SIAM J. SCI. COMPUT.
Vol. 14, No. 6, pp. 1368–1393, November 1993

FAST FOURIER TRANSFORMS FOR NONEQUISPACED DATA*

A. DUTT† AND V. ROKHLIN†

Abstract. A group of algorithms is presented generalizing the fast Fourier transform to the case of noninteger frequencies and nonequispaced nodes on the interval $[-\pi, \pi]$. The schemes of this paper are based on a combination of certain analytical considerations with the classical fast Fourier transform and generalize both the forward and backward FFTs. Each of the algorithms requires $O(N \cdot \log N + N \cdot \log(1/\varepsilon))$ arithmetic operations, where ε is the precision of computations and N is the number of nodes. The efficiency of the approach is illustrated by several numerical examples.

Key words. fast Fourier transform, Fourier analysis, trigonometric series, interpolation, approximation theory

AMS subject classifications. 65R10, 65T10, 65T20, 65Y20

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 $e^{i\omega x}$ are eigenfunctions of the differentiation operator and can be effectively used to model solutions of differential equations which arise in the fields mentioned above.

More recently, the arrival of digital computers and the development of the fast Fourier transform (FFT) algorithm in the 1960s (see [6]) have established Fourier analysis as a powerful and practical numerical tool. The FFT, which computes discrete Fourier transforms (DFTs), is now central to many areas, most notably spectral analysis and signal processing. In some applications, however, the input data is not uniformly spaced, a condition that is required for the FFT. In this paper we present a set of algorithms for computing more efficiently some generalizations of the DFT, namely, the forward and inverse transformations described by the equations

$$f_j = \sum_{k=0}^{N} \alpha_k \cdot e^{i\omega_k x_j} \tag{1}$$

for $j = 0, \ldots, N$, where $f_j \in \mathbf{C}$, $\alpha_k \in \mathbf{C}$, $\omega_k \in [-N/2, N/2]$, and $x_j \in [-\pi, \pi]$. Each algorithm requires a number of arithmetic operations proportional to

$$N \cdot \log N + N \cdot \log\left(\frac{1}{\varepsilon}\right), \tag{2}$$

where ε is the desired accuracy, compared with $O(N^2)$ operations required for the direct evaluation of (1) and $O(N^3)$ for the direct inversion.

Remark 1.1. The DFT can be described by either of the two closely related formulae

$$f_j = \sum_{k=0}^{N-1} \alpha_k \cdot e^{2\pi i k j/N} \tag{3}$$

*Received by the editors June 17, 1992; accepted for publication (in revised form) November 19, 1992.

†Department of Computer Science, Yale University, P.O. Box 2158, Yale Station, New Haven, Connecticut 06520. The work of these authors was supported in part by Office of Naval Research grant N00014-89-J-1527 and in part by National Science Foundation grant DMS-9012751.

for $j = 0, \ldots, N-1$, and

$$f_j = \sum_{k=-N/2}^{N/2-1} \alpha_k \cdot e^{2\pi i k j/N} \tag{4}$$

for $j = -N/2, \ldots, N/2-1$. While the form (3) is normally used when the FFT is discussed, the form (4) is usually preferred in applications of the DFT to analysis (see, for example, [2], [9]).

Remark 1.2. The FFT algorithm reduces the number of operations for the DFT from $O(N^2)$ to $O(N \cdot \log N)$ by a sequence of algebraic manipulations (for more details on FFTs, see [4], [6], [7], [12]–[14]). In the more general case of (1), the structure of the linear transformation is also exploitable, and the algorithms of this paper combine certain analytical results with the existing FFT.

The plan of the paper is as follows. We start in §2 with some results from the analysis and approximation theory which are used in the design of the algorithms. An exact statement of the problem in §3 is then followed by informal descriptions of the algorithms in §4. In §5 we introduce some notation that is used in a set of more detailed algorithm descriptions in §6. Six numerical examples are presented in §7 to illustrate the performance of the schemes. Finally, §8 lists some generalizations and conclusions.

2. Mathematical and numerical preliminaries.

2.1. Elementary analytical tools. In this subsection we summarize some well-known results to be used in the remainder of the paper. Lemmas 2.1 and 2.2 are obvious, and Lemmas 2.3 and 2.4 can be found, for example, in [10].

LEMMA 2.1. *For any real c,*

$$\int_{-\pi}^{\pi} e^{icx} dx = \frac{2}{c} \sin(c\pi). \tag{5}$$

LEMMA 2.2. *For any integer k,*

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} e^{ikx} dx = \begin{cases} 1 & \text{if } k = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

LEMMA 2.3. *For any real $b > 0$ and complex z,*

$$\int_{-\infty}^{\infty} e^{-bx^2} \cdot e^{zx} dx = \sqrt{\frac{\pi}{b}} \cdot e^{z^2/4b}. \tag{7}$$

LEMMA 2.4. *For any real $b > 0$ and $a > 0$,*

$$\int_{a}^{\infty} e^{-bx^2} dx < \frac{e^{-ba^2}}{2ba}. \tag{8}$$

Proof.

$$\int_{a}^{\infty} e^{-bx^2} dx = \int_{0}^{\infty} e^{-b(x+a)^2} dx < e^{-ba^2} \int_{0}^{\infty} e^{-2bax} dx = \frac{e^{-ba^2}}{2ba}. \tag{9}$$

□

2.2. Relevant facts from approximation theory. The principal tool of this paper is a somewhat detailed analysis of Fourier series of functions $\phi : [-\pi, \pi] \rightarrow \mathbf{C}$ given by the formula

$$\phi(x) = e^{-bx^2} \cdot e^{icx}, \tag{10}$$

where $b > \frac{1}{2}$ and c are real numbers. We present this analysis in the lemmas and theorems of this subsection, numbered 2.5–2.10.

Lemmas 2.5 and 2.6 provide two inequalities that are used in Theorem 2.7, and Theorems 2.7–2.9 are intermediate results leading to Theorem 2.10. This final theorem explains how to approximate functions of the form e^{icx} using a small number of terms, and the algorithms of this paper are based upon this result. We derive error bounds for all approximations that allow us to perform numerical computations to any specified accuracy.

LEMMA 2.5. *For any real $b > \frac{1}{2}$, c and any integer k,*

$$\left| 2 \int_{\pi}^{\infty} e^{-bx^2} \cos((c-k)x) dx + e^{-b\pi^2} \cdot \int_{-\pi}^{\pi} e^{i(c-k)x} dx \right| < 2\pi e^{-b\pi^2} \cdot \left(1 + \frac{1}{\pi^2}\right). \tag{11}$$

Proof. Using the triangle inequality and Lemma 2.4, we have

$$\begin{aligned} &\left| 2 \int_{\pi}^{\infty} e^{-bx^2} \cdot \cos((c-k)x) dx + e^{-b\pi^2} \cdot \int_{-\pi}^{\pi} e^{i(c-k)x} dx \right| \\ &\leq 2 \int_{\pi}^{\infty} e^{-bx^2} dx + 2\pi e^{-b\pi^2} < 2\pi e^{-b\pi^2} \cdot \left(\frac{1}{2b\pi^2} + 1 \right) \\ &< 2\pi e^{-b\pi^2} \cdot \left(\frac{1}{\pi^2} + 1 \right). \qquad \square \end{aligned} \tag{12}$$

LEMMA 2.6. *For any real $b > \frac{1}{2}$, c and any integer k,*

$$\left| 2 \int_{\pi}^{\infty} e^{-bx^2} \cos((c-k)x) dx + e^{-b\pi^2} \cdot \int_{-\pi}^{\pi} e^{i(c-k)x} dx \right| < \frac{8b\pi e^{-b\pi^2}}{(c-k)^2} \cdot \left(1 + \frac{1}{\pi^2}\right). \tag{13}$$

Proof. Integrating by parts, we have

$$\begin{aligned} &2 \int_{\pi}^{\infty} e^{-bx^2} \cdot \cos((c-k)x) dx \\ &= \frac{2}{c-k} \left[e^{-bx^2} \sin((c-k)x) \right]_{\pi}^{\infty} + \frac{4b}{c-k} \int_{\pi}^{\infty} x e^{-bx^2} \sin((c-k)x) dx \\ &= -\frac{2}{c-k} e^{-b\pi^2} \sin((c-k)\pi) + \frac{4b}{c-k} \int_{\pi}^{\infty} x e^{-bx^2} \sin((c-k)x) dx. \end{aligned} \tag{14}$$

After rearranging the terms in (14) and integrating by parts again, we obtain

$$\left| 2 \int_{\pi}^{\infty} e^{-bx^2} \cos((c-k)x) dx + \frac{2e^{-b\pi^2}}{c-k} \sin((c-k)\pi) \right| \tag{15}$$

$$= \left| \frac{4b}{c-k} \int_{\pi}^{\infty} x e^{-bx^2} \sin((c-k)x) dx \right|$$

$$= \left| -\frac{4b}{(c-k)^2} \left(\left[x e^{-bx^2} \cos((c-k)x) \right]_{\pi}^{\infty} - \int_{\pi}^{\infty} (1-2bx^2) e^{-bx^2} \cos((c-k)x) dx \right) \right|$$

$$\leq \frac{4b}{(c-k)^2} \left(\pi e^{-b\pi^2} + \int_{\pi}^{\infty} e^{-bx^2} dx + \int_{\pi}^{\infty} x \cdot 2bx e^{-bx^2} dx \right)$$

$$< \frac{4b}{(c-k)^2} \left(\pi e^{-b\pi^2} + \int_{\pi}^{\infty} e^{-bx^2} dx + \left[-x e^{-bx^2} \right]_{\pi}^{\infty} + \int_{\pi}^{\infty} e^{-bx^2} dx \right).$$

Finally, due to (15) and Lemmas 2.1 and 2.4, we have

(16)

$$\left| 2 \int_{\pi}^{\infty} e^{-bx^2} \cos((c-k)x) dx + e^{-b\pi^2} \cdot \int_{-\pi}^{\pi} e^{i(c-k)x} dx \right| < \frac{4b e^{-b\pi^2}}{(c-k)^2} \cdot \left(2\pi + \frac{2}{2b\pi} \right)$$

$$< \frac{8b\pi e^{-b\pi^2}}{(c-k)^2} \cdot \left(1 + \frac{1}{\pi^2} \right). \qquad \square$$

The following theorem provides an explicit expression for the coefficients of a Fourier series that approximates functions of the form (10).

