An algorithm is presented for the rapid evaluation of expressions of the form $\sum_{k=1}^{N} \alpha_k \cdot e^{i\omega_k x}$, at the points $x_1, \ldots, x_m$ which are arbitrarily distributed in the interval $[-\pi, \pi]$. The algorithm is based on trigonometric interpolation techniques and in most cases of practical interest, evaluation of the above sum at m points requires $O(N \cdot \log N + m)$ operations. This problem can be viewed as a generalization of the Discrete Fourier Transform and the scheme of the paper is widely applicable to many problems encountered in mathematics, science and engineering.

A Fast Algorithm for the Evaluation of Trigonometric Series

A. Dutt

Research Report YALEU/DCS/RR-841
January 1991

The author was supported by the National Science Foundation under Grant NSF-DMS9012751.
Approved for public release: distribution is unlimited.
Keywords: *Trigonometric Series, Fourier Analysis, Interpolation, Approximation Theory*

1 Introduction

Fourier techniques have been a popular analytical tool in the study of physics and engineering for more than two centuries. A reason for the usefulness of such techniques is that the trigonometric functions are eigenfunctions of the differentiation operator and can be effectively used to model solutions of differential equations which arise in the fields mentioned above.

With the arrival of digital computers, it became theoretically possible to calculate the Fourier series and Fourier transform of a function numerically. This was unrealistic in practice however owing to the prohibitive complexity of even modestly sized problems. A major breakthrough in overcoming this difficulty was the development of the Fast Fourier Transform (FFT) algorithm in the 1960s which established Fourier analysis as a useful and practical numerical tool.

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

$$\sum_{k=1}^{N} \alpha_k \cdot e^{i\omega_k x}, \tag{1}$$

where $\omega_k \in \mathbf{R}$ and $\alpha_k \in \mathbf{C}$ for $k = 1, \ldots, N$, and $x \in [-\pi, \pi]$.

The performance of the scheme is dependent on the distribution of frequencies, and we let n denote the smallest power of 2 such that $\omega_k \in [-\frac{n}{2}, \frac{n}{2}]$ for all k. Our algorithm then requires a number of arithmetic operations proportional to

$$n \cdot \log_2 n + (N + m) \cdot \log_e \left(\frac{1}{\varepsilon}\right) \tag{2}$$

to evaluate the sum at m arbitrary points in $[-\pi, \pi]$ where ε is our required accuracy for the results. In many cases of practical interest it turns out that $n \sim N$ and the operation count reduces to

$$O(N \cdot \log_2 N + (N + m) \cdot \log_e \left(\frac{1}{\varepsilon}\right)) \tag{3}$$

giving us a significant improvement over the $N \cdot m$ operations required for the direct evaluation.

The problem as described above can be viewed as a generalization of the discrete Fourier transform, which is defined by the equations

$$F_j = \frac{1}{N} \sum_{k=0}^{N-1} f_k \cdot e^{-2\pi i j k/N}, \qquad j = 0, \ldots, N-1 \tag{4}$$

for a given sequence of N numbers $f_k \in \mathbf{C}$. In this special case of (1), the frequencies ω_k are integers and the points x_j are equally spaced in $[0, 2\pi]$. Under these conditions, the matrix representation of the Fourier kernel has a simple structure which can be exploited, and the well known and highly efficient Classical FFT algorithm provides a way of evaluating the N sums (4) in $O(N \cdot \log N)$ arithmetic operations with a very small constant as opposed to $O(N^2)$ operations for the direct evaluation. There are in fact a variety of such schemes, but all are purely algebraic in nature.

In the more general case, however, the underlying structure of the matrix is not so easily exploitable. The algorithm of this paper makes use of some results from approximation theory coupled with existing FFT techniques to give a very versatile solution for the generalized problem.

The plan of the paper is as follows. We start in section 2 with some results from analysis and approximation theory which are used in the design of the algorithm. An exact statement of the problem in section 3 is then followed by an informal description of the algorithm in section 4. In section 5 we introduce some notation which is used in a more detailed description of the algorithm in section 6. In section 7 we describe some modifications to the scheme to improve its performance for certain frequency distributions. The results of our numerical experiments are presented in section 8 to illustrate the behavior and performance of the algorithm. Finally, section 9 lists some generalizations of the method and some conclusions.

2 Mathematical and Numerical Preliminaries

2.1 Analytical Tools

The following well-known lemma gives us the convergence rate of Fourier series and can be found in slightly different forms in, for example, [2, 4].

Lemma 2.1 *Let $f : \mathbf{R} \mapsto \mathbf{C}$ be a 2π-periodic function with a Fourier series expansion*

$$f(\theta) = \sum_{k=-\infty}^{\infty} c_k e^{ik\theta} \tag{5}$$

If f has a piecewise continuous p-th derivative then

$$c_k = O\left(\frac{1}{k^{p+1}}\right). \tag{6}$$

Another standard result is the translation-invariance of a Gaussian integral (see e.g.[1]).

Lemma 2.2 *Let $b, u, v \in \mathbf{R}$. Then*

$$\int_{-\infty}^{\infty} e^{-b(x-u-iv)^2} dx = \int_{-\infty}^{\infty} e^{-bx^2} dx = \sqrt{\frac{\pi}{b}}. \tag{7}$$

2.2 Relevant facts from Approximation Theory

The principal numerical tool of this paper is based on one simple observation to which this section is devoted.

Consider the 2π-periodic function given by e^{icx} on the interval $[-\pi, \pi]$ with non-integer frequency, c. This is discontinuous at odd integer multiples of π and from lemma 2.1 its spectrum decays like $1/k$. The Fourier series of such a function thus has poor convergence properties. A standard technique for substantially improving the convergence rate is to multiply

the function by a Gaussian bell e^{-bx^2} where b is chosen such that $e^{-b\pi^2} = \epsilon$ for a specified error tolerance $\epsilon > 0$, giving us

$$b = \frac{\log_e(1/\epsilon)}{\pi^2}. \tag{8}$$

The effect of this is to smoothly force the discontinuity to be small, thereby increasing the number of continuous derivatives at the ends of the interval to a precision ϵ while preserving the continuity of all derivatives away from these ends.

We consider now the properties of the new function,

$$G(x; b, c) = e^{-bx^2} \cdot e^{icx}. \tag{9}$$

G still has a discontinuity of size $\sim \epsilon$ at $x = \pm\pi$. Let us define a slightly perturbed function by

$$\begin{aligned} \tilde{G}(x) &= (e^{-bx^2} - \epsilon) \cdot e^{icx} && (10) \\ &= G(x) - \epsilon \cdot e^{icx}. && (11) \end{aligned}$$

Then $\tilde{G}(x)$ has the following properties:

- $|\tilde{G}(x) - G(x)| = \epsilon$ for $x \in [-\pi, \pi]$
- $\tilde{G}$ is continuous and $\tilde{G}(-\pi) = \tilde{G}(\pi) = 0$
- $\tilde{G}$ has a piecewise continuous first derivative with a single discontinuity of size $\sim \epsilon$ at $\pm\pi$.

