field in double precision was used as a standard for comparing the accuracies of the first two calculations. In all cases, we set \( \epsilon = 10^{-8} \), and

\[ m = n = 10 \cdot 2^k. \]  

(66)

with \( k \) varying from 1 to 8.

Tables 1, 3, 5 contain the CPU timings for the examples 1, 2, 3 respectively. Following is a detailed description of the entries in these Tables.

- \( n \) - the number of points at which the sum (1) is being evaluated.
- \( T_{\text{init}} \) - the initialization time of the algorithm.
- \( T_{\text{dir}} \) - the CPU time required by the algorithm once it has been initialized.
- \( T_{\text{HDR}} \) - the CPU time required by the direct calculation.

Tables 2, 4, 6 contain the accuracies for the examples 1, 2, 3 respectively. In the description of the entries of these tables below, \( S_k \) denotes the sum (1) at the point \( x_k \) as evaluated directly in double precision. \( S_{\text{HDR}}^k \) denotes the sum (1) at the point \( x_k \) as evaluated directly in single precision, and \( S_{\text{dir}}^k \) denotes the sum (1) at the point \( x_k \) as evaluated in single precision via the
algorithm of the present paper. Following is a detailed description of the entries in the Tables 2, 4, 6.

- $n$: the number of points at which the sum \((1)\) is being evaluated.
- $\delta_{alg}^{max}$: the maximum error produced by the algorithm at any point. It is defined by the formula,
  \[
  \delta_{alg}^{max} = \max_{1 \leq k \leq n} | S_k^{alg} - S_k | .
  \]  

- $\epsilon_{alg}^{max}$: the maximum error produced by the direct calculation at any point. It is defined by the formula
  \[
  \epsilon_{alg}^{max} = \max_{1 \leq k \leq n} \frac{| S_k^{alg} - S_k |}{S_k} .
  \]

- $\epsilon_{alg}^{max, rel}$: the maximum relative error produced by the algorithm at any point. It is defined by the formula
  \[
  \epsilon_{alg}^{max, rel} = \max_{1 \leq k \leq n} \frac{S_k^{alg} - S_k}{S_k} .
  \]

- $\epsilon_{alg}$: the relative error as defined in Section 3 as produced by the algorithm. It is given by the formula
  \[
  \epsilon_{alg} = \sum_{k=1}^{n} \frac{| S_k^{alg} - S_k |}{\sum_{k=1}^{n} | S_k |} .
  \]

- $\epsilon_{dir}$: the relative error as defined in Section 3 as produced by the direct calculation. It is given by the formula
  \[
  \epsilon_{dir} = \sum_{k=1}^{n} \frac{| S_k^{dir} - S_k |}{\sum_{k=1}^{n} | S_k |} .
  \]

Following is a detailed description of the three examples.

**Example 1.** In this example, the points $\beta_1, \beta_2, \ldots, \beta_m$ and $x_1, x_2, \ldots, x_n$ were defined by the formulae

- $\beta_i = \frac{5}{m - 1} \cdot (i - 1)$, \hspace{1cm} (73)
- $x_k = \frac{5}{n - 1} \cdot (k - 1)$, \hspace{1cm} (74)
and the weights $\alpha_1, \alpha_2, \ldots, \alpha_m$ were generated randomly on the interval $[0, 1]$. Here, by "direct algorithm" we mean a straightforward implementation of the formula (17). The results of this set of experiments are summarized in Tables 1, 2.

**Example 2.** In this example, the points $\beta_1, \beta_2, \ldots, \beta_m$ and $x_1, x_2, \ldots, x_n$ were generated randomly on the interval $[0, 5]$, and the weights $\alpha_1, \alpha_2, \ldots, \alpha_m$ were generated randomly on the interval $[0, 1]$. Again, by "direct algorithm" we mean a straightforward implementation of the formula (17). The results of this set of experiments are summarized in Tables 3, 4.

**Example 3.** Here, we evaluate a polynomial of order $n$ at a collection of randomly generated points on the interval $[0, 1]$. The coefficients of the $n$-th order polynomial are randomly distributed on the interval $[0, 1]$. In this example, the direct evaluation of the polynomials is performed via the Horner’s rule (see, for example, [3]), and the algorithm of this paper is applied via the formula (1). The results of this set of experiments are presented in Tables 5, 6.

The following observations can be made from the Tables 1-6, and are in agreement with the results of our more extensive experiments.

1. In all cases, the accuracy produced by the algorithm of the present paper is comparable to that obtained by the direct calculation. For large $n$, the algorithm tends to be slightly more accurate.

2. The CPU times and accuracies produced by the algorithm are virtually independent of the distributions of points $\alpha_i, \beta_i, x_k$ in $R^1$.

3. When used for evaluating expressions of the form (17), the algorithm becomes faster than the direct calculation at $n = m \leq 20$, if the initialization time is ignored. If we include the initialization time, the break-even point is between $n = m = 40$ and $n = m = 60$.

4. When used for evaluating polynomials, the algorithm becomes faster than the direct calculation at roughly $n = m = 40$, if the initialization time is ignored. If we include the initialization time, the break-even point is roughly $n = m = 300$. 