THEOREM 2.7. *Let $\phi(x) = e^{-bx^2} e^{icx}$ for any real $b > \frac{1}{2}$, c. Then, for any $x \in (-\pi, \pi)$,*

$$\left| \phi(x) - \sum_{k=-\infty}^{\infty} \rho_k e^{ikx} \right| < e^{-b\pi^2} \cdot \left(4b + \frac{70}{9} \right), \tag{17}$$

where

$$\rho_k = \frac{1}{2\sqrt{b\pi}} e^{-(c-k)^2/4b} \tag{18}$$

for $k = -\infty, \ldots, \infty$.

Proof. We denote by σ_k the kth Fourier coefficient for ϕ, so that for $x \in (-\pi, \pi)$,

$$\phi(x) = \sum_{k=-\infty}^{\infty} \sigma_k e^{ikx}, \tag{19}$$

and due to Lemma 2.3 and (18), we have

(20)

$$\sigma_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} \phi(x) e^{-ikx} dx$$

$$= \frac{1}{2\pi} \left(\int_{-\infty}^{\infty} e^{-bx^2} e^{icx} e^{-ikx} dx - \int_{-\infty}^{-\pi} e^{-bx^2} e^{icx} e^{-ikx} dx - \int_{\pi}^{\infty} e^{-bx^2} e^{icx} e^{-ikx} dx \right)$$

$$= \frac{1}{2\pi} \left(\sqrt{\frac{\pi}{b}} \cdot e^{-(c-k)^2/4b} + \int_{\infty}^{\pi} e^{-bx^2 - icx + ikx} dx - \int_{\pi}^{\infty} e^{-bx^2 + icx - ikx} dx \right)$$

$$= \rho_k - \frac{1}{\pi} \int_{\pi}^{\infty} e^{-bx^2} \cos((c-k)x) dx.$$

Rearranging (20) and applying Lemmas 2.5 and 2.6, we obtain the inequalities

$$\left|\sigma_k - \rho_k - \frac{e^{-b\pi^2}}{2\pi}\int_{-\pi}^{\pi} e^{icx}e^{-ikx}dx\right| < e^{-b\pi^2}\cdot\left(1+\frac{1}{\pi^2}\right), \tag{21}$$

$$\left|\sigma_k - \rho_k - \frac{e^{-b\pi^2}}{2\pi}\int_{-\pi}^{\pi} e^{icx}e^{-ikx}dx\right| < \frac{4be^{-b\pi^2}}{(c-k)^2}\cdot\left(1+\frac{1}{\pi^2}\right), \tag{22}$$

and it now follows from the combination of (19), (21), and (22) that, for any $x \in (-\pi,\pi)$,

$$\begin{aligned}
&\left|\phi(x) - \sum_{k=-\infty}^{\infty}\rho_k e^{ikx} - e^{-b\pi^2}\cdot e^{icx}\right| \\
&= \left|\sum_{k=-\infty}^{\infty} e^{ikx}\cdot\left(\sigma_k - \rho_k - \frac{e^{-b\pi^2}}{2\pi}\int_{-\pi}^{\pi} e^{icx}e^{-ikx}dx\right)\right| \\
&< \sum_{k,|c-k|\geq 3}\frac{4be^{-b\pi^2}}{(c-k)^2}\cdot\left(1+\frac{1}{\pi^2}\right) + \sum_{k,|c-k|<3} e^{-b\pi^2}\cdot\left(1+\frac{1}{\pi^2}\right) \\
&< 4be^{-b\pi^2}\cdot\frac{9}{8}\cdot 2\cdot\sum_{k=3}^{\infty}\frac{1}{k^2} + 6e^{-b\pi^2}\cdot\frac{10}{9}.
\end{aligned} \tag{23}$$

Some elementary analysis yields

$$\sum_{k=3}^{\infty}\frac{1}{k^2} < \frac{1}{9} + \int_3^{\infty}\frac{dx}{x^2} = \frac{1}{9}+\frac{1}{3} = \frac{4}{9}, \tag{24}$$

and substituting (24) into (23), we have

$$\left|\phi(x) - \sum_{k=-\infty}^{\infty}\rho_k e^{ikx} - e^{-b\pi^2}\cdot e^{icx}\right| < e^{-b\pi^2}\cdot\left(4b+\frac{60}{9}\right). \tag{25}$$

To complete the proof we make use of the triangle inequality and (25) to obtain

$$\begin{aligned}
\left|\phi(x) - \sum_{k=-\infty}^{\infty}\rho_k e^{ikx}\right| &\leq \left|\phi(x) - \sum_{k=-\infty}^{\infty}\rho_k e^{ikx} - e^{-b\pi^2}\cdot e^{icx}\right| + \left|e^{-b\pi^2}\cdot e^{icx}\right| \\
&< e^{-b\pi^2}\cdot\left(4b+\frac{70}{9}\right). \qquad \square
\end{aligned} \tag{26}$$

Remark 2.1. According to Theorem 2.7, functions of the form $e^{-bx^2}e^{icx}$ can be approximated by a Fourier series whose coefficients are given analytically, and the error of the approximation decreases exponentially as b increases.

Remark 2.2. The coefficients ρ_k as defined by (18) have a peak at $k = [c]$, the nearest integer to c, and decay exponentially as $k \to \pm\infty$. We keep only the $q+1$ largest coefficients, where the integer q is chosen such that

$$q \geq 4b\pi, \tag{27}$$

so as to satisfy the inequality

$$e^{-(q/2)^2/4b} \leq e^{-b\pi^2}. \tag{28}$$

The following theorem estimates the truncation error under the conditions of Remark 2.2 and thus provides a way of approximating functions of the form (10) by a $(q+1)$-term series.

THEOREM 2.8. *Let $\phi(x) = e^{-bx^2}e^{icx}$ for any real $b > \frac{1}{2}$, c, and let q be an even integer such that $q \geq 4b\pi$. Then, for any $x \in (-\pi, \pi)$,*

$$\left|\phi(x) - \sum_{k=[c]-q/2}^{[c]+q/2} \rho_k e^{ikx}\right| < e^{-b\pi^2} \cdot (4b+9), \tag{29}$$

where $\{\rho_k\}$ are defined by (18).

Proof. For any $x \in (-\pi, \pi)$,

$$\begin{aligned}&\left|\phi(x) - \sum_{k=[c]-q/2}^{[c]+q/2} \rho_k e^{ikx}\right| \\ &\leq \left|\phi(x) - \sum_{k=-\infty}^{\infty} \rho_k e^{ikx}\right| + \left|\sum_{k>[c]+q/2} \rho_k e^{ikx}\right| + \left|\sum_{k<[c]-q/2} \rho_k e^{ikx}\right|.\end{aligned} \tag{30}$$

Due to (18) and the triangle inequality, we have the inequalities

$$\left|\sum_{k>[c]+q/2} \rho_k e^{ikx}\right| \leq \sum_{k=[c]+q/2+1}^{\infty} \frac{e^{-(c-k)^2/4b}}{2\sqrt{b\pi}} < \sum_{k=q/2}^{\infty} \frac{e^{-k^2/4b}}{\sqrt{2\pi}}, \tag{31}$$

$$\left|\sum_{k<[c]-q/2} \rho_k e^{ikx}\right| \leq \sum_{k=-\infty}^{[c]-q/2-1} \frac{e^{-(c-k)^2/4b}}{2\sqrt{b\pi}} < \sum_{k=q/2}^{\infty} \frac{e^{-k^2/4b}}{\sqrt{2\pi}}. \tag{32}$$

Some elementary analysis and an application of Lemma 2.4 yields

$$\sum_{k=q/2}^{\infty} e^{-k^2/4b} < e^{-(q/2)^2/4b} + \int_{q/2}^{\infty} e^{-x^2/4b}dx < e^{-(q/2)^2/4b} \cdot \left(1 + \frac{4b}{2q/2}\right), \tag{33}$$

and it follows from the combination of (27), (28), and (33) that

$$\sum_{k=q/2}^{\infty} e^{-k^2/4b} < e^{-b\pi^2} \cdot \left(1 + \frac{1}{\pi}\right). \tag{34}$$

Substituting (34) into (31) and (32), we have

$$\left|\sum_{k>[c]+q/2} \rho_k e^{ikx}\right| + \left|\sum_{k<[c]-q/2} \rho_k e^{ikx}\right| < \frac{2e^{-b\pi^2}}{\sqrt{2\pi}} \cdot \left(1 + \frac{1}{\pi}\right) < e^{-b\pi^2} \cdot \frac{10}{9}, \tag{35}$$

and finally, substituting (26) and (35) into (30), we obtain

$$\left|\phi(x) - \sum_{k=[c]-q/2}^{[c]+q/2} \rho_k e^{ikx}\right| < e^{-b\pi^2} \cdot \left(4b + \frac{70}{9} + \frac{10}{9}\right) < e^{-b\pi^2} \cdot (4b+9). \tag{36}$$

□

The following corollary describes a formula for approximating e^{icx} using a series of $q+1$ terms.