By lemma 2.1, we can write

$$\tilde{G}(x) = \sum_{k=-\infty}^{\infty} g_k e^{ikx} \tag{12}$$

where the g_k decay like $O(1/k^2)$.

This suggests that if we keep only those coefficients which are bigger than ϵ the truncation error incurred will be $O(\epsilon)$. As $\tilde{G}$ is smooth except for a small discontinuity in its derivative, we expect too that the number of such coefficients is small.

A precise statement of this fact along with more detailed error analysis are contained in the following theorems.

Theorem 2.3 *Let $\tilde{G}(x)$ be defined by (10) for a given tolerance $\epsilon > 0$. Then,*

$$\tilde{G}(x) = \sum_{k=-\infty}^{\infty} h_k e^{ikx} + O(\epsilon) \tag{13}$$

where

$$h_k = \frac{1}{2\sqrt{b\pi}} e^{-(c-k)^2/4b}. \tag{14}$$

Proof. The Fourier coefficients for $\tilde{G}$ are given analytically by the formula

$$2\pi g_k = \int_{-\pi}^{\pi} \tilde{G}(x)e^{-ikx}dx \tag{15}$$

$$= \int_{-\infty}^{\infty} G(x)e^{-ikx}dx - \int_{-\infty}^{-\pi} G(x)e^{-ikx}dx - \int_{\pi}^{\infty} G(x)e^{-ikx}dx - \varepsilon \cdot \int_{-\pi}^{\pi} e^{icx}e^{-ikx}dx \tag{16}$$

The first integral can be obtained explicitly by completing the square and using Lemma 2.2 as follows

$$\begin{aligned}\int_{-\infty}^{\infty} G(x)e^{-ikx}dx &= \int_{-\infty}^{\infty} e^{-bx^2+icx-ikx}dx \\ &= \int_{-\infty}^{\infty} e^{-b(x-i(c-k)/2)^2-(c-k)^2/4b}dx \\ &= \sqrt{\frac{\pi}{b}} \cdot e^{-(c-k)^2/4b}.\end{aligned}$$

We also have

$$\begin{aligned}&\int_{-\infty}^{-\pi} G(x)e^{-ikx}dx + \int_{\pi}^{\infty} G(x)e^{-ikx}dx \\ &= \int_{-\infty}^{-\pi} e^{-bx^2}e^{i(c-k)x}dx + \int_{\pi}^{\infty} e^{-bx^2}e^{i(c-k)x}dx \\ &= 2\int_{\pi}^{\infty} e^{-bx^2}\cos((c-k)x)dx \\ &= \frac{2}{c-k}\left[e^{-bx^2}\sin((c-k)x)\right]_{\pi}^{\infty} + \frac{4b}{c-k}\int_{\pi}^{\infty} xe^{-bx^2}\sin((c-k)x)dx \\ &= -\frac{2\varepsilon}{c-k}\sin((c-k)\pi) + \varepsilon \cdot O\left(\frac{1}{(c-k)^2}\right).\end{aligned}$$

Finally,

$$\varepsilon \cdot \int_{-\pi}^{\pi} e^{icx}e^{-ikx}dx = \frac{2\varepsilon}{c-k}\sin((c-k)\pi).$$

Substituting for these integrals in the formula (16), we find that the $1/(c-k)$ terms cancel, and that

$$g_k = h_k + \varepsilon \cdot O\left(\frac{1}{(c-k)^2}\right). \tag{17}$$

where h_k are defined by

$$h_k = \frac{1}{2\pi}\int_{-\infty}^{\infty} G(x)e^{-ikx}dx \tag{18}$$

$$= \frac{1}{2\sqrt{b\pi}}e^{-(c-k)^2/4b}. \tag{19}$$

Substituting now for g_k in (12),

$$\tilde{G}(x) = \sum_{k=-\infty}^{\infty} h_k e^{ikx} + R(x) \tag{20}$$

where

$$R(x) = \varepsilon \sum_{k=-\infty}^{\infty} \frac{r_k}{(c-k)^2} e^{ikx} \tag{21}$$

for some set of coefficients r_k.

The size of this error can be bounded as follows

$$|R(x)| \leq \varepsilon r \sum_{k=-\infty}^{\infty} \frac{1}{(c-k)^2} < \varepsilon r' \tag{22}$$

where r, r' are real constants.

We can thus rewrite (20) as

$$\tilde{G}(x) = \sum_{k=-\infty}^{\infty} h_k e^{ikx} + O(\varepsilon). \tag{23}$$

□

As a consequence of the fact that $|\tilde{G} - G| = \varepsilon$ and the result of the previous theorem, we have

Corollary 2.4 *Let $G(x) = e^{-bx^2} e^{icx}$ and h_k be defined by (14) for a given tolerance $\varepsilon > 0$. Then*

$$G(x) = \sum_{k=-\infty}^{\infty} h_k e^{ikx} + O(\varepsilon) \tag{24}$$

for $x \in [-\pi, \pi]$.

The coefficients h_k themselves look like a Gaussian with a peak at the nearest integer to c which we denote by k_c. They decay superalgebraically, i.e. faster than any finite power of $1/k$. Suppose we now keep only the $q+1$ largest terms in the series, where

$$e^{-((q+1)/2)^2/4b} < \varepsilon \tag{25}$$

Rearranging and substituting for b from (8), we get

$$q \geq 2 \cdot \left\lceil \frac{2}{\pi} \cdot \log_e \left(\frac{1}{\varepsilon} \right) \right\rceil. \tag{26}$$

Note that the bandwidth q is independent of the frequency c.

The next theorem gives an estimate for the truncation error of the series under these conditions.

Theorem 2.5 *Let $G(x) = e^{-bx^2}e^{icx}$. Then, in the notation of this section,*

$$G(x) = \sum_{k=k_c-q/2}^{k_c+q/2} h_k e^{ikx} + O(\varepsilon) \tag{27}$$

for $x \in [-\pi, \pi]$.