References


Table 1

Example 1: Timings

<table>
<thead>
<tr>
<th>$n$</th>
<th>$T_{ini}$</th>
<th>$T_{alg}$</th>
<th>$T_{dir}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.0112</td>
<td>0.0015</td>
<td>0.0081</td>
</tr>
<tr>
<td>40</td>
<td>0.0369</td>
<td>0.0042</td>
<td>0.0318</td>
</tr>
<tr>
<td>80</td>
<td>0.0802</td>
<td>0.0092</td>
<td>0.1278</td>
</tr>
<tr>
<td>160</td>
<td>0.136</td>
<td>0.0165</td>
<td>0.5202</td>
</tr>
<tr>
<td>320</td>
<td>0.218</td>
<td>0.0283</td>
<td>2.069</td>
</tr>
<tr>
<td>640</td>
<td>0.333</td>
<td>0.0468</td>
<td>8.368</td>
</tr>
<tr>
<td>1280</td>
<td>0.484</td>
<td>0.0784</td>
<td>33.25</td>
</tr>
<tr>
<td>2560</td>
<td>0.727</td>
<td>0.137</td>
<td>133.58</td>
</tr>
</tbody>
</table>

Table 2

Example 1: Accuracies

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\delta_{\text{max},\text{rel}}^{\text{alg}}$</th>
<th>$\delta_{\text{max},\text{rel}}^{\text{dir}}$</th>
<th>$\delta_{\text{max},\text{rel}}^{\text{rel}}$</th>
<th>$\delta_{\text{rel}}^{\text{alg}}$</th>
<th>$\delta_{\text{rel}}^{\text{dir}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>.412E-06</td>
<td>.638E-06</td>
<td>.383E-06</td>
<td>.243E-06</td>
<td>.359E-07</td>
</tr>
<tr>
<td>40</td>
<td>.179E-05</td>
<td>.219E-05</td>
<td>.459E-06</td>
<td>.418E-06</td>
<td>.783E-07</td>
</tr>
<tr>
<td>80</td>
<td>.408E-05</td>
<td>.688E-05</td>
<td>.623E-06</td>
<td>.671E-06</td>
<td>.100E-06</td>
</tr>
<tr>
<td>160</td>
<td>.138E-04</td>
<td>.262E-04</td>
<td>.825E-06</td>
<td>.116E-06</td>
<td>.173E-06</td>
</tr>
<tr>
<td>320</td>
<td>.378E-04</td>
<td>.873E-04</td>
<td>.597E-06</td>
<td>.195E-05</td>
<td>.238E-06</td>
</tr>
<tr>
<td>640</td>
<td>.922E-04</td>
<td>.231E-03</td>
<td>.103E-05</td>
<td>.258E-05</td>
<td>.279E-06</td>
</tr>
<tr>
<td>1280</td>
<td>.273E-03</td>
<td>.740E-03</td>
<td>.841E-06</td>
<td>.473E-05</td>
<td>.420E-06</td>
</tr>
<tr>
<td>2560</td>
<td>.522E-03</td>
<td>.233E-02</td>
<td>.880E-06</td>
<td>.886E-05</td>
<td>.407E-06</td>
</tr>
</tbody>
</table>
### Table 3

Example 2: Timings

<table>
<thead>
<tr>
<th>n</th>
<th>$T_{init}$</th>
<th>$T_{alg}$</th>
<th>$T_{dir}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.0097</td>
<td>0.0011</td>
<td>0.0083</td>
</tr>
<tr>
<td>40</td>
<td>0.0275</td>
<td>0.0033</td>
<td>0.0332</td>
</tr>
<tr>
<td>80</td>
<td>0.0768</td>
<td>0.0089</td>
<td>0.1328</td>
</tr>
<tr>
<td>160</td>
<td>0.126</td>
<td>0.0157</td>
<td>0.536</td>
</tr>
<tr>
<td>320</td>
<td>0.210</td>
<td>0.0271</td>
<td>2.12</td>
</tr>
<tr>
<td>640</td>
<td>0.326</td>
<td>0.0455</td>
<td>8.50</td>
</tr>
<tr>
<td>1280</td>
<td>0.497</td>
<td>0.0784</td>
<td>34.12</td>
</tr>
<tr>
<td>2560</td>
<td>0.698</td>
<td>0.1351</td>
<td>138.34</td>
</tr>
</tbody>
</table>

