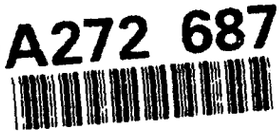


Fast Fourier Transforms for Nonequispaced Data

Alok Dutt

YALEU/CSD/RR #981
August, 1993

Abstract
Fast Fourier Transforms for Nonequispaced Data

Alok Dutt
Yale University
1993

Two groups of algorithms are presented generalizing the fast Fourier transform (FFT) to the case of non-integer frequencies and nonequispaced nodes on the interval $[-\pi, \pi]$. These schemes are based on combinations of certain analytical considerations with the classical fast Fourier transform, and generalize both the forward and backward FFTs. The number of arithmetic operations required by each of the algorithms is proportional to $N \cdot \log N + N \cdot \log(1/\varepsilon)$, where ε is the desired precision of computations and N is the number of nodes. Several related algorithms are also presented, each of which utilizes a similar set of techniques from analysis and linear algebra. These include an efficient version of the Fast Multipole Method in one dimension and fast algorithms for the evaluation, integration and differentiation of Lagrange polynomial interpolants. Several numerical examples are used to illustrate the efficiency of the approach, and to compare the performances of the two sets of nonuniform FFT algorithms.

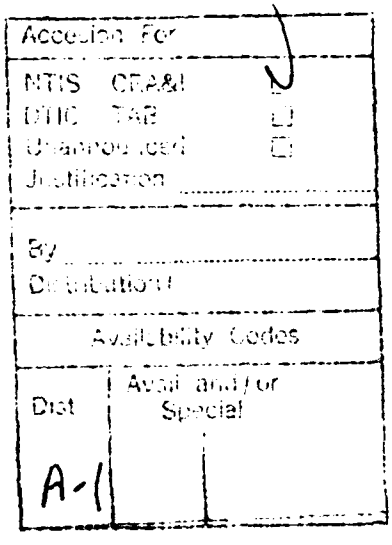
The work of this author was supported in part by the Office of Naval Research under Grant N00014-89-J-1527 and in part by the National Science Foundation under Grant DMS9012751.
Approved for public release: distribution is unlimited.

Keywords: *FFT, Trigonometric Series, Fourier Analysis, Interpolation, Fast Multipole Method, Approximation Theory*

Fast Fourier Transforms for Nonequispaced Data

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

by
Alok Dutt
May, 1993



To my family, my friends and my teachers.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Fourier techniques have been a popular analytical tool in physics and engineering for more than two centuries. A reason for this popularity is that the trigonometric functions $e^{i\omega x}$ are eigenfunctions of the differentiation operator and thus form a natural basis for representing solutions of many classes of differential equations.

More recently, the arrival of digital computers and the development of the fast Fourier transform (FFT) algorithm in the 1960s (see [8]) have established Fourier analysis as a powerful and practical numerical tool. The FFT, which computes discrete Fourier transforms (DFTs), is now a central component in many scientific and engineering applications, most notably in the areas of spectral analysis and signal processing. Numerous applications, however, involve unevenly spaced data, whereas the FFT requires input data to be tabulated on a uniform grid. In this dissertation, a collection of algorithms is presented which overcome this limitation of the FFT while preserving its computational efficiency. These algorithms are designed for the efficient computation of certain generalizations of the DFT, namely the forward and inverse transformations described by the equations

$$f_j = \sum_{k=0}^{N} \alpha_k \cdot e^{i\omega_k x_j} \tag{1.1}$$

for $j = 0, \ldots, N$, where $f_j \in \mathbf{C}$, $\alpha_k \in \mathbf{C}$, $\omega_k \in [-N/2, N/2]$ and $x_j \in [-\pi, \pi]$. The number of arithmetic operations required by each of the algorithms is proportional to

$$N \cdot \log N + N \cdot \log\left(\frac{1}{\varepsilon}\right) \tag{1.2}$$

where ε is the desired accuracy, compared with $O(N^2)$ operations required for the direct evaluation of (1.1) and $O(N^3)$ for the direct inversion.

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

$$f_j = \sum_{k=0}^{N-1} \alpha_k \cdot e^{2\pi ikj/N} \tag{1.3}$$

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

$$f_j = \sum_{k=-N/2}^{N/2-1} \alpha_k \cdot e^{2\pi ikj/N} \tag{1.4}$$

for $j = -N/2, \ldots, N/2-1$. While the form (1.3) is normally used when the FFT is discussed, the form (1.4) is usually preferred in applications of the DFT to analysis (see, for example, [3], [12]).

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 [5], [8], [9], [23], [24], [25]). In the more general case of (1.1), the structure of the linear transformation can also be exploited via a combination of certain analytical results with the FFT, and the algorithms of this thesis are based on this fact.

Remark 1.3 All algorithms described in this thesis are approximate ones in the sense that they compute a vector $\tilde{f}$ such that

$$\frac{||\tilde{f} - f||}{||f||} \leq \varepsilon, \tag{1.5}$$

where the vector f is the exact result of the calculation. We will derive error bounds for all approximations used by the algorithms, thereby allowing us to perform numerical computations to any specified accuracy ε.

A somewhat extensive literature is devoted to the development, improvement and implementation of classical FFT schemes, whereas the numerical analysis of generalizations of the DFT appears to be relatively incomplete. The problem of interpolating Fourier series from irregular to regular grids has arisen in numerous areas, computer assisted tomography to mention just one. Traditional attempts to speed up computations for problems of this type combined linear or other low-order polynomial interpolation methods with filtering techniques. The effectiveness of this approach is limited in part by the low accuracy of the interpolation scheme used (see, for example, [6], [20], [21]). Another generalization of the DFT, referred to as the Fractional Fourier Transform, is addressed by Bailey and Swarztrauber [3] who describe an algorithm for efficiently computing the numbers

$$f_j = \sum_{k=0}^{N-1} \alpha_k \cdot e^{2\pi ikj\lambda/N} \tag{1.6}$$

for $j = 0, \ldots, N-1$, where λ is any complex constant. However, this algorithm does not permit unevenly spaced input data.

1.1 Outline of the Dissertation

The principal results of this thesis are two groups of efficient algorithms for computing FFTs for nonuniformly spaced data to any specified precision. In addition the thesis contains a number of secondary results upon which the main algorithms are based. Portions of this work have been published previously [10], [11].

One of the groups of nonuniform FFT algorithms views the problem as a form of polynomial interpolation which can be solved efficiently utilizing a fast multipole-type method in one dimension. These algorithms require some preliminary results which are presented in Chapter 2. In this chapter, the problem of evaluating the Lagrange interpolating polynomial on the real line is discussed. An efficient Fast Multipole Method (FMM) for one dimensional problems is first developed whose computational cost is $O(N)$ arithmetic operations. Following this is a description of an adaptive version of this algorithm which requires $O(N)$ operations independently of the distribution of the points on the real line. The chapter ends by discussing how these FMM algorithms can be used for the efficient evaluation, integration and differentiation of Lagrange polynomial interpolants.

In Chapter 3 we describe a set of four algorithms for nonuniform FFTs which are based on a combination of the classical FFT with a scheme for the rapid evaluation of trigonometric interpolants. This interpolation scheme is based on the one dimensional FMM of Chapter 2, and is used to interpolate from uniform to nonuniform grids and back.

A second set of algorithms for nonuniform FFTs is presented in Chapter 4. This set of algorithms also uses the classical FFT, but uses a different interpolation technique based on the Fourier analysis of the Gaussian bell.

Each of the algorithms described in this thesis has been implemented and tested for a variety of problem sizes and input data. Chapter 5 contains details of these implementations and results of several numerical experiments to demonstrate the performance of the two groups of nonuniform FFT algorithms.

Finally, Chapter 6 lists several generalizations and conclusions.