Proof. The truncation error for the series is given by

$$\begin{aligned}
\left| \sum_{k=k_c+q/2+1}^{\infty} h_k e^{ikx} + \sum_{k=-\infty}^{k_c-q/2-1} h_k e^{ikx} \right| &\leq \frac{1}{2\sqrt{b\pi}} \sum_{k=k_c+q/2+1}^{\infty} (e^{-(c-k)^2/4b} + e^{-(c+k)^2/4b}) \\
&< \frac{1}{\sqrt{b\pi}} \sum_{k=q/2+1}^{\infty} e^{-k^2/4b} \\
&= \frac{1}{\sqrt{b\pi}} \sum_{k=0}^{\infty} e^{-(k+q/2+1)^2/4b} \\
&< \frac{1}{\sqrt{b\pi}} e^{-(q/2+1)^2/4b} \sum_{k=0}^{\infty} e^{-k^2/4b} \\
&< \frac{\varepsilon}{\sqrt{b\pi}} \left(1 + \int_0^{\infty} e^{-x^2/4b} dx \right) \\
&= \frac{\varepsilon}{\sqrt{b\pi}} \left(1 + \frac{\sqrt{4b\pi}}{2} \right) \\
&= \varepsilon \cdot \left(\sqrt{\frac{\pi}{\log_e(1/\varepsilon)}} + 1 \right)
\end{aligned}$$

□

In summary, any function of the form $e^{-bx^2} \cdot e^{icx}$ can be accurately represented using a small number of terms of the form e^{ikx}.

Multiplying both sides of the formula (27) by e^{bx^2}, we obtain an approximation to the original function on the subinterval $[-\frac{\pi}{2}, \frac{\pi}{2}]$:

$$e^{icx} = e^{bx^2} \cdot \sum_{k=k_c-q/2}^{k_c+q/2} h_k e^{ikx} + O(\kappa \cdot \varepsilon) \tag{28}$$

where

$$\kappa = e^{b\pi^2/4} = \frac{1}{\sqrt[4]{\varepsilon}} \tag{29}$$

is the maximum size of e^{bx^2} on the subinterval.

The worst-case estimates obtained above provide us with rather pessimistic upper bounds for the truncation errors.

We estimated the actual error in approximating e^{icx} using terms of the form $e^{bx^2}e^{ikx}$ for different choices of tolerance ϵ and number of terms q. The results are presented in Table 1. We only tested the approximation for the constant function $f(x) = 1$ corresponding to the frequency $c = 0$ as the spectra for all other frequencies are translates of the spectrum for this one. The error estimates are thus valid for all real frequencies c. The approximation

$$H(x) = e^{bx^2} \cdot \sum_{k=-q/2}^{q/2} h_k e^{ikx} \tag{30}$$

to 1 with h_k defined as in equation (14) was computed at $n = 1000$ equally spaced nodes x_k in $[-\pi/2, \pi/2]$ and the entries in the table are defined as follows:

- The numbers of terms, q, we used are given by

$$q = 2 \cdot \left\lceil \frac{2}{\pi} \cdot \log_e \left(\frac{1}{\epsilon} \right) \right\rceil + 4 \cdot i \tag{31}$$

  for $i = 0, 1, 2$.

- E^∞ is the maximum absolute error defined by

$$E^\infty = \max_{1 \le k \le n} |1 - H(x_k)| \tag{32}$$

- E^1 is the mean absolute error defined by

$$E^1 = \frac{1}{n} \sum_{k=1}^{n} |1 - H(x_k)|. \tag{33}$$

TABLE 1:

ϵ	q	E^∞	E^1
1 E-4	12	0.671 E-05	0.202 E-05
	16	0.100 E-07	0.581 E-09
	20	0.100 E-07	0.542 E-09
1 E-6	18	0.264 E-06	0.311 E-07
	22	0.101 E-09	0.118 E-10
	26	0.100 E-11	0.367 E-13
1 E-8	24	0.332 E-08	0.478 E-09
	28	0.197 E-11	0.266 E-12
	32	0.434 E-14	0.815 E-15
1 E-10	30	0.142 E-09	0.924 E-11
	34	0.983 E-13	0.660 E-14
	38	0.140 E-13	0.152 E-14

We observe from the above results that in all our tests,

$$|e^{icx} - e^{bx^2} \cdot \sum_{k=-q/2}^{q/2} h_k e^{ikx}| < 2\varepsilon \tag{34}$$

and for larger q the truncation errors are several orders of magnitude less than ε. The error estimate $O(\kappa \cdot \varepsilon)$ in (28) can thus be replaced by $O(\varepsilon)$ for most practical purposes.

Finally, by means of a simple linear transformation, the formula (28) generalizes from $[-\pi, \pi]$ to any interval $[a-d, a+d]$ to give us the result upon which the algorithm is based.

Corollary 2.6 *Let c be any real frequency, a, d be real numbers with $d > 0$ and $\varepsilon > 0$ be a given tolerance. Then, in the notation of this section,*

$$e^{icx} = e^{b((x-a)\pi/d)^2} \cdot \sum_{k=k_c-q/2}^{k_c+q/2} h_k e^{ik(x-a)\pi/d} + O(\varepsilon) \tag{35}$$

for $x \in [a - d/2, a + d/2]$.

3 Exact Statement of the Problem

In the following sections, we will assume that:

1. $\{\omega_1, \ldots, \omega_N\}$ and $\{x_1, \ldots, x_m\}$ are finite sequences of real numbers
2. n is an integer such that $\omega_k \in [-\frac{n}{2}, \frac{n}{2}]$ for $k = 1, \ldots, N$
3. $-\pi \leq x_1 < \cdots < x_m \leq \pi$
4. $\{\alpha_1, \ldots, \alpha_N\}$ is a finite sequence of complex numbers
5. We wish to evaluate the sums

$$S_{\alpha,\omega}(x_j) = \sum_{k=1}^{N} \alpha_k \cdot e^{i\omega_k x_j} \tag{36}$$

for $j = 1, \ldots, m$ with a relative accuracy $\varepsilon > 0$. More precisely, we are looking for a set of numbers $\tilde{S}(x_j)$ such that

$$\frac{|\tilde{S}(x_j) - S_{\alpha,\omega}(x_j)|}{\sum_{k=1}^{N} |\alpha_k|} \leq \varepsilon \tag{37}$$

4 Informal Description of the Algorithm

The results of section 2 lead us to observe that the linear transformation described by equation (36) has an approximate sparse factorization to a relative precision ϵ.