COROLLARY 2.9. *Suppose that $m \geq 2$ is an integer and that the conditions of Theorem 2.8 are satisfied. Then, multiplying both sides of* (29) *by e^{bx^2}, we obtain*

$$\left| e^{icx} - e^{bx^2} \cdot \sum_{k=[c]-q/2}^{[c]+q/2} \rho_k e^{ikx} \right| < e^{bx^2} \cdot e^{-b\pi^2} \cdot (4b+9) < e^{b\pi^2/m^2} \cdot e^{-b\pi^2} \cdot (4b+9) \tag{37}$$

for any $x \in [-\frac{\pi}{m}, \frac{\pi}{m}]$.

Finally, Theorem 2.10 makes use of a simple linear scaling to generalize the inequality (37) from $[-\frac{\pi}{m}, \frac{\pi}{m}]$ to any interval $[-d, d]$. This is the principal result of the section.

THEOREM 2.10. *Let $b > \frac{1}{2}, c, d > 0$ be real numbers, and let $m \geq 2, q \geq 4b\pi$ be integers. Then, for any $x \in [-d, d]$,*

$$\left| e^{icx} - e^{b(x\pi/md)^2} \cdot \sum_{k=[cmd/\pi]-q/2}^{[cmd/\pi]+q/2} \rho_k e^{ikx\pi/md} \right| < e^{-b\pi^2(1-1/m^2)} \cdot (4b+9) \tag{38}$$

where $\{\rho_k\}$ are defined by (18).

Remark 2.3. The error bounds obtained in the above theorems are rather pessimistic. Numerical estimates for the actual errors can be found in the Appendix to this paper.

3. Exact statement of the problem. In the remainder of this paper we will operate under the following assumptions:

1. $\omega = \{\omega_0, \ldots, \omega_N\}$ and $x = \{x_0, \ldots, x_N\}$ are finite sequences of real numbers.
2. $\omega_k \in [-N/2, N/2]$ for $k = 0, \ldots, N$.
3. $x_j \in [-\pi, \pi]$ for $j = 0, \ldots, N$.
4. $\alpha = \{\alpha_0, \ldots, \alpha_N\}$, $f = \{f_{-N/2}, \ldots, f_{N/2}\}$, $\beta = \{\beta_{-N/2}, \ldots, \beta_{N/2}\}$, $g = \{g_0, \ldots, g_N\}$, $\gamma = \{\gamma_0, \ldots, \gamma_N\}$ and $h = \{h_0, \ldots, h_N\}$ are finite sequences of complex numbers.

We will consider the problems of applying and inverting the Fourier matrix and its transpose, i.e., we are interested in the transformations $F, G : \mathbf{C}^{N+1} \to \mathbf{C}^{N+1}$ and their inverses defined by the formulae

$$f_j = F(\alpha)_j = \sum_{k=0}^{N} \alpha_k \cdot e^{i\omega_k \cdot 2\pi j/N} \tag{39}$$

for $j = -N/2, \ldots, N/2$, and

$$g_j = G(\beta)_j = \sum_{k=-N/2}^{N/2} \beta_k \cdot e^{ikx_j} \tag{40}$$

for $j = 0, \ldots, N$.

Remark 3.1. If $x_k = -\omega_k \cdot 2\pi/N$ for $k = 0, \ldots, N$, then (39) can be rewritten as

$$f_j = \sum_{k=0}^{N} \alpha_k \cdot e^{-ijx_k}, \tag{41}$$

or alternatively as $F = G^*$.

We will also consider the more general transformation $H : \mathbf{C}^{N+1} \to \mathbf{C}^{N+1}$ defined by the formula

$$h_j = H(\gamma)_j = \sum_{k=0}^{N} \gamma_k \cdot e^{i\omega_k x_j}. \tag{42}$$

More formally, we consider the following problems:

- Problem 1. Given α, find $f = F(\alpha)$.
- Problem 2. Given β, find $g = G(\beta)$.
- Problem 3. Given γ, find $h = H(\gamma)$.
- Problem 4. Given f, find $\alpha = F^{-1}(f)$.
- Problem 5. Given g, find $\beta = G^{-1}(g)$.

Remark 3.2. We wish to perform all calculations with a fixed relative accuracy $\varepsilon > 0$. In the case of Problem 1, for instance, we are looking for a vector $\tilde{f} = \{\tilde{f}_{-N/2}, \ldots, \tilde{f}_{N/2}\}$ such that

$$\frac{||\tilde{f} - f||}{||f||} \leq \varepsilon. \tag{43}$$

In this sense, all algorithms described in this paper are approximate ones.

4. Informal descriptions of the algorithms. In this section we give informal outlines of algorithms for Problems 1–5 of §3. More formal descriptions of these algorithms are presented in §6.

4.1. Algorithms 1, 2, and 3 for Problems 1, 2, and 3. The algorithms for these problems are based on the following principal observation.

Observation 4.1. According to Theorem 2.10, any function of the form e^{icx} can be accurately represented on any finite interval on the real line using a small number of terms of the form $e^{bx^2} \cdot e^{ikx}$, and this number of terms, q, is independent of the value of c.

The FFT algorithm applies the Fourier matrix to arbitrary complex vectors in $O(N \log N)$ operations when $\{\omega_k\}$ are integers and $\{x_j\}$ are equally spaced in $[-\pi, \pi]$. For the efficient application of the transformations described by (39), (40), and (42), we relate these more general cases to the equispaced case of the FFT. Observation 4.1 is used in two ways to achieve this:

- to approximate each $e^{i\omega_k x}$ in terms of a q-term Fourier series;
- to approximate the value of a Fourier series at each x_j in terms of values at the nearest q equispaced nodes.

This interpolation between equispaced and nonequispaced sets of points can thus be performed in $O(Nq)$ operations.

Observation 4.2. The overall complexity of each such algorithm that couples the FFT with the interpolation scheme will be $O(N \log N + Nq)$ operations.

4.2. Algorithms 4 and 5 for Problems 4 and 5. Here we are interested in applying the complex matrices A^{-1} and $(A^*)^{-1}$ to arbitrary complex vectors where the elements of A are defined by

$$A_{jk} = e^{ikx_j} \tag{44}$$

for $j = 0, \ldots, N$ and $k = -N/2, \ldots, N/2$.

We make use of the following two simple observations.

Observation 4.3. The matrix AA^* is Toeplitz, and furthermore, its $2N+1$ distinct elements can be computed in $O(N \log N + Nq)$ operations due to Observation 4.2.

Proof. It is obvious from (44) that

$$(AA^*)_{jl} = \sum_{k=0}^{N} e^{ijx_k} \cdot e^{-ilx_k} = \sum_{k=0}^{N} e^{i(j-l)x_k}, \tag{45}$$

which is a function only of $(j-l)$, and is of the same form as (41), the description for Problem 1. □

Observation 4.4. From elementary matrix identities we see that

$$A^{-1} \equiv A^*(AA^*)^{-1}, \tag{46}$$

$$(A^*)^{-1} \equiv (AA^*)^{-1}A. \tag{47}$$

The Toeplitz matrix AA^* can be applied to arbitrary vectors in $O(N \log N)$ operations using an FFT-based discrete convolution. $(AA^*)^{-1}$ can therefore be applied to a vector in $O(\kappa(A) \cdot N \log N)$ operations using the conjugate gradient method where $\kappa(A)$ is the condition number of A.

Observation 4.5. A^{-1} and $(A^*)^{-1}$ can be applied to arbitrary vectors using $O(\kappa(A) \cdot N \log N + Nq)$ operations due to Observations 4.2 and 4.4.

Remark 4.6. It is well known that the condition number of A is 1 if the points $\{x_j\}$ are equally spaced. While the condition number deteriorates as the distribution of points becomes more nonuniform, in many cases of practical interest the points will be fairly uniformly spaced, so the condition number will not be very large.

4.3. Algorithm 6 for a variant of Problem 5. The following lemma describes a way of computing the coefficients of an $(N/2+1)$-term Fourier series that is tabulated at $N+1$ points.

LEMMA 4.1. *Suppose that the* $N+1$ *function values* $g_0, \ldots, g_N$ *are given by the formula*

$$g_j = \sum_{k=-N/4}^{N/4} \beta_k \cdot e^{ikx_j}, \tag{48}$$

and the vector $\xi = \{\xi_0, \ldots, \xi_N\}$ *is the unique solution of the linear system described by the equations*

$$\sum_{j=0}^{N} \xi_j \cdot e^{ikx_j} = \begin{cases} 1 & \text{if } k = 0, \\ 0 & \text{otherwise} \end{cases} \tag{49}$$

for $k = -N/2, \ldots, N/2$. *Then, for* $k = -N/4, \ldots, N/4$,

$$\beta_k = \sum_{j=0}^{N} \xi_j \cdot g_j \cdot e^{-ikx_j}. \tag{50}$$

Proof. Substituting for g_j from (48), we have for $k = -N/4, \ldots, N/4$

$$\sum_{j=0}^{N} \xi_j \cdot g_j \cdot e^{-ikx_j} = \sum_{j=0}^{N} \xi_j \cdot e^{-ikx_j} \cdot \sum_{l=-N/4}^{N/4} \beta_l \cdot e^{ilx_j} \tag{51}$$

$$= \sum_{l=-N/4}^{N/4} \beta_l \cdot \sum_{j=0}^{N} \xi_j \cdot e^{i(l-k)x_j} \tag{52}$$

$$= \beta_k. \quad \square \tag{53}$$

Remark 4.7. According to (49) and Lemma 2.2,

$$\sum_{j=0}^{N} \xi_j \cdot e^{ikx_j} = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{ikx} dx \tag{54}$$

for $k = -N/2, \ldots, N/2$. Thus, the set of numbers $\{\xi_j\}$ can be considered as quadrature weights that integrate exactly all Nth order trigonometric polynomials at the nodes $\{x_j\}$.