Chapter 2

Polynomial Interpolation and the FMM

Polynomials form the theoretical basis for many areas of applied mathematics. While the mathematical properties of polynomials have been quite well understood for over a century, attempts to use them in practical calculations have met with difficulties. Typical problems accompanying the employment of classical schemes are those of prohibitive computational cost and numerical instability. However, certain classes of orthogonal polynomials do have stable, fast transforms associated with them, the most popular being Chebyshev polynomials which can be manipulated in a stable and efficient manner via the fast cosine transform or FFT.

In this chapter we present a group of three algorithms for the interpolation, integration and differentiation of functions tabulated at nodes other than Chebyshev. The first of these algorithms takes as input a set of points, $\{x_1, \ldots, x_N\}$, and a set of function values, $\{f_1, \ldots, f_N\}$, and evaluates the unique interpolating polynomial of degree $N-1$ at the points $\{y_1, \ldots, y_N\}$ for a computational cost proportional to N. The other two algorithms perform spectral integration and differentiation of this interpolant. We will also describe three efficient versions of the Fast Multipole Method (FMM) for one dimensional problems; these algorithms are used by the polynomial interpolation algorithm of this chapter. Throughout the chapter we will be using the well known Lagrange representation of interpolating polynomials which is defined by the formula

$$P_N(x) = \sum_{j=1}^{N} f_j \cdot \prod_{\substack{k=1 \\ k \neq j}}^{N} \frac{x - x_k}{x_j - x_k}. \tag{2.1}$$

A simple algebraic manipulation converts (2.1) to the form

$$P_N(x) = \prod_{k=1}^{N}(x - x_k) \cdot \sum_{j=1}^{N} \frac{f_j}{x - x_j} \cdot \prod_{\substack{k=1 \\ k \neq j}}^{N} \frac{1}{x_j - x_k}, \tag{2.2}$$

which can be evaluated at N points in

$$O\left(N \cdot \log\left(\frac{1}{\varepsilon}\right)\right) \tag{2.3}$$

arithmetic operations using the Fast Multipole Method (FMM) of [16] (ε here is the desired accuracy). In comparison, a direct evaluation of (2.2) requires $O(N^2)$ operations.

Remark 2.1 A somewhat different classical polynomial interpolation problem consists of determining a set of parameters $\{a_1, \ldots, a_N\}$ such that, for $j = 1, \ldots, N$,

$$\sum_{k=1}^{N} a_k \cdot x_j^{k-1} = f_j \tag{2.4}$$

where $\{x_1, \ldots, x_N\}$ is a given a set of points and $\{f_1, \ldots, f_N\}$ is a given set of function values. This problem is highly ill-posed for anything other than very small values of N and this formulation is seldom used when actual calculations are being performed.

Remark 2.2 The Lagrange interpolation formula has traditionally been less favored for practical calculations than other classical methods (see, for example, [23]). However, the algorithms of this chapter are numerically stable and very efficient, as demonstrated by our numerical experiments, thus affording the Lagrange formula a substantial advantage over other techniques for the manipulation of polynomials.

Following is a plan of the chapter. The first three sections are devoted to efficient versions of the FMM which can be used to evaluate expressions of the form (2.2): Section 2.1 contains a number of results from analysis which are used in the design of these algorithms, in Section 2.2 we present the FMM algorithm itself, and in Section 2.3 we present an adaptive version of this algorithm. A brief discussion of versions of the FMM for other kernels is contained in Section 2.4. A Fast Polynomial Interpolation algorithm utilizing the results of the previous sections is described in Section 2.5, and finally, in Section 2.6 we describe how this algorithm can be used to construct fast algorithms for high order numerical integration and differentiation.

2.1 Mathematical and Numerical Preliminaries

The algorithms of this chapter are based on several results from the Chebyshev approximation theory of the function $1/x$. This analysis is presented in the Lemmas and Theorems of this section, numbered 2.1-2.10. The main results of this section fall into two categories. Theorems 2.5 and 2.7 describe how the function $1/x$ can be approximated on different regions of the real line using Chebyshev expansions, and Theorems 2.8, 2.9 and 2.10 provide three ways of manipulating these expansions which are needed by the fast algorithms of this chapter.

We begin with three classical definitions which can be found, for example, in [14], [23].

Definition 2.1 *The n-th degree Chebyshev polynomial $T_n(x)$ is defined by the following equivalent formulae:*

$$T_n(x) = \cos(n \arccos x), \tag{2.5}$$

$$T_n(x) = \frac{1}{2} \cdot \left((x + \sqrt{x^2 - 1})^n + (x - \sqrt{x^2 - 1})^n \right). \tag{2.6}$$

Definition 2.2 *The roots $\{t_1, \ldots, t_n\}$ of the n-th degree Chebyshev polynomial T_n lie in the interval $[-1, 1]$ and are defined by the formulae*

$$t_k = -\cos\left(\frac{2k-1}{n} \cdot \frac{\pi}{2} \right) \tag{2.7}$$

for $k = 1, \ldots, n$. They are referred to as Chebyshev nodes of order n.

Definition 2.3 *We will define the polynomials $u_1, \ldots, u_n$ of order $n-1$ by the formulae*

$$u_j(t) = \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{t - t_k}{t_j - t_k} \tag{2.8}$$

for $j = 1, \ldots, n$, where t_k are defined by (2.7).

The order $n-1$ Chebyshev approximation for a function $f : [-1, 1] \to \mathbf{C}$ is defined as the unique polynomial of order $n-1$ which agrees with f at the nodes $t_1, \ldots, t_n$. There exist several standard representations for this polynomial, and the one we will use in this chapter is given by the expression

$$\sum_{j=1}^{n} f(t_j) \cdot u_j(t). \tag{2.9}$$

For the purposes of this chapter, Chebyshev expansions for any function will be characterized by values of this function tabulated at Chebyshev nodes.

Lemmas 2.1–2.3 provide estimates involving Chebyshev expansions which are used in the remainder of this section. The proof of Lemma 2.1 is obvious from (2.5).

Lemma 2.1 *Let $T_n(x)$ be the Chebyshev polynomial of degree n. Then,*

$$|T_n(x)| \leq 1 \tag{2.10}$$

for any $x \in [-1, 1]$.

Lemma 2.2 *Let $T_n(x)$ be the Chebyshev polynomial of degree n. Then,*

$$|T_n(x)| > \frac{1}{2} \cdot \left|\frac{5x}{3}\right|^n \tag{2.11}$$

for any x such that $|x| \geq 3$.

Proof. From Definition 2.1, we have

$$\begin{aligned} |T_n(x)| &= \frac{1}{2} \cdot \left|(x + \sqrt{x^2 - 1})^n + (x - \sqrt{x^2 - 1})^n\right| \\ &> \frac{1}{2} \cdot |x + \sqrt{x^2 - (x/3)^2}|^n = \frac{1}{2} \cdot |x \cdot (1 + \sqrt{8/9})|^n \\ &> \frac{1}{2} \cdot \left|\frac{5x}{3}\right|^n \end{aligned} \tag{2.12}$$

for any x such that $|x| \geq 3$. □

Lemma 2.3 *Let $u_j(x)$ be defined by (2.8) for $j = 1, \ldots, n$. Then, for any $x \in [-1, 1]$,*

$$|u_j(x)| \leq 1. \tag{2.13}$$

Proof It is obvious from (2.8) that $u_j(t_j) = 1$, and that $u_j(t_k) = 0$ when $k \neq j$. In addition, some algebraic manipulation reveals that the expression

$$\frac{1}{n} \sum_{k=1}^{n} T_k(t_j) \cdot T_k(x) \tag{2.14}$$

is also equal to 1 when $x = t_j$ and equal to 0 when $x = t_k$ for all other k. Since u_j and (2.14) are both polynomials of order $n - 1$, we have

$$u_j(x) = \frac{1}{n} \sum_{k=1}^{n} T_k(t_j) \cdot T_k(x) \tag{2.15}$$

for $j = 1, \ldots, n$. Furthermore, due to the combination of (2.15) and the triangle inequality, we have

$$|u_j(x)| = \left| \frac{1}{n} \sum_{k=1}^{n} T_k(t_j) \cdot T_k(x) \right| \leq \frac{1}{n} \sum_{k=1}^{n} |T_k(t_j)| \cdot |T_k(x)| \leq 1 \tag{2.16}$$

for any $x \in [-1, 1]$. □

The following lemma provides an identity which is used in the proof of Theorem 2.5.