The factors correspond to the following operations which are performed by the algorithm:

1. Obtain approximate Fourier coefficients for $e^{-bx^2} S_{\alpha,\omega}(x)$ on the interval $[-2\pi, 2\pi]$.
2. Evaluate this Fourier series at equally-spaced points in $[-2\pi, 2\pi]$ using FFT.
3. Retrieve the approximate values of $S_{\alpha,\omega}(x)$ at equally spaced points in $[-\pi, \pi]$ by multiplying by e^{bx^2}.
4. Split $[-\pi, \pi]$ into small equal subintervals. Extend each subinterval to twice its length and multiply the approximate values of $S_{\alpha,\omega}(x)$ on each one by a Gaussian bell.
5. Apply FFT to new set of values on each subinterval, thereby obtaining short interpolating trigonometric polynomials which closely approximate $e^{-b'(x-c_k)^2} S_{\alpha,\omega}(x)$ on each k-th subinterval whose center is c_k.
6. Evaluate each series directly at the relevant points.
7. Retrieve approximate values of $S_{\alpha,\omega}(x_j)$ on the appropriate k-th subinterval by multiplying by $e^{b'(x_j-c_k)^2}$.

Remarks.

- In Step 1 a matrix P is applied to the vector of coefficients α_k. Due to (27), this matrix is given analytically and is banded to a precision ϵ with bandwidth q.
- Steps 3-7 can be combined and represented by a block-diagonal matrix Q where the k-th block is the product of the linear transformations on the k-th subinterval.
- The algorithm is divided into two parts:
  - Initialization, which precomputes and stores the matrix operators P and Q in terms of $\{\omega_k\}$ and $\{x_j\}$.
  - Evaluation, which applies the sequence of linear transformations to the vector $\{\alpha_k\}$ to obtain approximate values of the series at the specified points.

  For many applications, the frequencies and points are fixed and we wish to evaluate $S_{\alpha,\omega}(x)$ repeatedly for different sets of $\{\alpha_k\}$. With the above formulation the initialization only needs to be performed once for any number of subsequent evaluations giving considerable time savings in such cases.

- It remains for us to determine choices for the number of equally spaced points needed in step 2 and for the subinterval size in step 4 such that the operation count is minimized under the constraint that the interpolation errors are within our desired tolerance.

  The FFT requires at least 2 points per wavelength to resolve a particular frequency. The highest frequency term in the original series has a maximum of n wavelengths on the interval $[-2\pi, 2\pi]$. If the subinterval size is such that the highest frequency term has W wavelengths, the Fourier series for $e^{-b'(x-c_k)^2} S_{\alpha,\omega}(x)$ on each subinterval will have $W + q/2$ as its highest frequency. We thus need $2W + q$ points on each subinterval, or at least $2 + q/W$ points per wavelength to resolve this highest frequency and achieve our desired accuracy. Our investigations showed that an optimal choice is a subinterval size such that $W = q/2$, thus making our requirements 4 points per wavelength, $4n$ points on $[-2\pi, 2\pi]$ and $2q$ points on each extended subinterval. As there are $2n$ points in $[-\pi, \pi]$ and q points in each unextended subinterval, the number of subintervals is given by $2n/q$.

5 Notation

In this section we introduce some notation to be used in the detailed description of the algorithm.

We assume that the problem to be solved is that as described in section 3 above. We assume too that $\varepsilon > 0$ is a given real error tolerance.

First we use the result 2.6 of section 2.2 with $a = 2\pi$ to obtain an approximation to the series on $[-\pi, \pi]$.

We define p_k to be the nearest integer to $2\omega_k$ for $k = 1, \ldots, N$.

The real number b is defined such that $e^{-b(2\pi)^2} = \varepsilon$, giving us

$$b = \frac{\log_e(1/\varepsilon)}{4\pi^2}. \tag{38}$$

From equation (27), each term $e^{-bx^2} \cdot e^{i\omega_k x}$ can be accurately represented by a short trigonometric polynomial whose dominant frequency is p_k. We define sets of numbers $\{P_{j1}\}, \ldots, \{P_{jN}\}$ to be the coefficients for each such polynomial according to the expression

$$P_{jk} = \frac{1}{4\sqrt{b\pi}} e^{-(\omega_k - (p_k+j)/2)^2/4b} \tag{39}$$

for $k = 1, \ldots, N$ and $j = -q/2, \ldots, q/2$. Here we define the bandwidth q to be the smallest power of 2 such that

$$q \geq 2 \cdot \left\lceil \frac{2}{\pi} \cdot \log_e\left(\frac{1}{\varepsilon}\right) \right\rceil \tag{40}$$

as in (26), so that due to corollary 2.6,

$$e^{i\omega_k x} = e^{bx^2} \cdot \sum_{j=-q/2}^{q/2} P_{jk} \cdot e^{i(p_k+j)x/2} + O(\varepsilon) \tag{41}$$

for $k = 1, \ldots, N$ and $x \in [-\pi, \pi]$.

We now define a function $T(x)$, polynomial coefficients $\{\beta_j\}$, and n to be a power of 2 such that

$$T(x) = \sum_{k=1}^{N} \alpha_k \sum_{j=-q/2}^{q/2} P_{jk} \cdot e^{i(p_k+j)x/2} \equiv \sum_{j=-n}^{n} \beta_j e^{ijx/2}. \tag{42}$$

We also define a function

$$U(x) = e^{bx^2} \cdot T(x) \tag{43}$$

Observation 5.1 *$U(x)$ can be viewed as an approximation to $S_{\alpha,\omega}(x)$ and furthermore, due to (41),(42),(43) and the triangle inequality,*

$$|S_{\alpha,\omega}(x) - U(x)| = O(\varepsilon) \cdot \sum_{j=1}^{N} |\alpha_j| \tag{44}$$

for all $x \in [-\pi, \pi]$.

Let $\{t_j\}$ be a set of $4n$ equally spaced points in $[-2\pi, 2\pi]$ defined by

$$t_j = -2\pi + (j-1)\pi/n. \tag{45}$$

for $j = 1, \ldots, 4n$.

We next consider the result 2.6 applied to each of a set of small subintervals of $[-\pi, \pi]$, and introduce a notation for such subintervals and their associated expansions.

Let $M = 2n/q$ denote the number of subintervals on $[-\pi, \pi]$.

We define the set of subintervals $\{A_k\}$ for $k = 1, \ldots, M$ to be an equal partitioning of $[-\pi, \pi]$ by the formula

$$A_k = [c_k - \frac{\sigma}{2}, c_k + \frac{\sigma}{2}] \tag{46}$$

where the length of each subinterval is $\sigma = 2\pi/M$, and the center of each is given by

$$c_k = -\pi + (k - \frac{1}{2})\sigma. \tag{47}$$

The extended subintervals $\{B_k\}$ are then defined for $k = 1, \ldots, M$ by

$$B_k = [c_k - \sigma, c_k + \sigma] \tag{48}$$

so that each A_k is of length σ and each B_k is of length 2σ, both centered at c_k.