Observation 4.8. Rewriting (49) in matrix notation, we see that

$$A^T \xi = (0, \ldots, 0, 1, 0, \ldots, 0)^T = A^* \xi, \tag{55}$$

where $A_{jk} = e^{ikx_j}$, so the vector ξ is real and can be computed using $O(\kappa(A) \cdot N \log N + Nq)$ operations due to Observation 4.5.

Observation 4.9. Equation (50) is of the same form as (41). Thus, provided the vector ξ is known, the vector β can be computed in $O(N \log N + Nq)$ operations due to Observation 4.2.

Remark 4.10. According to Observation 4.9, if a function described by an $(N/2+1)$-term Fourier series is tabulated at $N+1$ arbitrary nodes, the $(N/2+1)$ coefficients can be obtained in $O(N \log N + Nq)$ operations. Also, due to Observation 4.8, the precomputation of the numbers $\{\xi_j\}$ needed for this algorithm requires $O(\kappa(A) \cdot N \log N + Nq)$ operations.

5. Notation. In this section we introduce the notation to be used in the next section for the detailed algorithm descriptions.

For an integer $m \geq 2$ and a real number $b > 0$, we will define a real number $\varepsilon > 0$ by

$$\varepsilon = e^{-b\pi^2(1-1/m^2)} \cdot (4b+9), \tag{56}$$

and we will denote by q the smallest even natural number such that

$$q \geq 4b\pi. \tag{57}$$

For an integer m and a set of real numbers $\{\omega_k\}$, we will denote by μ_k the nearest integer to $m\omega_k$ for $k = 0, \ldots, N$, and by $\{P_{jk}\}$ a set of real numbers defined by the formula

$$P_{jk} = \frac{1}{2\sqrt{b\pi}} \cdot e^{-(m\omega_k - (\mu_k + j))^2/4b} \tag{58}$$

for $k = 0, \ldots, N$ and $j = -q/2, \ldots, q/2$.

Observation 5.1. Setting $d = \pi$ in Theorem 2.10, we see that

$$\left| e^{i\omega_k x} - e^{b(x/m)^2} \cdot \sum_{j=-q/2}^{q/2} P_{jk} \cdot e^{i(\mu_k + j)x/m} \right| < \varepsilon \tag{59}$$

for any $k = 0, \ldots, N$ and any $x \in [-\pi, \pi]$, where ε is defined by (56).

For a given set of complex numbers $\{\alpha_k\}$, we will denote by $\{\tau_j\}$ the unique set of complex coefficients such that

$$\sum_{k=1}^{N} \alpha_k \cdot \sum_{j=-q/2}^{q/2} P_{jk} \cdot e^{i(\mu_k + j)x/m} = \sum_{j=-mN/2}^{mN/2-1} \tau_j e^{ijx/m}, \tag{60}$$

so that

$$\tau_l = \sum_{j,k,\mu_k+j=l} \alpha_k \cdot P_{jk}. \tag{61}$$

We will denote by $\{T_j\}$ a set of complex numbers defined by the formula

$$T_j = \sum_{k=-mN/2}^{mN/2-1} \tau_k \cdot e^{2\pi ikj/mN} \tag{62}$$

for $j = -mN/2, \ldots, mN/2 - 1$.

Furthermore, we will denote by $\{\tilde{f}_j\}$ another set of complex numbers defined by the formula

$$\tilde{f}_j = e^{b(2\pi j/mN)^2} \cdot T_j \tag{63}$$

for $j = -N/2, \ldots, N/2$.

Observation 5.2. Combining (59)–(63) with the triangle inequality, we see that

$$|f_j - \tilde{f}_j| < \varepsilon \cdot \sum_{k=0}^{N} |\alpha_k| \tag{64}$$

for $j = -N/2, \ldots, N/2$, where $\{f_j = F(\alpha)_j\}$ are defined by (39).

For an integer m and a set of real numbers $\{x_j\}$, we will denote by ν_j the nearest integer to $x_j mN/2\pi$ for $j = 0, \ldots, N$, and by $\{Q_{jk}\}$ a set of real numbers defined by the formula

$$Q_{jk} = \frac{1}{2\sqrt{b\pi}} \cdot e^{-(x_j mN/2\pi - (\nu_j + k))^2/4b} \tag{65}$$

for $j = 0, \ldots, N$ and $k = -q/2, \ldots, q/2$.

Observation 5.3. Setting $d = N/2$ in Theorem 2.10, we see that

$$\left| e^{ikx_j} - e^{b(2\pi k/mN)^2} \cdot \sum_{l=-q/2}^{q/2} Q_{jk} \cdot e^{i(\nu_j + l)2\pi k/mN} \right| < \varepsilon \tag{66}$$

for any $j = 0, \ldots, N$ and any $k \in [-N/2, N/2]$, where ε is defined by (56).

For a given set of complex numbers $\{\beta_k\}$, we will denote by $\{u_k\}$ a set of complex numbers defined by the formula

$$u_k = \beta_k \cdot e^{b(2\pi k/mN)^2} \tag{67}$$

for $k = -N/2, \ldots, N/2$, and by $\{U_l\}$ a set of complex numbers defined by the formula

$$U_l = \sum_{k=-N/2}^{N/2} u_k \cdot e^{2\pi i k l/mN} \tag{68}$$

for $l = -mN/2, \ldots, mN/2 - 1$.

Furthermore, we will denote by $\{\tilde{g}_j\}$ another set of complex numbers defined by the formula

$$\tilde{g}_j = \sum_{l=-q/2}^{q/2} Q_{jl} \cdot U_{\nu_j + l} \tag{69}$$

for $j = 0, \ldots, N$.

Observation 5.4. Combining (66)–(69) with the triangle inequality, we see that

$$|g_j - \tilde{g}_j| < \varepsilon \cdot \sum_{k=0}^{N} |\beta_k| \tag{70}$$

for $j = 0, \ldots, N$, where $\{g_j = G(\beta)_j\}$ are defined by (40).

For a set of real numbers $\{x_j\}$, we will denote by η_j the nearest integer to $x_j N/2\pi$ for $j = 0, \ldots, N$, and by $\{R_{jk}\}$ a set of real numbers defined by the formula

$$R_{jk} = \frac{1}{2\sqrt{b\pi}} \cdot e^{-(x_j N/2\pi - (\eta_j + k))^2/4b} \tag{71}$$

for $j = 0, \ldots, N$ and $k = -q/2, \ldots, q/2$.

Observation 5.5. Setting $d = N/2$ in Theorem 2.10, we see that

$$\left| e^{ikx_j/m} - e^{b(2\pi k/mN)^2} \cdot \sum_{l=-q/2}^{q/2} R_{jk} \cdot e^{i(\eta_j + l)2\pi k/mN} \right| < \varepsilon \tag{72}$$

for any $j = 0, \ldots, N$ and any $k \in [-N/2, N/2]$, where ε is defined by (56).

For a given set of complex numbers $\{\gamma_k\}$, we will denote by $\{v_j\}$ the unique set of complex coefficients such that

$$\sum_{k=0}^{N} \gamma_k \cdot \sum_{j=-q/2}^{q/2} P_{jk} \cdot e^{i(\mu_k + j)x/m} = \sum_{j=-mN/2}^{mN/2} v_j \cdot e^{ijx/m}, \tag{73}$$

so that

$$v_l = \sum_{j,k,\eta_k + j = l} \gamma_k \cdot P_{jk}. \tag{74}$$

We denote by $\{V_l\}$ a set of complex numbers defined by the formula

$$V_l = \sum_{k=-mN/2}^{mN/2} v_k \cdot e^{b(2\pi k/m^2 N)^2} \cdot e^{2\pi i k l/m^2 N} \tag{75}$$

for $l = -m^2 N/2, \ldots, m^2 N/2 - 1$.

Furthermore, we will denote by $\{\tilde{h}_j\}$ another set of complex numbers defined by the formula

$$\tilde{h}_j = e^{b(x_j/m)^2} \cdot \sum_{l=-q/2}^{q/2} R_{jl} \cdot V_{\eta_j+l} \tag{76}$$

for $j = 0, \ldots, N$.

Observation 5.6. Combining (59) and (72)–(76) with the triangle inequality, we see that

$$|h_j - \tilde{h}_j| < \delta \cdot \sum_{k=0}^{N} |\gamma_k| \tag{77}$$

for $j = 0, \ldots, N$, where $\{h_j = H(\gamma)_j\}$ are defined by (42), and

$$\delta = 2e^{-b\pi^2(1-2/m^2)} \cdot (4b + 9). \tag{78}$$

For a set of real numbers $\{x_j\}$, A will denote a complex matrix whose elements are given by

$$A_{jk} = e^{ikx_j} \tag{79}$$

for $k = -N/2, \ldots, N/2$ and $j = 0, \ldots, N$, and $a = \{a_{-N}, \ldots, a_N\}$ will denote a set of complex numbers defined by the formula

$$a_k = \sum_{j=0}^{N} e^{ikx_j}. \tag{80}$$

Finally, $\xi = \{\xi_0, \ldots, \xi_N\}$ will denote a real vector defined by

$$\xi = (A^*)^{-1}(0, \ldots, 0, 1, 0, \ldots, 0)^T. \tag{81}$$

Remark 5.7. It is clear from Observation 4.3 that

$$(AA^*)_{jk} = a_{j-k}. \tag{82}$$

6. Detailed descriptions of the algorithms. This section contains step-by-step descriptions and operation counts for the six algorithms of this paper. In the tables below we will make use of the facts that $q \sim \log(\frac{1}{\varepsilon})$ and $m^2 \ll N$.