Lemma 2.4 *Suppose that $n \geq 2$, and that $b > 0$ and x_0 are real numbers with $|x_0| \geq 3b$. Then, for all x,*

$$1 - (x - x_0) \cdot \sum_{j=1}^{n} \frac{1}{bt_j - x_0} \cdot u_j\left(\frac{x}{b}\right) = \frac{T_n(x/b)}{T_n(x_0/b)}. \tag{2.17}$$

Proof. Let $Q(x)$ by the polynomial of degree n defined by the formula

$$Q(x) = 1 - (x - x_0) \cdot \sum_{j=1}^{n} \frac{1}{bt_j - x_0} \cdot u_j\left(\frac{x}{b}\right). \tag{2.18}$$

Using (2.8) we obtain

$$\begin{aligned} Q(bt_k) &= 1 - (bt_k - x_0) \cdot \sum_{j=1}^{n} \frac{1}{bt_j - x_0} \cdot u_j(t_k) \\ &= 1 - (bt_k - x_0) \cdot \frac{1}{bt_k - x_0} = 0, \end{aligned} \tag{2.19}$$

for $k = 1, \ldots, n$. Clearly, then, $Q(x)$ satisfies the conditions

$$\begin{aligned} Q(x_0) &= 1 \\ Q(bt_1) &= 0 \\ &\vdots \\ Q(bt_n) &= 0. \end{aligned} \tag{2.20}$$

It is clear that the function $T_n(x/b)/T_n(x_0/b)$ is also a polynomial of degree n which satisfies the $n + 1$ conditions (2.20). Therefore,

$$Q(x) \equiv \frac{T_n(x/b)}{T_n(x_0/b)}, \tag{2.21}$$

and (2.17) follows as an immediate consequence of (2.18) and (2.21). □

Theorem 2.5 *Suppose that $n \geq 2$, and that $b > 0$ and x_0 are real numbers with $|x_0| \geq 3b$. Then,*

$$\left| \frac{1}{x - x_0} - \sum_{j=1}^{n} \frac{1}{bt_j - x_0} \cdot u_j\left(\frac{x}{b}\right) \right| < \frac{1}{b \cdot 5^n} \tag{2.22}$$

for any $x \in [-b, b]$.

Proof. Due to Lemma 2.4, we have

$$\left| \frac{1}{x - x_0} - \sum_{j=1}^{n} \frac{1}{bt_j - x_0} \cdot u_j\left(\frac{x}{b}\right) \right| = \frac{1}{|x - x_0|} \cdot \frac{|T_n(x/b)|}{|T_n(x_0/b)|}. \tag{2.23}$$

Also, due to Lemmas 2.1 and 2.2, we have

$$|T_n(x/b)| \leq 1 \tag{2.24}$$

for any $x \in [-b, b]$, and

$$|T_n(x_0/b)| > \frac{1}{2} \cdot \left|\frac{5x_0}{3b}\right|^n > \frac{5^n}{2} \tag{2.25}$$

for any $|x_0| \geq 3b$. It follows from the combination of (2.23), (2.24) and (2.25) that

$$\left| \frac{1}{x - x_0} - \sum_{j=1}^{n} \frac{1}{bt_j - x_0} \cdot u_j\left(\frac{x}{b}\right) \right| \leq \frac{1}{2b} \cdot \frac{2}{5^n} \leq \frac{1}{b \cdot 5^n} \tag{2.26}$$

for any $x \in [-b, b]$. □

The following lemma provides an identity which is used in the proof of Theorem 2.7.

Lemma 2.6 *Suppose that $n \geq 2$, and that $b > 0$ and x_0 are real numbers with $|x_0| \leq b$. Then, for all ξ,*

$$\xi - (3b - \xi x_0) \cdot \sum_{j=1}^{n} \frac{t_j}{3b - t_j x_0} \cdot u_j(\xi) = \frac{3b}{x_0} \cdot \frac{T_n(\xi)}{T_n(3b/x_0)}. \tag{2.27}$$

Proof. Let $Q(\xi)$ be the polynomial of degree n defined by the formula

$$Q(\xi) = \xi - (3b - \xi x_0) \cdot \sum_{j=1}^{n} \frac{t_j}{3b - t_j x_0} \cdot u_j(\xi). \tag{2.28}$$

Using (2.8) we obtain

$$\begin{aligned} Q(t_k) &= t_k - (3b - t_k x_0) \cdot \sum_{j=1}^{n} \frac{t_j}{3b - t_j x_0} \cdot u_j(t_k) \\ &= t_k - (3b - t_k x_0) \cdot \frac{t_k}{3b - t_k x_0} = 0, \end{aligned} \tag{2.29}$$

for $k = 1, \ldots, n$. Clearly, then, $Q(\xi)$ satisfies the conditions

$$\begin{aligned} Q(3b/x_0) &= 3b/x_0 \\ Q(t_1) &= 0 \\ &\vdots \\ Q(t_n) &= 0. \end{aligned} \tag{2.30}$$

It is clear that the function

$$\frac{3b}{x_0} \cdot \frac{T_n(\xi)}{T_n(3b/x_0)} \tag{2.31}$$

is also a polynomial of degree n which satisfies the $n+1$ conditions (2.30). Therefore,

$$Q(\xi) \equiv \frac{3b}{x_0} \cdot \frac{T_n(\xi)}{T_n(3b/x_0)}, \tag{2.32}$$

and (2.27) follows as an immediate consequence of (2.28) and (2.32). □

Theorem 2.7 *Suppose that $n \geq 2$, and that $b > 0$ and x_0 are real numbers with $|x_0| \leq b$. Then,*

$$\left| \frac{1}{x - x_0} - \sum_{j=1}^{n} \frac{t_j}{3b - t_j x_0} \cdot u_j\left(\frac{3b}{x}\right) \right| < \frac{3}{b \cdot 5^n} \tag{2.33}$$

for any x such that $|x| \geq 3b$.

Proof. Writing $\xi = 3b/x$, we have $|\xi| \leq 1$ whenever $|x| \geq 3b$, and

$$\frac{1}{x - x_0} = \frac{1}{3b/\xi - x_0} = \frac{\xi}{3b - \xi x_0}. \tag{2.34}$$

Due to Lemma 2.6, we have

$$\left| \frac{\xi}{3b - \xi x_0} - \sum_{j=1}^{n} \frac{t_j}{3b - t_j x_0} \cdot u_j(\xi) \right| = \frac{1}{|3b - \xi x_0|} \cdot \frac{3b}{|x_0|} \cdot \frac{|T_n(\xi)|}{|T_n(3b/x_0)|}. \tag{2.35}$$