The scaled Gaussian bell $e^{-b'(x-c_k)^2}$ on each B_k is characterized by a number b' such that $e^{-b'\sigma^2} = \varepsilon$, giving us

$$b' = \frac{\log_e(1/\varepsilon)}{\sigma^2} = \frac{4\pi^2}{\sigma^2} \cdot b. \tag{49}$$

The subintervals have been chosen such that on each B_k the Fourier series approximation to $e^{-b'(x-c_k)^2} \cdot S_{\alpha,\omega}(x)$ to a precision ε has $2q$ terms.

For $k = 1, \ldots, M$ let the set of points $\{t_j^k\}$ be the subset of $\{t_j\}$ which lie in B_k defined by

$$t_j^k = c_k + \frac{j\sigma}{q} \tag{50}$$

for $j = -q, \ldots, q-1$.

Let us now define the sets of numbers $\{V_j^1\}, \ldots, \{V_j^M\}$ by

$$V_j^k = e^{-b'(t_j^k - c_k)^2} \cdot U(t_j^k) \tag{51}$$

$$= e^{-b'(t_j^k - c_k)^2} \cdot e^{b(t_j^k)^2} \cdot T(t_j^k) \tag{52}$$

for $j = -q, \ldots, q-1$.

We denote by $\{\gamma_l^k\}$ the Discrete Fourier Transform of each $\{V_j^k\}$ viewed as a sequence in j for each fixed k, i.e.

$$\gamma_l^k = \frac{1}{2q} \sum_{j=-q}^{q-1} V_j^k \cdot e^{-2\pi i l j/2q} \tag{53}$$

for $j = -q, \ldots, q-1$ and $k = 1, \ldots, M$.

Observation 5.2 *For $k = 1, \ldots, M$, the set of coefficients $\{\gamma_j^k\}$ defines a q-th order interpolating trigonometric polynomial which can be viewed as an approximation to $e^{-b'(x-c_k)^2} S_{\alpha,\omega}(x)$ on the interval B_k. Furthermore, defining a function on $[-\pi, \pi]$ by*

$$\tilde{S}(x) = e^{b'(x-c_k)^2} \cdot \sum_{l=-q}^{q-1} \gamma_l^k \cdot e^{il(x-c_k)\pi/\sigma} \qquad \text{when } x \in A_k, \tag{54}$$

we get

$$\frac{|\tilde{S}(x) - S_{\alpha,\omega}(x)|}{\sum_{j=1}^{N} |\alpha_j|} = O(\epsilon) \tag{55}$$

for all $x \in [-\pi, \pi]$.

We partition the set of points $\{x_j\}$ according to subinterval, and for each $k = 1, \ldots, M$ we define $\{x_1^k, \ldots, x_{n_k}^k\}$ to be the subset consisting of all points which lie in A_k. The integer n_k denotes the number of points in the k-th subinterval.

We notice now that (52),(53) and (54) constitute a sequence of linear transformations which can be combined into a single matrix on each subinterval. To this end we define

$$Q_{jl}^k = e^{b'(x_j^k - c_k)^2} \cdot \sum_{p=-q}^{q-1} e^{ip(x_j^k - c_k)\pi/\sigma} \cdot \frac{1}{2q} \cdot e^{-2\pi i p l/2q} \cdot e^{-b'(t_l^k - c_k)^2} \cdot e^{b(t_l^k)^2} \tag{56}$$

for $k = 1, \ldots, M$, $j = 1, \ldots, n_k$ and $l = -q, \ldots, q-1$, so that,

Observation 5.3 *In the notation of this section,*

$$\tilde{S}(x_j^k) = \sum_{l=-q}^{q-1} Q_{jl}^k \cdot T(t_l^k) \tag{57}$$

for $k = 1, \ldots, M$ *and* $j = 1, \ldots, n_k$.

A closer inspection of (56) reveals that the sum over p is in fact a geometrical series and substituting for

$$\frac{l}{q} = \frac{(t_l^k - c_k)}{\sigma}$$

from (50), the sum can be written as

$$\sum_{p=-q}^{q-1} e^{ip\pi((x_j^k - c_k)/\sigma) - ip\pi l/q} = \sum_{p=-q}^{q-1} e^{ip(x_j^k - t_l^k)\pi/\sigma}. \tag{58}$$

Writing

$$y_{jl}^k = (x_j^k - t_l^k)\pi/\sigma, \tag{59}$$

the sum can be further simplified to

$$\sum_{p=-q}^{q-1} e^{ipy_{jl}^k} = \left(\frac{e^{-iqy_{jl}^k} - e^{iqy_{jl}^k}}{1 - e^{iy_{jl}^k}} \right) \tag{60}$$

$$= e^{-iy_{jl}^k/2} \cdot \frac{\sin(qy_{jl}^k)}{\sin(y_{jl}^k/2)} \quad \text{if } y_{jl}^k \neq 0 \tag{61}$$

and

$$\sum_{p=-q}^{q-1} e^{ipy_{jl}^k} = 2q \quad \text{if } y_{jl}^k = 0. \tag{62}$$

This latter equation corresponds to the case when $x_j^k = t_l^k$ *for some* j, k, l, i.e. when one of the points for evaluation coincides with one of the equally spaced points.

The definition (56) can thus be rewritten as

$$Q_{jl}^k = \begin{cases} e^{b'(x_j^k - c_k)^2} \cdot \frac{1}{2q} \cdot e^{-iy_{jl}^k/2} \cdot \frac{\sin(qy_{jl}^k)}{\sin(y_{jl}^k/2)} \cdot e^{-b'(t_l^k - c_k)^2} \cdot e^{b(t_l^k)^2} & \text{if } y_{jl}^k \neq 0 \\ e^{b(t_l^k)^2} & \text{if } y_{jl}^k = 0 \end{cases}. \tag{63}$$

6 Detailed Description of the Algorithm

6.1 Numerical Procedure

This subsection contains a step by step breakdown of the details of the algorithm.

Initialization Phase
Comment [Input to this phase are the vectors $\{\omega_1, \ldots, \omega_N\}$ and $\{x_1, \ldots, x_m\}$ and an error tolerance ϵ.]

Step 1.

Comment [Choose parameters. Compute nearest integer frequencies $\{p_k\}$. Evaluate Fourier coefficients $\{P_{jk}\}$ for each term $e^{-bx^2}e^{i\omega_k x}$]

Determine n
Compute b and q in terms of the tolerance ϵ
do $k = 1, N$
  $p_k = int(2\omega_k)$
  do $j = -q/2, q/2$
    Compute P_{jk} according to (39)
  end do
end do

Step 2.

Comment [Geometrical preprocessing.]