ALGORITHM 1.

Step	Complexity	Description
Init	$O(Nq)$	**Comment** [Input parameter is the vector $\{\omega_0, \ldots, \omega_N\}$.]
		Choose precision ε to be achieved.
		Set $b \approx \log(1/\varepsilon)$ and $q = \lceil 4b\pi \rceil$.
		do $k = 0, N$
		Determine μ_k, the nearest integer to $m\omega_k$
		do $j = -q/2, q/2$
		Compute P_{jk} according to (58)
		end do
		end do
		do $j = -N/2, N/2$
		Compute $e^{b(2\pi j/mN)^2}$
		end do

1	$O(Nq)$	**Comment** [Input parameter is the vector $\{\alpha_0, \ldots, \alpha_N\}$.]
		Comment [Compute Fourier coefficients τ_j.]
		do $k = 0, N$
		  do $j = -q/2, q/2$
		  $\tau_{\mu_k+j} \leftarrow \tau_{\mu_k+j} + P_{jk} \cdot \alpha_k$
		  end do
		end do

2	$O(mN \log N)$	**Comment** [Evaluate this Fourier Series at equispaced points in $[-m\pi, m\pi]$ using inverse FFT of size mN.] Compute $T_j = \sum_{k=-mN/2}^{mN/2-1} \tau_k \cdot e^{2\pi ikj/mN}$ for $j = -mN/2, \ldots, mN/2 - 1$.
3	$O(N)$	**Comment** [Scale the values at those points which lie in $[-\pi, \pi]$.] **do** $j = -N/2, N/2$ $\tilde{f}_j = T_j \cdot e^{b(2\pi j/mN)^2}$ **end do**
Total	$O(N \cdot \log(\frac{1}{\varepsilon}) + mN \cdot \log N)$	

ALGORITHM 2.

Step	Complexity	Description

Init	$O(Nq)$	**Comment** [Input parameter is the vector $\{x_0, \ldots, x_N\}$.]
		Choose precision ε to be achieved.
		Set $b \approx \log(1/\varepsilon)$ and $q = \lceil 4b\pi \rceil$.
		do $j = 0, N$
		  Determine ν_j, the nearest integer to $x_j mN/2\pi$
		  do $k = -q/2, q/2$
		  Compute Q_{jk} according to (65)
		  end do
		end do
		do $k = -N/2, N/2$
		  Compute $e^{b(2\pi k/mN)^2}$
		end do

1	$O(N)$	**Comment** [Input parameter is the complex vector $\{\beta_{-N/2}, \ldots, \beta_{N/2}\}$.] **Comment** [Compute new, scaled Fourier coefficients.] **do** $k = -N/2, N/2$ $u_k = \beta_k \cdot e^{b(2\pi k/mN)^2}$ **end do**
2	$O(mN \log N)$	**Comment** [Evaluate this Fourier Series at equispaced points in $[-\pi, \pi]$ using inverse FFT of size mN.] Compute $U_j = \sum_{k=-N/2}^{N/2} u_k \cdot e^{2\pi ikj/mN}$ for $j = -mN/2, \ldots, mN/2 - 1$.

3	$O(Nq)$	**Comment** [Compute approximate values at desired points in terms of the values at equispaced points.]
		do $j = 0, N$
		  do $k = -q/2, q/2$
		  $\tilde{g}_j \leftarrow \tilde{g}_j + Q_{jk} \cdot U_{\nu_j+k}$
		  end do
		end do

Total	$O(mN \cdot \log N + N \cdot \log(\frac{1}{\varepsilon}))$	

ALGORITHM 3.

Step	Complexity	Description
Init	$O(Nq)$	**Comment** [Input parameters are the vectors $\{\omega_0, \ldots, \omega_N\}$ and $\{x_0, \ldots, x_N\}$.] Choose precision ε to be achieved. Set $b \approx \log(1/\varepsilon)$ and $q = \lceil 4b\pi \rceil$. **do** $k = 0, N$ Determine μ_k, the nearest integer to $m\omega_k$ **do** $j = -q/2, q/2$ Compute P_{jk} according to (58) **end do** **end do** **do** $k = -mN/2, mN/2$ Compute $e^{b(2\pi k/m^2N)^2}$ **end do** **do** $j = 0, N$ Determine η_j, the nearest integer to $x_j N/2\pi$ **do** $k = -q/2, q/2$ Compute R_{jk} according to (71) **end do** **end do** **do** $j = 0, N$ Compute $e^{b(x_j/m)^2}$ **end do**
1	$O(Nq)$	**Comment** [Input parameter is the vector $\{\gamma_0, \ldots, \gamma_N\}$.] **Comment** [Compute Fourier coefficients v_j.] **do** $k = 0, N$ **do** $j = -q/2, q/2$ $v_{\mu_k+j} \leftarrow v_{\mu_k+j} + P_{jk} \cdot \gamma_k$ **end do** **end do**
2	$O(mN)$	**Comment** [Scale the coefficients.] **do** $k = -mN/2, mN/2$ $v_k \leftarrow v_k \cdot e^{b(2\pi k/m^2N)^2}$ **end do**
3	$O(m^2N \log N)$	**Comment** [Evaluate this Fourier Series at equispaced points in $[-m\pi, m\pi]$ using inverse FFT of size m^2N.] Compute $V_j = \sum_{k=-mN/2}^{mN/2} v_k \cdot e^{2\pi ikj/m^2N}$ for $j = -m^2N/2, \ldots, m^2N/2 - 1$.
4	$O(Nq)$	**Comment** [Compute approximate values at desired points in terms of the values at equispaced points.] **do** $j = 0, N$ **do** $k = -q/2, q/2$ $\tilde{h}_j \leftarrow \tilde{h}_j + R_{jk} \cdot V_{\eta_j+k}$ **end do** **end do**

5	$O(N)$	**Comment** [Scale the values.] **do** $j = 0, N$ $\tilde{h}_j \leftarrow \tilde{h}_j \cdot e^{b(x_j/m)^2}$ **end do**
Total	$O(m^2 N \cdot \log N + N \cdot \log(\frac{1}{\varepsilon}))$	

ALGORITHM 4.

Step	**Complexity**	**Description**
Init	$O(N \log N + Nq)$	Initialization for Algorithm 1. Initialization for Algorithm 2. Compute elements $\{a_k\}$ of Toeplitz matrix $(AA^*)^{-1}$ as defined by (80) using Algorithm 1.
1	$O(N \log N + Nq)$	Compute Af using Algorithm 2.
2	$O(\kappa(A) \cdot N \log N)$	Compute $\tilde{\alpha} = (AA^*)^{-1}(Af)$ using Conjugate Gradient algorithm.
Total	$O(\kappa(A) \cdot N \cdot \log N + N \cdot \log(\frac{1}{\varepsilon}))$	

ALGORITHM 5.

Step	**Complexity**	**Description**
Init	$O(N \log N + Nq)$	Initialization for Algorithm 1. Compute elements $\{a_k\}$ of Toeplitz matrix $(AA^*)^{-1}$ as defined by (80) using Algorithm 1.
1	$O(\kappa(A) \cdot N \log N)$	Compute $(AA^*)^{-1}g$ using Conjugate Gradient algorithm.
2	$O(N \log N + Nq)$	Compute $\tilde{\beta} = A^*((AA^*)^{-1}g)$ using Algorithm 1.
Total	$O(\kappa(A) \cdot N \cdot \log N + N \cdot \log(\frac{1}{\varepsilon}))$	

ALGORITHM 6.

Step	**Complexity**	**Description**
Init	$O(\kappa(A) \cdot N \log N + Nq)$	Initialization for Algorithm 5. Compute ξ as defined by (81) using Algorithm 5.
1	$O(N)$	Compute $\hat{g}_j = \xi_j g_j$ for $j = 0, \ldots, N$.
2	$O(N \log N + Nq)$	Compute $\tilde{\beta} = A^* \hat{g}$ using Algorithm 1.
Total	$O(N \cdot \log N + N \cdot \log(\frac{1}{\varepsilon}))$	

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

$$\lambda \cdot N \cdot q, \tag{83}$$

where the coefficient λ is software- and hardware-dependent.

7. Implementation and numerical results. We have written FORTRAN implementations of the six algorithms of this paper in both single and double precision arithmetic and have applied these programs to a variety of situations. In this section we discuss several details of our implementations and demonstrate the performance of the algorithms with six numerical examples.

Several technical details of our implementations appear to be worth mentioning here.

1. Each implementation consists of two main subroutines: the first is an initialization stage in which the matrix operators of the algorithm are precomputed and stored, and the second is an evaluation stage in which these operators are applied. Successive application of the linear transformations to multiple vectors requires the initialization to be performed only once.

2. For the single precision versions of each algorithm our choice of parameters was $m = 2$, $b = 0.5993$, and $q = 10$. For double precision we chose $m = 2$, $b = 1.5629$, and $q = 28$.