In addition, due to Lemmas 2.1 and 2.2 we have

$$3b \cdot |T_n(\xi)| \leq 3b \tag{2.36}$$

for $\xi \in [-1,1]$, and

$$|x_0 \cdot T_n(3b/x_0)| > \frac{|x_0|}{2} \cdot \left|\frac{5 \cdot 3b}{3x_0}\right|^n > \frac{5^n \cdot b}{2} \tag{2.37}$$

for $|x_0| \leq b$. Substituting (2.36) and (2.37) into (2.35), we obtain

$$\left|\frac{\xi}{3b - \xi x_0} - \sum_{j=1}^{n} \frac{t_j}{3b - t_j x_0} \cdot u_j(\xi)\right| < \frac{1}{2b} \cdot \frac{3b \cdot 2}{5^n \cdot b} = \frac{3}{b \cdot 5^n} \tag{2.38}$$

for $\xi \in [-1,1]$. Now it follows from the combination of (2.34) and (2.38) that

$$\left|\frac{1}{x - x_0} - \sum_{j=1}^{n} \frac{t_j}{3b - t_j x_0} \cdot u_j\left(\frac{3b}{x}\right)\right| < \frac{3}{b \cdot 5^n}. \tag{2.39}$$

□

The following three theorems provide formulae for translating along the real axis Chebyshev expansions of the type described in the previous two theorems. Theorem 2.8 provides a formula for translating expansions described in Theorems 2.5, Theorem 2.9 describes a mechanism of converting the expansion of Theorem 2.5 to the expansion of Theorem 2.7, and Theorem 2.10 provides a way of translating the expansion of Theorem 2.7. These theorems are one-dimensional counterparts of the three theorems in [16] which provide translation operators for multipole expansions in the complex plane.

Theorem 2.8 *Suppose that $n \geq 2$ and let c, d be a pair of real numbers such that $[c-d, c+d] \subset [-1,1]$. Then, for any set of complex numbers $\Psi_1, \ldots, \Psi_n$, and any $x \in [c-d, c+d]$,*

$$\sum_{j=1}^{n} \Psi_j \cdot u_j(x) = \sum_{k=1}^{n} \left(\tilde{\Psi}_k \cdot u_j(c + dt_k)\right) \cdot u_j\left(\frac{x-c}{d}\right). \tag{2.40}$$

Proof. To prove this theorem we first show that

$$u_j(x) = \sum_{k=1}^{n} u_j(c + dt_k) \cdot u_k\left(\frac{x-c}{d}\right) \tag{2.41}$$

for $x \in [c-d, c+d]$. Indeed, the right hand side of (2.41) is simply the $(n-1)$-th degree Lagrange interpolating polynomial for the function $u_j(x)$ at the nodes $c + dt_1, \ldots, c + dt_n$. However, $u_j(x)$ itself is a polynomial of degree $n-1$, and is therefore equal to its Lagrange interpolant of order $n-1$.

The formula (2.40) then follows as an immediate consequence of (2.41). □

Theorem 2.9 *Suppose that $n, N \geq 2$ and let c, d be a pair of real numbers such that $|c| - d > 3$. Let the function $f : \mathbf{R} \to \mathbf{C}$ be defined by the formula*

$$f(x) = \sum_{k=1}^{N} \frac{\alpha_k}{x - x_k} \tag{2.42}$$

where $x_k \in [-1, 1]$ for $k = 1, \ldots, N$, and $\alpha_1, \ldots, \alpha_N$ is a set of complex numbers. Further, let $t_1, \ldots, t_n$ be Chebyshev nodes defined by (2.7), let $\Phi_1, \ldots, \Phi_n$ be a set of complex numbers defined by the formula

$$\Phi_k = f\left(\frac{3}{t_k}\right) \tag{2.43}$$

for $k = 1, \ldots, n$, and let $\Psi_1, \ldots, \Psi_n$ be a set of complex numbers defined by the formula

$$\Psi_k = \sum_{j=1}^{n} \Phi_j \cdot u_j\left(\frac{3}{c + dt_k}\right). \tag{2.44}$$

Then, for any $x \in [c-d, c+d]$,

$$\left| f(x) - \sum_{k=1}^{n} \Psi_k \cdot u_j\left(\frac{x-c}{d}\right) \right| < A \cdot \frac{3n+1}{b \cdot 5^n}, \tag{2.45}$$

where $A = \sum_{k=1}^{N} |\alpha_k|$.

Proof. Due to the triangle inequality,

$$\left| f(x) - \sum_{k=1}^{n} \Psi_k \cdot u_j\left(\frac{x-c}{d}\right) \right| \leq S_1 + S_2, \tag{2.46}$$

where

$$S_1 = \left| f(x) - \sum_{k=1}^{n} f(c + dt_k) \cdot u_j\left(\frac{x-c}{d}\right) \right|, \tag{2.47}$$

and

$$S_2 = \left| \sum_{k=1}^{n} (f(c + dt_k) - \Psi_k) \cdot u_j\left(\frac{x-c}{d}\right) \right|. \tag{2.48}$$

Combining Theorem 2.5 with the triangle inequality we obtain

$$S_1 < \frac{A}{b \cdot 5^n}, \tag{2.49}$$

and from the combination of Theorem 2.7, Lemma 2.3 and the triangle inequality, we obtain

$$S_2 \le \sum_{k=1}^{n} \left| f(c + dt_k) - \sum_{j=1}^{n} \Phi_j \cdot u_j \left(\frac{3}{c + dt_k} \right) \right| < \frac{3An}{b \cdot 5^n}, \tag{2.50}$$

where $A = \sum_{k=1}^{N} |\alpha_k|$. Finally, substituting (2.49) and (2.50) into (2.46) we have

$$\left| f(x) - \sum_{k=1}^{n} \Psi_k \cdot u_j \left(\frac{x - c}{d} \right) \right| < A \cdot \frac{3n + 1}{b \cdot 5^n} \tag{2.51}$$

for any $x \in [c - d, c + d]$. □

Theorem 2.10 *Suppose that $n, N \ge 2$ and let c, d be a pair of real numbers such that $[c - d, c + d] \supset [-1, 1]$. Let the function $f : \mathbf{R} \to \mathbf{C}$ be defined by the formula*

$$f(x) = \sum_{k=1}^{N} \frac{\alpha_k}{x - x_k} \tag{2.52}$$

where $x_k \in [-1, 1]$ for $k = 1, \ldots, N$, and $\alpha_1, \ldots, \alpha_N$ is a set of complex numbers. Further, let $t_1, \ldots, t_n$ be Chebyshev nodes defined by (2.7), let $\Phi_1, \ldots, \Phi_n$ be a set of complex numbers defined by the formula

$$\Phi_k = f\left(\frac{3}{t_k} \right) \tag{2.53}$$

for $k = 1, \ldots, n$, and let $\tilde{\Phi}_1, \ldots, \tilde{\Phi}_n$ be a set of complex numbers defined by the formula

$$\tilde{\Phi}_k = \sum_{j=1}^{n} \Phi_j \cdot u_j \left(\frac{t_k}{3d + ct_k} \right). \tag{2.54}$$

Then, for any x such that $|x - c| \ge 3d$,

$$\left| f(x) - \sum_{k=1}^{n} \tilde{\Phi}_k \cdot u_j \left(\frac{3d}{x - c} \right) \right| < A \cdot \frac{3(n + 1)}{b \cdot 5^n}, \tag{2.55}$$

where $A = \sum_{k=1}^{N} |\alpha_k|$.