### Table 4

Example 2: Accuracies

<table>
<thead>
<tr>
<th>n</th>
<th>$\Delta^{max}_{alg}$</th>
<th>$\Delta^{max}_{dir}$</th>
<th>$\Delta^{max,rel}_{alg}$</th>
<th>$\Delta^{max,rel}_{dir}$</th>
<th>$\delta^{rel}_{alg}$</th>
<th>$\delta^{rel}_{dir}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>3.12E-06</td>
<td>1.64E-06</td>
<td>1.60E-06</td>
<td>1.92E-06</td>
<td>2.68E-07</td>
<td>1.41E-07</td>
</tr>
<tr>
<td>40</td>
<td>1.09E-05</td>
<td>2.70E-05</td>
<td>4.05E-06</td>
<td>3.16E-06</td>
<td>5.01E-07</td>
<td>1.24E-06</td>
</tr>
<tr>
<td>80</td>
<td>3.54E-05</td>
<td>3.64E-05</td>
<td>6.58E-06</td>
<td>8.60E-06</td>
<td>8.32E-07</td>
<td>8.57E-07</td>
</tr>
<tr>
<td>160</td>
<td>8.35E-05</td>
<td>1.61E-04</td>
<td>8.60E-06</td>
<td>8.45E-06</td>
<td>1.00E-06</td>
<td>1.94E-06</td>
</tr>
<tr>
<td>320</td>
<td>1.98E-04</td>
<td>2.98E-04</td>
<td>9.74E-06</td>
<td>1.58E-06</td>
<td>1.15E-06</td>
<td>1.73E-06</td>
</tr>
<tr>
<td>640</td>
<td>6.06E-04</td>
<td>1.28E-03</td>
<td>8.24E-06</td>
<td>2.88E-05</td>
<td>1.85E-06</td>
<td>3.89E-06</td>
</tr>
<tr>
<td>1280</td>
<td>3.36E-03</td>
<td>6.57E-03</td>
<td>9.14E-06</td>
<td>4.23E-05</td>
<td>2.07E-06</td>
<td>1.00E-05</td>
</tr>
<tr>
<td>2560</td>
<td>4.51E-03</td>
<td>1.49E-02</td>
<td>8.27E-06</td>
<td>7.99E-05</td>
<td>3.48E-06</td>
<td>1.15E-05</td>
</tr>
</tbody>
</table>
### Table 5

Example 3: Timings

<table>
<thead>
<tr>
<th>n</th>
<th>$T_{init}$</th>
<th>$T_{alg}$</th>
<th>$T_{dir}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.0135</td>
<td>0.0015</td>
<td>0.0013</td>
</tr>
<tr>
<td>40</td>
<td>0.0435</td>
<td>0.0052</td>
<td>0.0047</td>
</tr>
<tr>
<td>80</td>
<td>0.0948</td>
<td>0.0109</td>
<td>0.0179</td>
</tr>
<tr>
<td>160</td>
<td>0.1327</td>
<td>0.0172</td>
<td>0.0729</td>
</tr>
<tr>
<td>320</td>
<td>0.222</td>
<td>0.0286</td>
<td>0.2779</td>
</tr>
<tr>
<td>640</td>
<td>0.306</td>
<td>0.0445</td>
<td>1.101</td>
</tr>
<tr>
<td>1280</td>
<td>0.422</td>
<td>0.0718</td>
<td>4.54</td>
</tr>
<tr>
<td>2560</td>
<td>0.664</td>
<td>0.1322</td>
<td>18.37</td>
</tr>
</tbody>
</table>

### Table 6

Example 3: Accuracies

<table>
<thead>
<tr>
<th>n</th>
<th>$\delta_{max}^{alg}$</th>
<th>$\delta_{max}^{dir}$</th>
<th>$\delta_{max,rel}^{alg}$</th>
<th>$\delta_{max,rel}^{dir}$</th>
<th>$\delta_{rel}^{alg}$</th>
<th>$\delta_{rel}^{dir}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>.772E-06</td>
<td>.260E-05</td>
<td>.301E-06</td>
<td>.305E-06</td>
<td>.673E-07</td>
<td>.227E-06</td>
</tr>
<tr>
<td>40</td>
<td>.218E-05</td>
<td>.275E-05</td>
<td>.396E-06</td>
<td>.337E-06</td>
<td>.955E-07</td>
<td>.121E-06</td>
</tr>
<tr>
<td>80</td>
<td>.518E-05</td>
<td>.554E-05</td>
<td>.528E-06</td>
<td>.210E-06</td>
<td>.127E-06</td>
<td>.136E-06</td>
</tr>
<tr>
<td>160</td>
<td>.615E-05</td>
<td>.462E-05</td>
<td>.671E-06</td>
<td>.304E-06</td>
<td>.766E-07</td>
<td>.576E-07</td>
</tr>
<tr>
<td>320</td>
<td>.103E-04</td>
<td>.232E-05</td>
<td>.851E-06</td>
<td>.387E-06</td>
<td>.646E-07</td>
<td>.146E-06</td>
</tr>
<tr>
<td>640</td>
<td>.224E-04</td>
<td>.295E-04</td>
<td>.832E-06</td>
<td>.422E-06</td>
<td>.680E-07</td>
<td>.894E-07</td>
</tr>
<tr>
<td>1280</td>
<td>.615E-04</td>
<td>.360E-04</td>
<td>.745E-06</td>
<td>.120E-05</td>
<td>.949E-07</td>
<td>.555E-06</td>
</tr>
<tr>
<td>2560</td>
<td>.757E-04</td>
<td>.120E-02</td>
<td>.862E-06</td>
<td>.191E-05</td>
<td>.590E-07</td>
<td>.932E-06</td>
</tr>
</tbody>
</table>