Set number of subintervals, $M = 2n/q$.
Compute subinterval length, $\sigma = 2\pi/M$, and $b' = b \cdot 4\pi^2/\sigma^2$.
do $k = 1, M$
  Construct subintervals A_k and B_k and their center c_k.
  Construct subset $\{x_1^k, \ldots, x_{n_k}^k\}$
end do

Comment [Compute elements Q_{jl}^k of the $n_k \times 2q$ matrix corresponding to each k-th subinterval.]

do $k = 1, M$
  do $j = 1, n_k$
    do $l = -q, q-1$
      Compute Q_{jl}^k according to (63)
    end do
  end do
end do

End of Initialization Phase

Evaluation Phase
Comment [Input to this phase is the vector $\{\alpha_1, \ldots, \alpha_N\}$.]

Step 1.

Comment [Compute Fourier coefficients β_j for $e^{-bx^2} S_{\alpha,\omega}(x)$.]

do $k = 1, N$
  do $j = -q/2, q/2$
    $\beta_{p_k+j} = \beta_{p_k+j} + P_{jk} \cdot \alpha_k$
  end do
end do

Step 2.

Comment [Evaluate this Fourier Series at equispaced points on $[-2\pi, 2\pi]$ using inverse FFT. Determine subsets of the resulting values according to subinterval.]

Compute $T(t_j) = \sum_{k=-n/2}^{n/2} \beta_k \cdot e^{ikt_j/2}$.
do $k = 1, M$
  Construct subset $\{T(t^k_{-q}), \ldots, T(t^k_{q-1})\}$
end do

Step 3.

Comment [On each subinterval, compute approximations to the series at points for evaluation in terms of the values at equally spaced points.]

do $k = 1, M$
  do $j = 1, n_k$
    $\tilde{S}(x^k_j) = \sum_{l=-q}^{q-1} Q^k_{jl} \cdot T(t^k_l)$
  end do
end do

End of Evaluation Phase

6.2 Complexity Analysis

In this subsection we present the operation counts for each stage of the algorithm.

Step Number	Operation Count	Explanation
INITIALIZATION		
1	$O(N \cdot q)$	Calculation of nearest integer to each of N frequencies. Fourier coefficients P_{jk} are computed. There are q of these for each of the N terms in the original series.
2	$O(m \cdot q)$	Each of m points x_j is assigned to a single subinterval A_k. Coefficients Q_{jl}^k are computed. There are $2q$ of these for each of the m points.
EVALUATION		
1	$O(N \cdot q)$	Computation of the Fourier coefficients β_j. Each of the N coefficients α_k contributes to q of these.
2	$O(4n \log 4n)$	Inverse FFT to evaluate the approximating Fourier series at $4n$ points.
3	$O(m \cdot q)$	Computation of the approximation $\tilde{S}(x_j)$ at each of the m points. Each of these is given by a linear combination of the values at $2q$ equally spaced points.

Remark. In the description of the algorithm, n was chosen to be a power of 2. This is not an essential requirement, but our reason for making such a choice is that step 2 of our scheme consists of an FFT of size $4n$, and the FFT algorithm is most efficient when n is a power of 2.

Adding the operation counts for each step of the algorithm, we obtain the estimates

$$A \cdot N \cdot q + B \cdot m \cdot q \tag{64}$$

for the initialization time, and

$$C \cdot n \cdot \log n + D \cdot N \cdot q + E \cdot m \cdot q \tag{65}$$

for the evaluation time where the coefficients A, B, C, D, E depend on the computer system, language, implementation, etc.

In many cases of practical interest $n \sim N$ and using the fact that $q \sim \log(\frac{1}{\varepsilon})$ the CPU time estimate for evaluation reduces to the form

$$C \cdot N \cdot \log N + (D \cdot N + E \cdot m) \cdot \log\left(\frac{1}{\varepsilon}\right). \tag{66}$$

The storage requirements of the algorithm are also an important characteristic. From the above explanations for the initialization steps the asymptotic storage requirements are of the form

$$\lambda \cdot N \cdot q + \mu \cdot m \cdot q \tag{67}$$

where once again the coefficients λ, μ are software- and hardware-dependent.

7 Modifications for Special Cases

In certain cases of interest, the nature of the frequency distribution can be exploited to reduce the operation count of the algorithm. Here we present informal outlines of such modifications to the scheme in two particular situations.

7.1 Evaluation of Fourier Series at Arbitrary Points

If the frequencies ω_k are all integers, the problem reduces to the evaluation of a Fourier series on $[-\pi, \pi]$. We can obtain the values at equally spaced points on this interval by just using an FFT with 4 points per wavelength and without having to first multiply the series by a Gaussian bell. This eliminates the first step and roughly halves the required time for the second step. Although the asymptotic time complexity is unchanged, such a modification yields an anticipated speedup of a factor of 2.

7.2 Non-Homogeneous Frequency Distributions

The complexity of the algorithm is given in terms not of the number of frequencies, but in terms of the size, n, of the smallest interval containing all the frequencies. Thus, for some fixed number N of frequencies, the operation count will be much lower if the frequencies are clustered together on an interval of size smaller than N than if they are widely spaced on an interval of size much larger than N.

In some cases the frequencies may be distributed in a highly non-uniform manner, and can be grouped into widely separated clusters on the real line. We may improve the algorithm's performance in such cases by treating the series as a sum of separate sub-series, applying the method to each and adding the results at the end. If a sub-series has $2q$ or fewer terms, then we evaluate it directly. We also use the following simple result:

Observation 7.1 *Let a, d be real numbers with $d > 0$ and suppose $\omega_k \in [a - d, a + d]$ for $k = 1, \ldots, N$. Then we can write*

$$\sum_{k=1}^{N} \alpha_k \cdot e^{i\omega_k x} = e^{iax} \sum_{k=1}^{N} \alpha_k \cdot e^{i(\omega_k - a)x}. \tag{68}$$

For the series on the left hand side, $n \sim |a| + d$ whereas for the series on the right hand side, all the frequencies are in $[-d, d]$, so $n \sim d$, giving us considerable time savings when $|a| \gg d$.

Remark.
This algorithm performs well when

- the frequencies within a cluster are close together
- there are very few clusters

and not so well if

- the frequencies are widely separated
- there are many clusters.

Most cases likely to be encountered in practice fall in the first category.

8 Numerical Results

A FORTRAN implementation of the algorithm of this paper has been written and consists of two main subroutines, the first implementing the initialization stage, and the second the evaluation stage.

Our numerical experiments were conducted on the Sun Sparcstation 1 and in this section we present the results from some of our investigations. The 3 examples below illustrate the performance of the algorithm when applied to a variety of input data for problems of different sizes and different error tolerances. All computations were performed in double precision arithmetic and in each case the series was evaluated by the direct method for error estimation and timing comparisons.