3. The algorithms as described in this paper all require an FFT of size proportional to N, and thus will perform efficiently whenever the FFT does. This restriction on the FFT size can be removed by extending the input vector to length $2^{\lceil \log_2 N \rceil}$ (i.e., the smallest power of 2 which is greater than N) and padding it with zeroes. This ensures that the algorithms will perform efficiently for any choice of N. In our implementations, these changes were made.

4. Each of the algorithms of this paper requires the evaluation of sums of the form

$$f_j = \sum_{k=-N/2}^{N/2-1} \alpha_k \cdot e^{2\pi i k j/N} \tag{84}$$

for $j = -N/2, \ldots, N/2 - 1$, whereas most FFT software computes sums of the form

$$f_j = \sum_{k=0}^{N-1} \alpha_k \cdot e^{2\pi i k j/N} \tag{85}$$

for $j = 0, \ldots, N - 1$. We used a standard FFT to evaluate sums of the form (84) by defining $\hat{\alpha}_k = \alpha_k$ for $k = 0, \ldots, N/2 - 1$, $\hat{\alpha}_k = \alpha_{k-N}$ for $k = N/2, \ldots, N - 1$, $\hat{f}_j = f_j$ for $j = 0, \ldots, N/2 - 1$, and $\hat{f}_j = f_{k-N}$ for $j = N/2, \ldots, N - 1$. This substitution converts the form (84) to the form (85).

Our implementations of the algorithms of this paper have been tested on the Sun SPARCstation 1 for a variety of input data. Six experiments are described below and their results are summarized in Tables 1–6. These tables contain accuracies and CPU timings for the algorithms with computations performed in both single and double precision arithmetic, and the input size N varying between 64 and 4096. In addition, each table contains the CPU times required to solve the same set of problems via a direct calculation, and Tables 1–3 include timings for an FFT of the same size. Tables 1–3 also contain the accuracies of the direct single precision calculations.

Two measures of accuracy were chosen for each example. In Examples 1, 2, and 3, these are defined by the formulae

$$E_\infty = \max_{0 \le j \le N} |\tilde{f}_j - f_j| \Big/ \sum_{j=0}^{N} |\alpha_j| \,, \tag{86}$$

and

$$E_2 = \sqrt{\sum_{j=0}^{N} |\tilde{f}_j - f_j|^2 \Big/ \sum_{j=0}^{N} |f_j|^2} \,, \tag{87}$$

where α is the input vector, f is the result of a direct computation in double precision arithmetic, and $\tilde{f}$ is the result of the computation being considered.

In Examples 4, 5, and 6, they are defined by

$$E_\infty = \max_{0\le j\le N} |\tilde{\alpha}_j - \alpha_j| \Big/ \max_{0\le j\le N} |\alpha_j| \tag{88}$$

and

$$E_2 = \sqrt{\sum_{j=0}^{N} |\tilde{\alpha}_j - \alpha_j|^2 \Big/ \sum_{j=0}^{N} |\alpha_j|^2}, \tag{89}$$

where α is the input for a direct double precision computation, and $\tilde{\alpha}$ is the result of applying the algorithm to the result of this computation.

Remark 7.1. The formulae (86)–(89) measure fairly accurately the errors of all single precision computations. However, they can only provide rough estimates of the errors produced by the double precision versions of the algorithms.

Remark 7.2. In the direct methods for Problems 1 and 2, we used the fact that $e^{ikx_j} = (e^{ix_j})^k$ to reduce the number of exponential computations from N^2 to N. N^2 exponentials are required in the direct method for Problem 3, and, for larger N, the available memory on the machine is insufficient for the precomputation and storage of these numbers. The direct implementation we used for this problem computes each exponential when it is needed.

Remark 7.3. Standard LINPACK Gaussian elimination subroutines were used as the direct methods for comparing timings in Examples 4, 5 and 6. Estimated timings are presented for larger N, where this computation became impractical.

Following are the descriptions of the experiments and the tables of numerical results.

Example 1. Here we consider the transformation $F : C^{N+1} \to C^{N+1}$ of Problem 1 as defined by the formula

$$F(\alpha)_j = \sum_{k=0}^{N} \alpha_k \cdot e^{i\omega_k \cdot 2\pi j/N} \tag{90}$$

for $j = -N/2, \ldots, N/2$. In this example, $\{\omega_0, \ldots, \omega_N\}$ were randomly distributed on the interval $[-N/2, N/2]$, and $\{\alpha_0, \ldots, \alpha_N\}$ were generated randomly on the unit square in the complex plane defined by the formulae

$$0 \le \mathrm{Re}(z) \le 1, \qquad 0 \le \mathrm{Im}(z) \le 1. \tag{91}$$

The results of applying Algorithm 1 to this problem are presented in Tables 1(a) and 1(b).

Example 2. Here we consider the transformation $G : \mathbf{C}^{N+1} \to \mathbf{C}^{N+1}$ of Problem 2 as defined by the formula

$$G(\beta)_j = \sum_{k=-N/2}^{N/2} \beta_k \cdot e^{ikx_j} \tag{92}$$

for $j = 0, \ldots, N$. In this example, $\{x_0, \ldots, x_N\}$ were randomly distributed on the interval $[-\pi, \pi]$, and $\{\beta_{-N/2}, \ldots, \beta_{N/2}\}$ were generated randomly on the unit square in the complex plane defined by the formulae

$$0 \le \mathrm{Re}(z) \le 1, \qquad 0 \le \mathrm{Im}(z) \le 1. \tag{93}$$

TABLE 1(a)
Example 1. *Single precision computations.*

N	Errors				Timings (sec.)			
	Algorithm		Direct		Algorithm		Direct	FFT
	E_∞	E_2	E_∞	E_2	Init.	Eval.		
64	0.959 E−06	0.149 E−05	0.411 E−06	0.979 E−06	0.011	0.003	0.01	0.0008
128	0.108 E−05	0.265 E−05	0.724 E−06	0.182 E−05	0.022	0.006	0.03	0.0015
256	0.122 E−05	0.329 E−05	0.996 E−06	0.370 E−05	0.044	0.014	0.13	0.0034
512	0.176 E−05	0.796 E−05	0.154 E−05	0.691 E−05	0.088	0.029	0.49	0.0078
1024	0.199 E−05	0.113 E−04	0.197 E−05	0.146 E−04	0.174	0.065	1.90	0.0174
2048	0.255 E−05	0.230 E−04	0.317 E−05	0.280 E−04	0.348	0.136	7.47	0.0352
4096	0.427 E−05	0.431 E−04	0.506 E−05	0.555 E−04	0.701	0.315	30.24	0.0846

TABLE 1(b)
Example 1. *Double precision computations.*

N	Errors		Timings (sec.)			
	E_∞	E_2	Alg. init.	Alg. eval.	Direct	FFT
64	0.602 E−14	0.638 E−14	0.036	0.008	0.02	0.001
128	0.356 E−14	0.715 E−14	0.075	0.016	0.06	0.002
256	0.437 E−14	0.946 E−14	0.148	0.034	0.22	0.005
512	0.519 E−14	0.160 E−13	0.297	0.075	0.84	0.012
1024	0.518 E−14	0.314 E−13	0.600	0.155	3.23	0.026
2048	0.755 E−14	0.631 E−13	1.204	0.322	12.55	0.059
4096	0.118 E−13	0.125 E−12	2.418	0.713	49.69	0.132

TABLE 2(a)
Example 2. *Single precision computations.*

N	Errors				Timings (sec.)			
	Algorithm		Direct		Algorithm		Direct	FFT
	E_∞	E_2	E_∞	E_2	Init.	Eval.		
64	0.870 E−06	0.176 E−05	0.147 E−06	0.298 E−06	0.011	0.003	0.01	0.0008
128	0.148 E−05	0.199 E−05	0.541 E−06	0.764 E−06	0.022	0.005	0.03	0.0015
256	0.780 E−06	0.349 E−05	0.414 E−06	0.114 E−05	0.043	0.012	0.12	0.0034
512	0.953 E−06	0.890 E−05	0.828 E−06	0.334 E−05	0.086	0.027	0.46	0.0078
1024	0.182 E−05	0.103 E−04	0.311 E−05	0.534 E−05	0.174	0.055	1.80	0.0174
2048	0.209 E−05	0.181 E−04	0.685 E−05	0.923 E−05	0.345	0.124	7.11	0.0352
4096	0.427 E−05	0.318 E−04	0.211 E−04	0.222 E−04	0.687	0.283	29.03	0.0846

TABLE 2(b)
Example 2. *Double precision compuations.*

N	Errors		Timings (sec.)			
	E_∞	E_2	Alg. init.	Alg. eval.	Direct	FFT
64	0.249 E−14	0.814 E−14	0.038	0.005	0.01	0.001
128	0.501 E−14	0.746 E−14	0.075	0.012	0.05	0.002
256	0.418 E−14	0.623 E−14	0.148	0.028	0.18	0.005
512	0.356 E−14	0.831 E−14	0.297	0.060	0.69	0.012
1024	0.793 E−14	0.192 E−13	0.596	0.126	2.72	0.026
2048	0.138 E−13	0.405 E−13	1.188	0.264	10.86	0.059
4096	0.278 E−13	0.904 E−13	2.387	0.573	43.36	0.132

The results of applying Algorithm 2 to this problem are presented in Tables 2(a) and 2(b).