Proof. It follows from the triangle inequality that

$$\left| f(x) - \sum_{k=1}^{n} \tilde{\Phi}_k \cdot u_j\left(\frac{3d}{x-c}\right)\right| \leq S_1 + S_2 \tag{2.56}$$

where

$$S_1 = \left| f(x) - \sum_{k=1}^{n} f\left(c + \frac{3d}{t_k}\right) \cdot u_j\left(\frac{3d}{x-c}\right)\right|, \tag{2.57}$$

and

$$S_2 = \left| \sum_{k=1}^{n} \left(f\left(c + \frac{3d}{t_k}\right) - \tilde{\Phi}_k \right) \cdot u_j\left(\frac{3d}{x-c}\right)\right|. \tag{2.58}$$

Combining Theorem 2.7 with the triangle inequality, we obtain

$$S_1 < \frac{3A}{b \cdot 5^n}, \tag{2.59}$$

and from the combination of Theorem 2.7, Lemma 2.3 and the triangle inequality, we obtain

$$S_2 \leq \sum_{k=1}^{n} \left| f\left(c + \frac{3d}{t_k}\right) - \sum_{j=1}^{n} \Phi_j \cdot u_j\left(\frac{t_k}{3d + ct_k}\right)\right| < \frac{3An}{b \cdot 5^n}, \tag{2.60}$$

where $A = \sum_{k=1}^{N} |\alpha_k|$. Finally, substituting (2.59) and (2.60) into (2.56) we have

$$\left| f(x) - \sum_{k=1}^{n} \tilde{\Phi}_k \cdot u_j\left(\frac{3d}{x-c}\right)\right| < A \cdot \frac{3(n+1)}{b \cdot 5^n} \tag{2.61}$$

for any x such that $|x - c| \geq 3d$. □

2.2 The Fast Multipole Method in One Dimension

In this section we consider the problem of computing the sums

$$f_j = \sum_{k=1}^{N} \frac{\alpha_k}{y_j - x_k} \tag{2.62}$$

for $j = 1, \ldots, N$, where $\{x_1, \ldots, x_N\}$ and $\{y_1, \ldots, y_N\}$ are sets of real numbers, and $\{\alpha_1, \ldots, \alpha_N\}$ and $\{f_1, \ldots, f_N\}$ are sets of complex numbers. This problem can be viewed as the special case of the N-body problem in physics where we wish to evaluate the electrostatic field due to N charges which lie on a straight line at a set of points on this line.

Remark 2.3 For the remainder of this chapter, we shall assume without loss of generality that $x_i, y_i \in [-1, 1]$ for $i = 1, \ldots, N$.

Remark 2.4 The fast multipole algorithm of Greengard and Rokhlin [16] computes sums of a more general form than (2.62) in $O(N)$ arithmetic operations. This more general form is described by the formulae

$$f_j = \sum_{k=1}^{N} \frac{\alpha_k}{w_j - z_k} \tag{2.63}$$

for $j = 1, \ldots, N$, where $\{z_1, \ldots, z_N\}$ and $\{w_1, \ldots, w_N\}$ are sets of complex numbers. From a physical viewpoint, this corresponds to the evaluation of the electrostatic field due to N charges which lie in the plane. While the two and three dimensional scenarios for the N-body problem have been discussed in some detail (see, for example, [7], [16]), the analysis and applications of one dimensional problems appear to have been largely overlooked, with one exception of which we are aware: the application of FMM techniques to various problems in numerical linear algebra, by Gu and Eisenstat (see [17], [18]).

In this section we present an $O(N)$ algorithm for the one-dimensional problem which is based on the two-dimensional FMM, but incorporates a number of modifications which accelerate the scheme significantly. We summarize these modifications below, with the assumption that the reader is familiar with [16].

1. Replacement of all complex by complex multiplications with real by complex multiplications.

2. Replacement of multipole expansions with Chebyshev expansions. Chebyshev series are known to converge more rapidly than multipole expansions (which are actually Taylor series).

3. Further compression of the Chebyshev expansions by a suitable change of basis (see Section 2.2.4). Using this technique, the function $1/x$ can be accurately represented by about a quarter of the number of coefficients which were required by the original two-dimensional FMM.

4. In one dimension, each subinterval has 3 other subintervals in its interaction list, whereas in two dimensions each box has 27 other boxes in its interaction list.

5. In one dimension, each subinterval has 2 nearest neighbor subintervals, whereas in two dimensions each box has 8 nearest neighbor boxes.

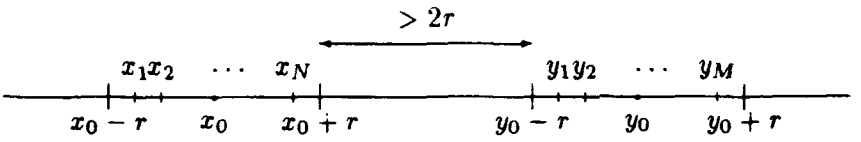


Figure 2.1: Well-separated intervals on the line.

This section is divided into four parts. Sections 2.2.1-2.2.3 are devoted to an algorithm for the FMM in one dimension which uses Chebyshev expansions in place of multipole expansions. In Section 2.2.4, we describe a more efficient algorithm which is based on the algorithm of Section 2.2.3 but uses a Singular Value Decomposition to further compress the Chebyshev expansions.

2.2.1 General Strategy

We will illustrate by means of a simple example how Chebyshev expansions can be used to evaluate expressions of the form (2.62) more efficiently. We will also give an informal description of how the method of this simple example is used in the construction of a fast algorithm for the general case.

First we introduce a definition which formalizes the notion of well-separated intervals on the real line. This is simply the one-dimensional analog of the definition of well-separatedness in [16].

Definition 2.4 *Let* $\{x_1, \ldots, x_N\}$ *and* $\{y_1, \ldots, y_M\}$ *be two sets of points in* $\mathbf{R}$. *We say that the sets* $\{x_i\}$ *and* $\{y_i\}$ *are* well-separated *if there exist points* $x_0, y_0 \in \mathbf{R}$ *and a real* $r > 0$ *such that*

$$\begin{aligned} &|x_i - x_0| < r \quad \forall \ \ i = 1, \ldots, N \\ &|y_i - y_0| < r \quad \forall \ \ i = 1, \ldots, M \text{ and} \\ &|x_0 - y_0| > 4r. \end{aligned} \tag{2.64}$$

Suppose now that $\{x_1, \ldots, x_N\}$ and $\{y_1, \ldots, y_M\}$ are well-separated sets of points in $\mathbf{R}$ (see Figure 2.1), that $\{\alpha_1, \ldots, \alpha_N\}$ is a set of complex numbers, and that we wish to compute the numbers $f(y_1), \ldots, f(y_M)$ where the function $f : \mathbf{R} \rightarrow \mathbf{C}$ is defined by the formula

$$f(x) = \sum_{k=1}^{N} \frac{\alpha_k}{x - x_k}. \tag{2.65}$$

A direct evaluation of (2.65) at the points $\{y_1, \ldots, y_M\}$ requires $O(NM)$ arithmetic operations. We will describe two different ways of speeding up this computation.

Following is the first of these approaches.

Let the function $\tilde{f}_1 : \mathbf{R} \to \mathbf{C}$ be defined by the formula

$$\tilde{f}_1(x) = \sum_{k=1}^{N} \alpha_k \cdot \sum_{j=1}^{p} \frac{t_j}{3r - t_j(x_k - x_0)} \cdot u_j \left(\frac{3r}{x - x_0} \right), \tag{2.66}$$

where p is an integer and $t_1, \ldots, t_p$ and $u_1, \ldots, u_p$ are given by Definitions 2.2 and 2.3.

Observation 2.5 *From the combination of (2.65), (2.66), Theorem 2.7 and the triangle inequality, we see that*

$$\left| f(x) - \tilde{f}_1(x) \right| < \frac{3 \cdot \sum_{k=1}^{N} |\alpha_k|}{r \cdot 5^p} \tag{2.67}$$

for any x such that $|x - x_0| > 3r$.

Now let $\{\Phi_1, \ldots, \Phi_p\}$ be a set of complex numbers defined by the formulae

$$\Phi_j = \sum_{k=1}^{N} \alpha_k \cdot \frac{t_j}{3r - t_j(x_k - x_0)} \tag{2.68}$$

for $j = 1, \ldots, p$. Then,

$$\tilde{f}_1(x) = \sum_{j=1}^{p} \Phi_j \cdot u_j \left(\frac{3r}{x - x_0} \right). \tag{2.69}$$

Remark 2.6 The vector Φ will be referred to as the far-field expansion for the interval $[x_0 - r, x_0 + r]$.