A description of the entries in the tables follows, where N is the number of terms in the series, m is the number of points for evaluation, $S_{\alpha,\omega}(x_k)$ denotes the sum (1) evaluated directly at the point x_k and $\tilde{S}(x_k)$ denotes our approximation to the sum (1) at the point x_k as evaluated using the algorithm of this paper.

- E^{∞} is the maximum relative error at any point of evaluation defined by:

$$E^{\infty} = \max_{1 \le k \le m} \frac{|\tilde{S}(x_k) - S_{\alpha,\omega}(x_k)|}{\sum_{j=1}^{N} |\alpha_j|} \tag{69}$$

- E^{1} is the mean relative error defined by:

$$E^{1} = \frac{1}{m} \sum_{k=1}^{m} \frac{|\tilde{S}(x_k) - S_{\alpha,\omega}(x_k)|}{\sum_{j=1}^{N} |\alpha_j|} \tag{70}$$

- t_{alg}^{init} is the initialization time for the algorithm
- t_{alg}^{eval} is the CPU time for the subsequent evaluation stage of the algorithm

- t_{dir} is the CPU time for the direct evaluation of the series
- t_{FFT} is the CPU time for a single FFT of size N.

Example 1.
Here we evaluate a Fourier series at non-equally spaced points by choosing frequencies $\{\omega_k\}$ and points $\{x_j\}$ as defined by the formulae:

$$\omega_k = k - 1 - N/2 \text{ for } k = 1, \ldots, N \tag{71}$$

$$x_j = -\pi \cos\left(\frac{j-1}{m-1}\pi\right) \text{ for } j = 1, \ldots, m \tag{72}$$

and the coefficients $\{\alpha_k\}$ generated randomly on the unit square in the complex plane with $\Re(\alpha_k) \in [0,1]$, $\Im(\alpha_k) \in [0,1]$. The method for such a problem as described in section 7 was employed in this case. Results of our experiments are presented in Tables 2-4.

TABLE 2: Numerical Results for Example 1 with $\epsilon = 10^{-2}$, $q = 8$

N	m	E^{∞}	E^{1}	t_{alg}^{init}	t_{alg}^{eval}	t_{dir}	t_{FFT}
64	64	0.127 E-03	0.359 E-04	0.02	0.01	0.04	0.002
128	128	0.131 E-03	0.247 E-04	0.06	0.02	0.17	0.004
256	256	0.102 E-03	0.176 E-04	0.10	0.05	0.65	0.009
512	512	0.808 E-04	0.130 E-04	0.22	0.10	2.51	0.021
1024	1024	0.705 E-04	0.950 E-05	0.46	0.20	10.01	0.047
2048	2048	0.556 E-04	0.659 E-05	0.92	0.43	40.03	0.102
4096	4096	0.624 E-04	0.469 E-05	1.81	0.97	160.48	0.219

TABLE 3: Numerical Results for Example 1 with $\epsilon = 10^{-5}$, $q = 16$

N	m	E^{∞}	E^{1}	t_{alg}^{init}	t_{alg}^{eval}	t_{dir}	t_{FFT}
64	64	0.294 E-06	0.523 E-07	0.04	0.02	0.04	0.002
128	128	0.375 E-06	0.421 E-07	0.10	0.04	0.17	0.004
256	256	0.258 E-06	0.198 E-07	0.18	0.06	0.65	0.009
512	512	0.210 E-06	0.192 E-07	0.35	0.14	2.51	0.021
1024	1024	0.144 E-06	0.127 E-07	0.74	0.29	10.01	0.047
2048	2048	0.112 E-06	0.935 E-08	1.47	0.64	40.03	0.102
4096	4096	0.972 E-07	0.629 E-08	2.95	1.41	160.48	0.219

TABLE 4: Numerical Results for Example 1 with $\epsilon = 10^{-10}$, $q = 32$

N	m	E^{∞}	E^{1}	t_{alg}^{init}	t_{alg}^{eval}	t_{dir}	t_{FFT}
64	64	0.207 E-10	0.190 E-11	0.08	0.03	0.04	0.002
128	128	0.318 E-10	0.298 E-11	0.15	0.05	0.17	0.004
256	256	0.161 E-10	0.112 E-11	0.30	0.11	0.65	0.009
512	512	0.162 E-10	0.808 E-12	0.60	0.22	2.51	0.021
1024	1024	0.920 E-11	0.526 E-12	1.25	0.46	10.01	0.047
2048	2048	0.124 E-10	0.379 E-12	2.63	0.93	40.03	0.102
4096	4096	0.819 E-11	0.292 E-12	5.36	2.13	160.48	0.219

Remark. In this example, direct evaluation of the trigonometric polynomial was performed via Horner's scheme which can be found in [3] for example.

Example 2.
In this example, the frequencies $\{\omega_k\}$ were randomly distributed on the interval $[-N/2, N/2]$, the points $\{x_j\}$ randomly distributed on $[-\pi, \pi]$ and the coefficients $\{\alpha_k\}$ generated randomly as before on the unit square in the complex plane with $\Re(\alpha_k) \in [0,1]$, $\Im(\alpha_k) \in [0,1]$. The results are presented in Tables 5-7.

TABLE 5: Numerical Results for Example 2 with $\epsilon = 10^{-2}$, $q = 8$

N	m	E^∞	E^1	Alg time		Dir time	
				init	eval	init	eval
64	64	0.280 E-03	0.837 E-04	0.06	0.02	0.15	0.02
128	128	0.295 E-03	0.499 E-04	0.11	0.05	0.51	0.08
256	256	0.247 E-03	0.371 E-04	0.20	0.09	2.33	0.33
512	512	0.127 E-03	0.258 E-04	0.46	0.19	9.29	1.41
1024	1024	0.141 E-03	0.181 E-04	0.83	0.37	44.38	
2048	2048	0.137 E-03	0.134 E-04	1.73	0.84	180.51	
4096	4096	0.734 E-04	0.916 E-05	3.47	1.71	787.71	

TABLE 6: Numerical Results for Example 2 with $\epsilon = 10^{-5}$, $q = 16$

N	m	E^∞	E^1	Alg time		Dir time	
				init	eval	init	eval
64	64	0.790 E-06	0.134 E-06	0.09	0.03	0.15	0.02
128	128	0.745 E-06	0.997 E-07	0.18	0.05	0.51	0.08
256	256	0.517 E-06	0.454 E-07	0.35	0.11	2.33	0.33
512	512	0.349 E-06	0.336 E-07	0.67	0.23	9.29	1.41
1024	1024	0.397 E-06	0.303 E-07	1.37	0.48	44.38	
2048	2048	0.250 E-06	0.180 E-07	2.72	1.10	180.51	
4096	4096	0.156 E-06	0.129 E-07	5.39	2.20	787.71	