Example 3. Here we consider the transformation $H : \mathbf{C}^{N+1} \to \mathbf{C}^{N+1}$ of Problem 3 as defined by the formula

$$H(\gamma)_j = \sum_{k=0}^{N} \gamma_k \cdot e^{i\omega_k x_j} \tag{94}$$

for $j = 0, \ldots, N$. In this example, $\{\omega_0, \ldots, \omega_N\}$ were randomly distributed on the interval $[-N/2, N/2]$, $\{x_0, \ldots, x_N\}$ were randomly distributed on the interval $[-\pi, \pi]$, and $\{\gamma_0, \ldots, \gamma_N\}$ were generated randomly on the unit square in the complex plane defined by the formulae

$$0 \le \mathrm{Re}(z) \le 1, \qquad 0 \le \mathrm{Im}(z) \le 1. \tag{95}$$

The results of applying Algorithm 3 to this problem are presented in Tables 3(a) and 3(b).

TABLE 3(a)
Example 3. *Single precision computations.*

N	Errors				Timings (sec.)			
	Algorithm		Direct		Algorithm		Direct	FFT
	E_∞	E_2	E_∞	E_2	Init.	Eval.		
64	0.133 E−05	0.232 E−05	0.341 E−06	0.673 E−06	0.022	0.007	0.12	0.0008
128	0.154 E−05	0.362 E−05	0.405 E−06	0.121 E−05	0.044	0.013	0.46	0.0015
256	0.152 E−05	0.550 E−05	0.736 E−06	0.292 E−05	0.087	0.029	1.90	0.0034
512	0.188 E−05	0.956 E−05	0.113 E−05	0.642 E−05	0.177	0.061	7.79	0.0078
1024	0.277 E−05	0.151 E−04	0.163 E−05	0.110 E−04	0.352	0.131	32.04	0.0174
2048	0.734 E−05	0.364 E−04	0.290 E−05	0.267 E−04	0.706	0.299	131.41	0.0352
4096	0.828 E−05	0.726 E−04	0.461 E−05	0.526 E−04	1.404	0.644	532.95	0.0846

TABLE 3(b)
Example 3. *Double precision computations.*

N	Errors		Timings (sec.)			
	E_∞	E_2	Alg. init.	Alg. eval.	Direct	FFT
64	0.166 E−13	0.226 E−13	0.074	0.015	0.20	0.001
128	0.252 E−13	0.216 E−13	0.153	0.034	0.79	0.002
256	0.318 E−13	0.315 E−13	0.302	0.069	3.18	0.005
512	0.131 E−13	0.289 E−13	0.601	0.146	12.76	0.012
1024	0.203 E−13	0.425 E−13	1.210	0.297	51.12	0.026
2048	0.324 E−13	0.801 E−13	2.403	0.643	205.17	0.059
4096	0.244 E−13	0.124 E−12	4.824	1.326	827.52	0.132

Example 4. Here we consider Problem 4 of §3. In this example, the numbers $\{\omega_k\}$ were defined by the formula

$$\omega_k = -\frac{N}{2} + (k + 0.5 + \delta_k) \cdot \frac{N}{N+1} \tag{96}$$

for $k = 0, \ldots, N$, where δ_k were randomly distributed on the interval $[-0.1, 0.1]$. In addition, the numbers $\{\alpha_0, \ldots, \alpha_N\}$ were generated randomly on the unit square in the complex plane defined by the formulae

$$0 \le \mathrm{Re}(z) \le 1, \qquad 0 \le \mathrm{Im}(z) \le 1, \tag{97}$$

and the numbers $\{f_{-N/2}, \ldots, f_{N/2}\}$ were computed directly in double precision arithmetic according to the formula

$$f_j = \sum_{k=0}^{N} \alpha_k \cdot e^{i\omega_k \cdot 2\pi j/N}. \tag{98}$$

The vector f was then used as input for Algorithm 4. Results of this experiment are presented in Tables 4(a) and 4(b).

TABLE 4(a)
Example 4. *Single precision computations.*

N	Errors		Timings (sec.)		
	E_∞	E_2	Alg. init.	Alg. eval.	Direct
64	0.492 E−05	0.339 E−05	0.02	0.05	0.36
128	0.154 E−04	0.568 E−05	0.04	0.11	2.78
256	0.424 E−04	0.128 E−04	0.08	0.20	23.0
512	0.809 E−04	0.252 E−04	0.18	0.45	184
1024	0.203 E−03	0.474 E−04	0.36	0.82	1472 (est.)
2048	0.428 E−03	0.979 E−04	0.78	1.91	11776 (est.)
4096	0.106 E−02	0.195 E−03	1.66	4.64	94208 (est.)

TABLE 4(b)
Example 4. *Double precision computations.*

N	Errors		Timings (sec.)		
	E_∞	E_2	Alg. init.	Alg. eval.	Direct
64	0.143 E−13	0.109 E−13	0.07	0.17	0.37
128	0.208 E−13	0.149 E−13	0.11	0.34	2.96
256	0.493 E−13	0.256 E−13	0.20	0.75	23.6
512	0.121 E−12	0.500 E−13	0.48	1.67	189
1024	0.279 E−12	0.926 E−13	0.94	3.55	1512 (est.)
2048	0.593 E−12	0.192 E−12	1.95	7.75	12096 (est.)
4096	0.138 E−11	0.375 E−12	4.02	18.28	96768 (est.)

Example 5. Here we consider Problem 5 of §3. In this example, the numbers $\{x_j\}$ were defined by the formula

$$x_j = -\pi + 2\pi \cdot \frac{j + 0.5 + \delta_j}{N + 1} \tag{99}$$

for $j = 0, \ldots, N$, where δ_j were randomly distributed on the interval $[-0.1, 0.1]$. In addition, the numbers $\{\beta_{-N/2}, \ldots, \beta_{N/2}\}$ were generated randomly on the unit square in the complex plane defined by the formulae

$$0 \le \mathrm{Re}(z) \le 1, \qquad 0 \le \mathrm{Im}(z) \le 1, \tag{100}$$

and the numbers $\{g_0, \ldots, g_N\}$ were computed directly in double precision arithmetic according to the formula

$$g_j = \sum_{k=-N/2}^{N/2} \beta_k \cdot e^{ikx_j}. \tag{101}$$

The vector g was then used as input for Algorithm 5. Results of this experiment are presented in Tables 5(a) and 5(b).

TABLE 5(a)
Example 5. *Single precision computations.*

N	Errors		Timings (sec.)		
	E_∞	E_2	Alg. init.	Alg. eval.	Direct
64	0.212 E−05	0.183 E−05	0.02	0.06	0.36
128	0.117 E−04	0.523 E−05	0.04	0.10	2.78
256	0.190 E−04	0.955 E−05	0.09	0.19	23.0
512	0.287 E−04	0.202 E−04	0.19	0.41	184
1024	0.560 E−04	0.376 E−04	0.37	0.83	1472 (est.)
2048	0.106 E−03	0.752 E−04	0.78	1.90	11776 (est.)
4096	0.225 E−03	0.148 E−03	1.66	4.67	94208 (est.)

TABLE 5(b)
Example 5. *Double precision computations.*

N	Errors		Timings (sec.)		
	E_∞	E_2	Alg. init.	Alg. eval.	Direct
64	0.310 E−13	0.120 E−13	0.06	0.18	0.37
128	0.389 E−13	0.146 E−14	0.10	0.36	2.96
256	0.577 E−13	0.204 E−13	0.24	0.76	23.6
512	0.673 E−13	0.325 E−13	0.47	1.61	189
1024	0.118 E−12	0.817 E−13	0.97	3.54	1512 (est.)
2048	0.190 E−12	0.134 E−12	1.86	7.73	12096 (est.)
4096	0.429 E−12	0.288 E−12	3.93	18.21	96768 (est.)

Example 6. Here we consider the variant of Problem 5 which was described in §4.3. In this example, the numbers $\{x_j\}$ were defined by the formula

$$x_j = -\pi + 2\pi \cdot \frac{j + 0.5 + \delta_j}{N+1} \tag{102}$$

for $j = 0, \ldots, N$, where δ_j were randomly distributed on the interval $[-0.1, 0.1]$. In addition, the numbers $\{\beta_{-N/4}, \ldots, \beta_{N/4}\}$ were generated randomly on the unit square in the complex plane defined by the formulae

$$0 \le \mathrm{Re}(z) \le 1, \qquad 0 \le \mathrm{Im}(z) \le 1, \tag{103}$$

and the numbers $\{g_0, \ldots, g_N\}$ were computed directly in double precision arithmetic according to the formula

$$g_j = \sum_{k=-N/4}^{N/4} \beta_k \cdot e^{ikx_j}. \tag{104}$$

The vector g was then used as input for Algorithm 6. Results of this experiment are presented in Tables 6(a) and 6(b).

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

1. The errors produced by Algorithms 1, 2, and 3 are comparable with those produced by the corresponding direct methods.

TABLE 6(a)
Example 6. *Single precision computations.*

N	Errors		Timings (sec.)		
	E_∞	E_2	Alg. init.	Alg. eval.	Direct
64	0.195 E−05	0.174 E−05	0.09	0.004	0.36
128	0.412 E−05	0.286 E−05	0.17	0.007	2.78
256	0.105 E−04	0.913 E−05	0.34	0.014	23.0
512	0.195 E−04	0.147 E−04	0.70	0.031	184
1024	0.474 E−04	0.320 E−04	1.41	0.066	1472 (est.)
2048	0.836 E−04	0.631 E−04	3.05	0.147	11776 (est.)
4096	0.173 E−03	0.135 E−03	7.22	0.330	94208 (est.)