Computation of the coefficients Φ_j requires $O(Np)$ operations, and a subsequent evaluation of $\tilde{f}_1(y_1), \ldots, \tilde{f}_1(y_M)$ is an $O(Mp)$ procedure. The total computational cost of approximating (2.65) to a relative precision $1/5^p$ is then $O(Np + Mp)$ operations.

An alternative way of speeding up this computation is described below.

Let the function $\tilde{f}_2 : \mathbf{R} \to \mathbf{C}$ be defined by the formula

$$\tilde{f}_2(x) = \sum_{k=1}^{N} \alpha_k \cdot \sum_{j=1}^{p} \frac{1}{rt_j - (x_k - y_0)} \cdot u_j \left(\frac{x - y_0}{r} \right), \tag{2.70}$$

where p is an integer and $t_1, \ldots, t_p$ and $u_1, \ldots, u_p$ are given by Definitions 2.2 and 2.3.

Observation 2.7 *From the combination of (2.65), (2.70), Theorem 2.5 and the triangle inequality, we see that*

$$\left|f(x) - \tilde{f}_2(x)\right| < \frac{\sum_{k=1}^{N} |\alpha_k|}{r \cdot 5^p} \tag{2.71}$$

for any x such that $|x - y_0| < r$.

Now let $\{\Psi_1, \ldots, \Psi_p\}$ be a set of complex numbers defined by the formulae

$$\Psi_j = \sum_{k=1}^{N} \alpha_k \cdot \frac{1}{rt_j - (x_k - x_0)}. \tag{2.72}$$

for $j = 1, \ldots, p$. Then,

$$\tilde{f}_2(x) = \sum_{j=1}^{p} \Psi_j \cdot u_j \left(\frac{x - y_0}{r}\right). \tag{2.73}$$

Remark 2.8 The vector Ψ will be referred to as the local expansion for the interval $[y_0 - r, y_0 + r]$.

Computation of the coefficients Ψ_j requires $O(Np)$ operations, and a subsequent evaluation of $\tilde{f}_2(y_1), \ldots, \tilde{f}_2(y_M)$ is an $O(Mp)$ procedure. Again the total computational cost of approximating (2.65) to a relative precision $1/5^p$ is $O(Np + Mp)$ operations.

Consider now the general case, where the points $\{x_1, \ldots, x_N\}$ and $\{y_1, \ldots, y_M\}$ are arbitrarily distributed on the interval $[-1, 1]$ (see Remark 2.3). We use a hierarchy of grids to subdivide the computational domain $[-1, 1]$ into progressively smaller subintervals, and to subdivide the sets $\{x_i\}$ and $\{y_i\}$ according to subinterval (see Figure 2.2). A tree structure is imposed on this hierarchy, so that the two subintervals resulting from the bisection of a larger (parent) interval are referred to as its children. Two Chebyshev expansions are associated with each subinterval: a far-field expansion for the points within the subinterval, and a local expansion for the points which are well-separated from the subinterval. Interactions between pairs of well-separated subintervals can be computed via these Chebyshev expansions in the manner described above, and all other interactions at the finest level can be computed directly. Once the precision has been fixed, the computational cost of the entire procedure is $O(N)$ operations (for a detailed description and complexity analysis, see Section 2.2.3).

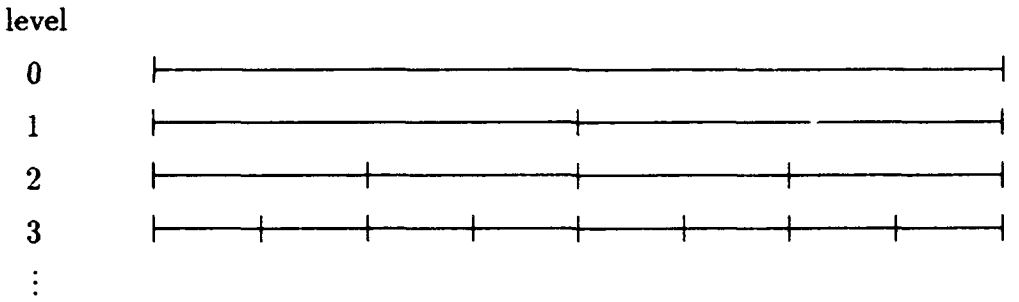


Figure 2.2: Hierarchy of subintervals.

2.2.2 Notation

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

- p will be an integer denoting the size of Chebyshev expansions used by the algorithm. Normally, $p = \lceil -\log_5(\varepsilon) \rceil$, where ε is the desired precision of computations.
- $t_1, \ldots, t_p$ will denote Chebyshev nodes of order p on the interval $[-1, 1]$, defined by the formulae

$$t_i = \cos\left(\frac{2i-1}{p} \cdot \frac{\pi}{2}\right) \tag{2.74}$$

  for $i = 1, \ldots, p$.
- $u_1(t), \ldots, u_p(t)$ will denote the set of polynomials defined by the formulae

$$u_j(t) = \prod_{\substack{k=1 \\ k \neq j}}^{p} \frac{t - t_k}{t_j - t_k}, \tag{2.75}$$

  for $j = 1, \ldots, p$.
- s will be an integer denoting the number of points in each subinterval at the finest level of subdivision. Normally, $s \approx 2p$ (see Remark 2.9).
- $nlevs = \lceil \log_2(N/s) \rceil$ will denote the level of finest subdivision of the interval $[-1, 1]$.
- $\Phi_{l,i}$ will be a p-vector denoting the far-field expansion for the subinterval i at level l.

- $\Psi_{l,i}$ will be a p-vector denoting the local expansion for the subinterval i at level l.

- M_L and M_R will be $p \times p$ matrices for obtaining far-field expansions for subintervals in terms of the far-field expansions of their children. These will be defined by the formulae

$$\begin{aligned} M_L(i,j) &= u_j\left(\frac{t_i}{2+t_i}\right), \\ M_R(i,j) &= u_j\left(\frac{t_i}{2-t_i}\right), \end{aligned} \tag{2.76}$$

  which are obtained from the formula (2.54) of Theorem 2.10 applied to the cases $c=-1, d=2$ and $c=1, d=2$.

- S_L and S_R will be $p \times p$ matrices for obtaining local expansions for subintervals in terms of the local expansions of their parent. These will be defined by the formulae

$$\begin{aligned} S_L(i,j) &= u_j\left(\frac{t_i-1}{2}\right), \\ S_R(i,j) &= u_j\left(\frac{t_i+1}{2}\right). \end{aligned} \tag{2.77}$$

  which are obtained from the formula (2.40) of Theorem 2.8 applied to the cases $c=-\frac{1}{2}, d=\frac{1}{2}$ and $c=\frac{1}{2}, d=\frac{1}{2}$.

- T_1, T_2, T_3 and T_4 will be $p \times p$ matrices for obtaining local expansions from far-field expansions. These will be defined by the formulae

$$\begin{aligned} T_1(i,j) &= u_j\left(\frac{3}{t_i-6}\right), \\ T_2(i,j) &= u_j\left(\frac{3}{t_i-4}\right), \\ T_3(i,j) &= u_j\left(\frac{3}{t_i+4}\right), \\ T_4(i,j) &= u_j\left(\frac{3}{t_i+6}\right). \end{aligned} \tag{2.78}$$

  which are obtained from the formula (2.44) of Theorem 2.9 applied to the cases $c=-6, d=1$, $c=-4, d=1$, $c=4, d=1$ and $c=6, d=1$.

2.2.3 Description of the Algorithm

Following is a formal description of an algorithm for the efficient evaluation of expressions of the form (2.62).