TABLE 7: Numerical Results for Example 2 with $\epsilon = 10^{-10}$, $q = 32$

N	m	E^∞	E^1	Alg time		Dir time	
				init	eval	init	eval
64	64	0.436 E-10	0.329 E-11	0.15	0.05	0.15	0.02
128	128	0.397 E-10	0.295 E-11	0.31	0.09	0.51	0.08
256	256	0.388 E-10	0.127 E-11	0.61	0.18	2.33	0.33
512	512	0.266 E-10	0.184 E-11	1.13	0.38	9.29	1.41
1024	1024	0.264 E-10	0.112 E-11	2.32	0.78	44.38	
2048	2048	0.185 E-10	0.714 E-12	4.63	1.57	180.51	
4096	4096	0.153 E-10	0.488 E-12	9.21	3.03	787.71	

Remark. In order to make a fair comparison of our method with the direct one in this example, we also split the latter into an initialization stage, in which all the required complex exponentials are precomputed and stored, and an evaluation stage in which the precomputed matrix is applied to the set of coefficients $\{\alpha_k\}$. However, such a formulation requires $N \cdot m$ storage, and for $N, m \geq 1024$ the available memory on the machine is insufficient. For larger problems, the direct scheme has to compute all exponentials as needed and timings are presented for such an implementation in these cases.

Example 3.
In this example, the frequencies $\{\omega_k\}$ were randomly distributed on the interval $[-N, -N/2]$ for $k = 1, N/2$ and on $[N/2, N]$ for $k = N/2 + 1, N$. As in the previous example, the points $\{x_j\}$ were randomly distributed on $[-\pi, \pi]$ and the coefficients $\{\alpha_k\}$ generated randomly on the unit square $\Re(\alpha_k) \in [0, 1]$, $\Im(\alpha_k) \in [0, 1]$.

The sums were calculated in two ways: first using the algorithm directly, and then using the method of section 7, i.e. viewing $S_{\alpha,\omega}$ as a sum of two separate sub-series, applying the algorithm to each and finally adding the results. Results of these tests are presented in Table 8.

TABLE 8: Numerical Results for Example 3 with $\epsilon = 10^{-2}$, $q = 8$

N	m	1 cluster		2 clusters		t_{dir}
		t_{alg}^{init}	t_{alg}^{eval}	t_{alg}^{init}	t_{alg}^{eval}	
128	256	0.18	0.08	0.30	0.06	1.64
256	512	0.37	0.18	0.48	0.18	6.69
512	1024	0.76	0.35	1.02	0.36	26.29
1024	2048	1.51	0.80	2.16	0.78	101.08

Remark. The timings for the 1 cluster and 2 cluster case are similar in this particular example.

Example 4.
In this example, the frequencies $\{\omega_k\}$ were randomly distributed on the interval $[-3N/2, -N]$ for $k = 1, N/2$ and on $[N, 3N/2]$ for $k = N/2 + 1, N$. Once again the points $\{x_j\}$ were randomly distributed on $[-\pi, \pi]$ and the coefficients $\{\alpha_k\}$ generated randomly on the unit square $\Re(\alpha_k) \in [0, 1]$, $\Im(\alpha_k) \in [0, 1]$.

The sums were calculated in two ways as in the previous example. Results of these tests are presented in Table 9.

TABLE 9: Numerical Results for Example 4 with $\epsilon = 10^{-2}$, $q = 8$

N	m	1 cluster		2 clusters		t_{dir}
		t_{alg}^{init}	t_{alg}^{eval}	t_{alg}^{init}	t_{alg}^{eval}	
128	256	0.25	0.15	0.30	0.06	1.64
256	512	0.50	0.32	0.48	0.18	6.69
512	1024	1.00	0.71	1.02	0.36	26.29
1024	2048	2.00	1.41	2.16	0.78	101.08

Remark. In this example the clusters of frequencies are more widely separated than in the previous example, and it is advantageous to consider the two sub-series separately in this case.

The results as presented in the Tables 2-9 lead us to make the following observations:

1. The accuracy of the results is well within the specified tolerance level, ϵ. For larger N, m the method tends to be slightly more accurate.

2. The break even point of our algorithm with the direct calculation depends on the tolerance, but is at roughly $N, m = 64$ if the initialization time is ignored. If the initialization time is included, the extrapolated break even point is at $N, m \approx 32$. For $N, m = 4096$ our scheme is about 400 times faster than the direct method.

3. As expected, the computation time grows slightly more slowly than linearly with the numbers of terms and points.

4. Evaluation of a Fourier series at an arbitrary set of points is roughly 4 times as expensive as an FFT of the same size for 4-5 digits of accuracy.

5. In cases when the frequencies can be separated into widely separated clusters, it is often cheaper computationally to consider the sub-series corresponding to each cluster separately.

9 Generalizations and Conclusions

The results of this paper can be generalized in the following ways:

1. One of the more far-reaching extensions of the method is a version of the algorithm in higher dimensions. Investigations into this are currently in progress.

2. The Helmholtz equation in 2 dimensions is given by

$$\nabla^2\phi + \kappa^2\phi = 0$$

   and has particular solutions of the form

$$\phi(x, y) = e^{i\mu x} \cdot e^{i\nu y}$$

   where $\mu^2 + \nu^2 = \kappa^2$. Solutions of this equation consist of linear combinations of such functions subject to some boundary conditions, and the results of this paper admit a generalization which constitutes a fast Helmholtz solver.

3. Some series of special functions reduce to the form (1) and can thus be rapidly evaluated using the algorithm of this paper. In addition, a set of interpolation techniques similar to those used in our scheme may be applied to other orthonormal bases of functions in place of trigonometric polynomials to give fast algorithms for other integral transforms.

In conclusion, an algorithm has been presented for the rapid evaluation of series of the form (1). The scheme is based on piecewise trigonometric interpolation of such series, and subsequent direct evaluation of the resulting trigonometric polynomials. The problem can be viewed as a generalization of the Discrete Fourier Transform, and the algorithm, while making use of some simple results from analysis, is very versatile, and has wide-ranging potential applications in many branches of mathematics, science and engineering.

Acknowledgements. I would like to thank Professor Vladimir Rokhlin for suggesting the topic of this paper, and for his continuing support and guidance. My thanks also to Yu Chen and Gregory Matvienko for many interesting and useful discussions.

References

[1] I.S. Gradshteyn and I.M. Ryzhik, *Table of Integrals, Series and Products*, Academic Press Inc., 1980.

[2] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, 1980.

[3] G. Dahlquist and A. Björck, *Numerical Methods*, Prentice Hall Inc., Englewood Cliffs, N.J., 1974.

[4] H. Joseph Weaver, *Theory of Discrete and Continuous Fourier Analysis*, Wiley, 1989.