TABLE 6(b)
Example 6. *Double precision computations.*

N	Errors		Timings (sec.)		
	E_∞	E_2	Alg. init.	Alg. eval.	Direct
64	0.878 E−14	0.762 E−14	0.32	0.008	0.37
128	0.102 E−13	0.981 E−14	0.62	0.018	2.96
256	0.213 E−13	0.180 E−13	1.25	0.039	23.6
512	0.444 E−13	0.376 E−13	2.64	0.083	189
1024	0.679 E−13	0.520 E−13	5.82	0.170	1512 (est.)
2048	0.151 E−12	0.115 E−12	12.37	0.359	12096 (est.)
4096	0.308 E−12	0.232 E−12	27.61	0.766	96768 (est.)

2. The timings for Algorithms 1 and 2 are similar, which is to be expected since Problem 2 is the adjoint of Problem 1. Algorithm 3 is about twice as costly, which is in agreement with the fact that it is a synthesis of Algorithms 1 and 2.

3. In single precision computations for this particular architecture, implementation and range of N, Algorithms 1 and 2 are less than four times as costly as an FFT of the same size. For double precision the ratio is roughly six. These ratios decrease as N increases.

4. The extrapolated break-even point of Algorithms 1 and 2 is at roughly $N = 32$ if the initialization time is ignored. If the initialization time is included, the break-even point is at $N = 256$. For Algorithm 3, the break-even points are at $N = 8$ without initialization and at $N = 32$ with initialization.

5. The timings for Algorithms 4 and 5 are similar as expected, since Problem 5 is the adjoint of Problem 4.

6. The break-even points of Algorithms 4 and 5 are at roughly $N = 32$. For Algorithm 6, the break-even points are at $N = 32$ if the initialization time is taken into account and at $N = 16$ if it is ignored.

7. Algorithms 1 and 2 tend to be slightly more accurate than their inverses, Algorithms 4 and 5.

8. The initialization for Algorithm 6 is computationally costly, but subsequent evaluations require much less CPU time than evaluations for Algorithm 5.

Remark 7.4. The CPU timings for Algorithms 1, 2, and 3 are independent of the particular distributions of ω and x, whereas the timings for Algorithms 4 and 5 are sensitive to the distributions of these vectors.

8. Generalizations and conclusions. The results of this paper can be generalized in the following ways.

1. Simple modifications to Algorithms 1, 2, and 3 will allow the efficient application of the linear transformations $F_1, G_1, H_1 : \mathbf{C}^{N+1} \to \mathbf{C}^{M+1}$ defined by

$$F_1(\alpha)_j = \sum_{k=0}^{N} \alpha_k \cdot e^{i\omega_k \cdot 2\pi j/M} \quad \text{for } j = -\frac{M}{2}, \ldots, \frac{M}{2}, \tag{105}$$

$$G_1(\beta)_j = \sum_{k=-N/2}^{N/2} \beta_k \cdot e^{ikx_j} \quad \text{for } j = 0, \ldots, M, \tag{106}$$

$$H_1(\gamma)_j = \sum_{k=0}^{N} \gamma_k \cdot e^{i\omega_k x_j} \quad \text{for } j = 0, \ldots, M. \tag{107}$$

These changes have been implemented.

2. The algorithms of this paper also assume that $\omega_k \in [-N/2, N/2]$ and $x_j \in [-\pi, \pi]$. Other distributions can be handled by appropriately partitioning the vectors ω and x, treating each partition separately and finally combining the results. The following observation describes translation operators which can be used for each partition, in combination with one of Algorithms 1, 2, or 3.

Observation 8.1. Let $a, b > 0, c, d > 0$ be real numbers and suppose that $\omega_k \in [a-b, a+b]$ for $k = 0, \ldots, N$ and $x_j \in [c-d, c+d]$ for $j = 0, \ldots, M$. Then we can write

$$\sum_{k=0}^{N} \alpha_k \cdot e^{i\omega_k x_j} = e^{iax_j} \cdot \sum_{k=0}^{N} \alpha_k \cdot e^{i(\omega_k - a)c} \cdot e^{i(\omega_k - a)(x_j - c)} \tag{108}$$

$$= e^{iax_j} \cdot \sum_{k=0}^{N} \alpha'_k \cdot e^{i\omega'_k x'_j}, \tag{109}$$

where

$$\begin{aligned} \alpha'_k &= \alpha_k e^{i(\omega_k - a)c}, \\ \omega'_k &= (\omega_k - a)d/\pi, \\ x'_j &= (x_j - c)\pi/d \in [-\pi, \pi]. \end{aligned} \tag{110}$$

Remark 8.2. Such an algorithm will perform efficiently when the points within a partition are close together and there are very few partitions and not so efficiently if the points are widely separated and there are many partitions. Most cases likely to be encountered in practice fall in the former category.

3. The algorithms of this paper are based on a special case of a more general idea, namely, the adaptive use of interpolation techniques to speed up large scale computations. Other examples of this approach include the use of wavelets for the construction of fast numerical algorithms (see, for example, [1], [3]) and the use of multipole or Chebyshev expansions for the compression of certain classes of linear operators (see, for example, [5], [11]).

4. A paper describing a set of algorithms based on a different interpolation technique is currently in preparation.

5. One of the more far-reaching extensions of the results of this paper is a set of algorithms for higher dimensional discrete Fourier transforms. Investigations into this are currently in progress.

6. The Helmholtz equation in two dimensions is given by

$$\nabla^2\phi + \kappa^2\phi = 0, \tag{111}$$

and has particular solutions of the form

$$\phi(x, y) = e^{i\mu x} \cdot e^{i\nu y}, \tag{112}$$

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

In conclusion, a group of algorithms has been presented for the rapid application and inversion of matrices of the Fourier kernel. These problems can be viewed as generalizations of the discrete Fourier transform, and the algorithms, while making use of certain simple results from analysis, are very versatile and have wide-ranging potential applications in many branches of mathematics, science, and engineering.

Appendix. In this Appendix we present numerical estimates of error bounds for Theorem 2.10 of §2.2. For our experiments we chose $c = 0$ and $d = \pi$ and chose the two sets of values of m, b, and q, which are used in our single and double precision implementations of the algorithms of this paper. The expression

$$r(x) = 1 - e^{bx^2} \cdot \sum_{k=-q/2}^{q/2} \rho_k \cdot e^{ikx}, \tag{113}$$

where ρ_k is defined by (18), was evaluated at $n = 1000$ equally spaced nodes $\{x_k\}$ in the interval $[-\frac{\pi}{m}, \frac{\pi}{m}]$, and the following three quantities were computed:

- the maximum absolute error E_∞ defined by the formula

$$E_\infty = \max_{1 \le k \le n} |r(x_k)|, \tag{114}$$

- the relative L_2 error E_2 defined by the formula

$$E_2 = \sqrt{\frac{\sum_{k=1}^{n} |r(x_k)|^2}{n}}, \tag{115}$$

- the error bound E_B of Theorem 2.10 defined by the formula

$$E_B = e^{-b\pi^2(1-1/m^2)} \cdot (4b + 9). \tag{116}$$

The results of these two experiments are presented in Table 7.

TABLE 7

m	b	q	E_∞	E_2	E_B
2	0.5993	10	0.825 E−05	0.176 E−05	0.135 E−00
2	1.5629	28	0.400 E−13	0.580 E−14	0.163 E−03

We observe from Table 7 that the error bound E_B of Theorem 2.10 is very weak compared with the experimentally obtained bounds. Indeed, the requirement that E_B be appropriately small would impose much larger values of b and q than are actually needed for the algorithms.

REFERENCES

[1] B. ALPERT, G. BEYLKIN, R. COIFMAN, AND V. ROKHLIN, *Wavelet-like bases for the fast solution of second-kind integral equations*, SIAM J. Sci. Comput., 14 (1993) pp. 159–184.

[2] D. H. BAILEY AND P. N. SWARZTRAUBER, *The fractional fast Fourier transform and applications*, SIAM Rev., 33 (1991), pp. 389–404.

[3] G. BEYLKIN, R. COIFMAN, AND V. ROKHLIN, *Fast wavelet transforms and numerical algorithms* I, Comm. Pure Appl. Math., 44 (1991), pp. 141–183.

[4] E. O. BRIGHAM, *The Fast Fourier Transform and its Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[5] J. CARRIER, L. GREENGARD, AND V. ROKHLIN, *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 669–686.

[6] J. W. COOLEY AND J. W. TUKEY, *An algorithm for the machine computation of complex Fourier series*, Math. Comp., 19 (1965), pp. 297–301.

[7] G. DAHLQUIST AND A. BJORCK, *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[8] A. DUTT, *A Fast Algorithm for the Evaluation of Trigonometric Series*, Tech. Rep. 841, Yale Computer Science Dept., Yale Univ., New Haven, CT, 1991.

[9] D. GOTTLIEB, M. Y. HUSSAINI, AND S. ORSZAG, in *Spectral Methods for Partial Differential Equations*, R. G. Voigt, D. Gottlieb, and M. Y. Hussaini, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1984, p. 1.

[10] I. S. GRADSHTEYN AND I. M. RYZHIK, *Table of Integrals, Series and Products*, Academic Press, New York, 1980.

[11] V. ROKHLIN, *A fast algorithm for the discrete Laplace transformation*, J. Complexity, 4 (1988), pp. 12–32.

[12] J. STOER AND R. BULIRSCH, *Introduction to Numerical Analysis*, Springer-Verlag, New York, 1980.

[13] C. VAN LOAN, *Computational Frameworks for the Fast Fourier Transform*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[14] H. J. WEAVER, *Theory of Discrete and Continuous Fourier Analysis*, Wiley, New York, 1989.