Algorithm 2.1

Step	Complexity	Description
1	$O(1)$	**Comment** [Input problem size, N, and a real number $\varepsilon > 0$.] Set size of Chebyshev expansions $p = \lceil -\log_5(\varepsilon) \rceil$, choose s and set level of refinement $nlevs = \lceil \log_2(N/s) \rceil$.
2	$O(Np)$	**Comment** [Determine far-field expansions for subintervals at finest level.] **do** $i = 1, \ldots, 2^{nlevs}$ Compute p-term far-field expansion $\Phi_{nlevs,i}$ due to the subset of $\{x_j\}$ which lie in subinterval i at level $nlevs$. **enddo**
3	$O(2Np^2/s)$	**Comment** [Determine p-term far-field expansions for each subinterval at every level by shifting and adding far-field expansions of the subinterval's children.] **do** $l = nlevs - 1, \ldots, 1$ **do** $i = 1, \ldots, 2^l$ $\Phi_{l,i} = M_L \cdot \Phi_{l+1,2i-1} + M_R \cdot \Phi_{l+1,2i}$ **enddo** **enddo**
4	$O(8Np^2/s)$	**Comment** [Determine p-term local expansions for each subinterval at every level by first shifting local expansion of the subinterval's parent, and then adding the interactions with the three subintervals which are well-separated from the subinterval, but which have not been accounted for at the parent's level.] **do** $l = 1, \ldots, nlevs - 1$ **do** $i = 1, \ldots, 2^l$ $\Psi_{l+1,2i-1} = S_L \cdot \Psi_{l,i} + T_1 \cdot \Phi_{l+1,2i-3} + T_3 \cdot \Phi_{l+1,2i+1} + T_4 \cdot \Phi_{l+1,2i+2}$ $\Psi_{l+1,2i} = S_R \cdot \Psi_{l,i} + T_1 \cdot \Phi_{l+1,2i-3} + T_2 \cdot \Phi_{l+1,2i-2} + T_4 \cdot \Phi_{l+1,2i+2}$ **enddo** **enddo**

5 $O(Np)$ **Comment** [Evaluate local expansions for subintervals at finest level.]

do $i = 1, \ldots, 2^{nlevs}$
Evaluate p-term local expansions $\Psi_{nlevs,i}$ at the subset of $\{y_j\}$ which lie in subinterval i at level $nlevs$.
enddo

6 $O(3Ns)$ **Comment** [Add nearest neighbour interactions which have not yet been accounted for via expansions.]

do $i = 1, \ldots, 2^{nlevs}$
For each y_k in subinterval i at level $nlevs$, compute interactions with all x_j in subintervals $i-1, i, i+1$, and add to well-separated values.
enddo

Total $O(N \cdot (2p + 10p^2/s + 3s))$

Remark 2.9 The number s can be chosen to minimize the total operation count of the algorithm, which yields $s \approx 2p$. The above algorithm then requires order

$$13 \cdot N \cdot p \tag{2.79}$$

arithmetic operations.

Remark 2.10 The operation count for Step 6 assumes that the points $\{x_1, \ldots, x_N\}$ are reasonably uniformly distributed. Highly nonuniform distributions are discussed in Section 2.3.

2.2.4 A More Efficient Algorithm

Chebyshev expansions are not the most efficient means of representing interactions between well-separated intervals. All the matrix operators of the algorithm are numerically rank-deficient, and can be further compressed by a suitable change of basis. The orthogonal matrices required for this basis change are obtained via singular value decompositions (SVDs) of appropriate matrices.

In this subsection we describe a more efficient version of the one dimensional FMM which is based upon this observation. The algorithm is very similar to Algorithm 2.1 with one important modification: separate changes of basis are needed for interactions from the left and interactions from the right, so for every subinterval we maintain two sets of expansions, leftward expansions and rightward expansions.

We will require some additional notation for the algorithm description of this section. The following denotations use the notation of Section 2.2.2.

- $\bar{p}$ will be an integer denoting the numerical rank of the operators to be compressed.

- t_j^L will denote the j-th Chebyshev node of order p on the interval $[0,1]$, and t_j^R will denote the j-th Chebyshev node of order p on the interval $[-1,0]$.

- U_L and V_L will denote $p \times \bar{p}$ matrices, each of whose columns forms an orthonormal set, and Σ will denote a $\bar{p} \times \bar{p}$ diagonal matrix such that

$$||U_L \Sigma V_L^T - \mathcal{M}_L|| < \varepsilon \cdot ||\mathcal{M}_L||, \tag{2.80}$$

  where

$$\mathcal{M}_L(i,j) = \frac{1}{3/t_j^L - t_i}, \tag{2.81}$$

  for $i,j = 1,\ldots,p$. U_L and V_L are used to compress leftward expansions, and $U_L \Sigma V_L^T$ is effectively the numerical SVD of $\mathcal{M}_L$.

- U_R and V_R will denote $p \times \bar{p}$ matrices, each of whose columns forms an orthonormal set, such that

$$||U_R \Sigma V_R^T - \mathcal{M}_R|| < \varepsilon \cdot ||\mathcal{M}_R||, \tag{2.82}$$

  where

$$\mathcal{M}_R(i,j) = \frac{1}{3/t_j^R - t_i}, \tag{2.83}$$

  for $i,j = 1,\ldots,p$. An examination of the matrices $\mathcal{M}_L$ and $\mathcal{M}_R$ reveals that U_R and U_L are closely related, and V_R and V_L are closely related. U_R and V_R are used to compress rightward expansions, and $U_R \Sigma V_R^T$ is effectively the numerical SVD of $\mathcal{M}_R$.

- $\Phi_{l,i}^L$ and $\Phi_{l,i}^R$ will be $\bar{p}$-vectors denoting respectively the left and right far-field expansions for subinterval i at level l. These will be defined by the formulae:

$$\begin{aligned} \Phi_{l,i}^L &= U_L^T \cdot \Phi_{l,i}, \\ \Phi_{l,i}^R &= U_R^T \cdot \Phi_{l,i}. \end{aligned} \tag{2.84}$$

- $\Psi_{l,i}^L$ and $\Psi_{l,i}^R$ will be $\bar{p}$-vectors denoting respectively the left and right local expansions for subinterval i at level l. These will be defined by the formulae:

$$\begin{aligned} \Psi_{l,i}^L &= V_L^T \cdot \Psi_{l,i}, \\ \Psi_{l,i}^R &= V_R^T \cdot \Psi_{l,i}. \end{aligned} \tag{2.85}$$

- M_L^L, M_R^L, M_L^R and M_R^R will be $\bar{p} \times \bar{p}$ matrices for obtaining left and right far-field expansions for subintervals in terms of the left and right far-field expansions of their children. These will be defined by the formulae

$$\begin{aligned} M_L^L &= U_L^T \cdot M_L \cdot U_L, \\ M_R^L &= U_L^T \cdot M_R \cdot U_L, \\ M_L^R &= U_R^T \cdot M_L \cdot U_R, \\ M_R^R &= U_R^T \cdot M_R \cdot U_R. \end{aligned} \tag{2.86}$$

- S_L^L, S_R^L, S_L^R and S_R^R will be $\bar{p} \times \bar{p}$ matrices for obtaining left and right local expansions for subintervals in terms of the left and right local expansions of their parent. These will be defined by the formulae

$$\begin{aligned} S_L^L &= V_L^T \cdot S_L \cdot V_L, \\ S_R^L &= V_L^T \cdot S_R \cdot V_L, \\ S_L^R &= V_R^T \cdot S_L \cdot V_R, \\ S_R^R &= V_R^T \cdot S_R \cdot V_R. \end{aligned} \tag{2.87}$$

- T_1^L and T_2^L will be $\bar{p} \times \bar{p}$ matrices for obtaining left local expansions from right far-field expansions. These will be defined by the formulae

$$\begin{aligned} T_1^L &= V_L^T \cdot T_3 \cdot U_L, \\ T_2^L &= V_L^T \cdot T_4 \cdot U_L. \end{aligned} \tag{2.88}$$

- T_1^R and T_2^R will be $\bar{p} \times \bar{p}$ matrices denoting for obtaining right local expansions from left far-field expansions. These will be defined by the form lae

$$\begin{aligned} T_1^R &= V_R^T \cdot T_2 \cdot U_R, \\ T_2^R &= V_R^T \cdot T_1 \cdot U_R. \end{aligned} \tag{2.89}$$

Following is a step-by-step description of a modified version of Algorithm 2.1.

Algorithm 2.2

Step	Complexity	Description
1	$O(1)$	**Comment** [Input problem size, N, and a real number $\varepsilon > 0$.] Set size of Chebyshev expansions $p = \lceil -\log_5(\varepsilon) \rceil$, choose $\bar{p}$, choose s and set level of refinement $nlevs = \lceil \log_2(N/s) \rceil$.

2 $O(2N\bar{p})$ **do** $i = 1, \ldots, 2^{nlevs}$
Form a $\bar{p}$-term left far-field expansion $\Phi^L_{nlevs,i}$.
Form a $\bar{p}$-term right far-field expansion $\Phi^R_{nlevs,i}$.
enddo

3 $O(4N\bar{p}^2/s)$ **do** $l = nlevs - 1, \ldots, 1$
do $i = 1, \ldots, 2^l$
$\Phi^L_{l,i} = M^L_L \cdot \Phi^L_{l+1,2i-1} + M^L_R \cdot \Phi^L_{l+1,2i}$
$\Phi^R_{l,i} = M^R_L \cdot \Phi^R_{l+1,2i-1} + M^R_R \cdot \Phi^R_{l+1,2i}$
enddo
enddo

4 $O(10N\bar{p}^2/s)$ **do** $l = 1, \ldots, nlevs - 1$
do $i = 1, \ldots, 2^l$
$\Psi^R_{l+1,2i-1} = S^R_L \cdot \Psi^R_{l,i} + T^R_1 \cdot \Phi^L_{l+1,2i+1} + T^R_2 \cdot \Phi^L_{l+1,2i+2}$
$\Psi^R_{l+1,2i} = S^R_R \cdot \Psi^R_{l,i} + T^R_1 \cdot \Phi^L_{l+1,2i+2}$
$\Psi^L_{l+1,2i-1} = S^L_L \cdot \Psi^L_{l,i} + T^L_1 \cdot \Phi^R_{l+1,2i-3}$
$\Psi^L_{l+1,2i} = S^L_R \cdot \Psi^L_{l,i} + T^L_1 \cdot \Phi^R_{l+1,2i-2} + T^L_2 \cdot \Phi^R_{l+1,2i-3}$
enddo
enddo

5 $O(2N\bar{p})$ **do** $i = 1, \ldots, 2^{nlevs}$
Evaluate and add left and right $\bar{p}$-term local expansions $\Psi^L_{nlevs,i} + \Psi^R_{nlevs,i}$.
enddo

6 $O(3Ns)$ **do** $i = 1, \ldots, 2^{nlevs}$
For each point in subinterval i at level $nlevs$, compute
interactions with all other points in subintervals $i-1, i, i+1$,
and add to far-field values.
enddo

Total $O(N \cdot (4\bar{p} + 14\bar{p}^2/s + 3s))$

Remark 2.11 We can choose s to minimize the total operation count of the algorithm, which yields $s \approx 2\bar{p}$. The above algorithm then requires order

$$17 \cdot N \cdot \bar{p} \tag{2.90}$$

arithmetic operations.

Remark 2.12 The results of our numerical experiments indicate that, for a fixed precision ε and corresponding p, $\bar{p}$ (the numerical rank of the matrix operators to

be compressed) is approximately $p/2$. This condition together with (2.79) and (2.90) leads us to expect that Algorithm 2.2 will require about two-thirds of the number of arithmetic operations needed by Algorithm 2.1.

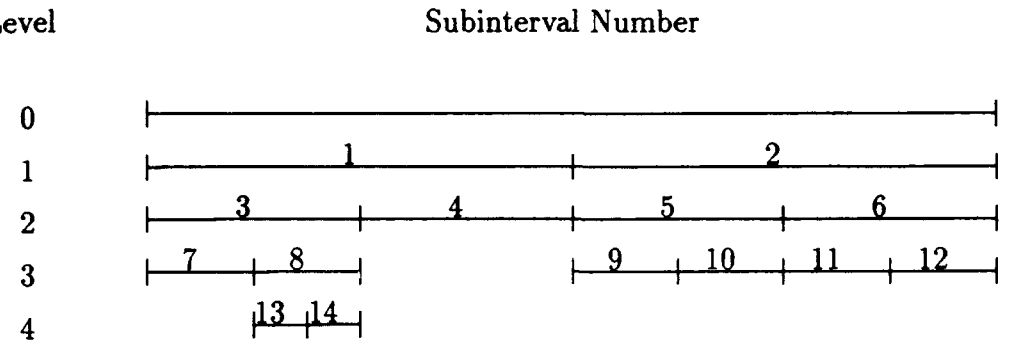
Remark 2.13 The operation count for Step 5 assumes that the points $\{x_1, \ldots, x_N\}$ are reasonably uniformly distributed. An adaptive version of this algorithm is described in Section 2.3, and is capable of handling highly nonuniform distributions while preserving computational efficiency and accuracy.

2.3 The Adaptive FMM in One Dimension

The algorithms of the previous section have one drawback: their operation count is quite sensitive to the distribution of points, and they become inefficient for highly nonuniform distributions. We now describe an adaptive version of Algorithm 2.2 which overcomes this deficiency, its complexity being $O(N)$ independently of the spacing of the nodes. This versatility is achieved by using different levels of subdivision for different parts of the computational domain. An integer s is fixed, and at each level of refinement, we subdivide only those intervals which contain more than s points. At each level, then, a list of *non-empty* subintervals is maintained whose members are the result of the selective subdivision of intervals at the previous level. This policy eliminates the inefficiency of the non-adaptive version, where, at the finest level of subdivision, we may encounter intervals with very few or very many points. In the non-adaptive version, each interval has two nearest neighbors of the same size, whereas in the adaptive version, intervals are permitted to have neighbors of differing size. Figure 2.3 depicts a subdivision of the computational domain for a nonuniform distribution of points.

Remark 2.14 The idea of selectively subdividing the computational domain is taken from the two dimensional adaptive FMM of [7]. This algorithm requires a somewhat more elaborate data structure than its non-adaptive counterpart to account for interactions between all the different sized boxes. The adaptive algorithm for problems in one dimension also needs additional bookkeeping to keep track of interactions between all the different sized subintervals. However, the simplified geometry of the real line suggests the use of a somewhat more efficient data structure than that of the two dimensional case (see Figure 2.3).

Remark 2.15 It is clear that for a fixed machine precision, ε, only certain distributions of points will yield meaningful results. For example, the points x_1 and x_2 are indistinguishable if $|x_2 - x_1| < \frac{\varepsilon}{2} \cdot |x_1 + x_2|$. To avoid such cases we will impose that the minimum distance between two points must be greater than $\varepsilon \cdot (b-a)$ where $[a, b]$



Interval	Neighbor		Child	
	Left	Right	Left	Right
1	-	2	3	4
2	1	-	5	6
3	-	4	7	8
4	3	5	-	-
5	4	6	9	10
6	5	-	11	12
7	-	8	-	-
8	7	4	13	14
9	4	10	-	-
10	9	11	-	-
11	10	12	-	-
12	11	-	-	-
13	7	14	-	-
14	13	4	-	-

Figure 2.3: Hierarchy of subintervals for nonuniform distribution.

is the computational domain. Under this condition, the highest level of refinement of the computational domain is bounded above by $|\log_2(\varepsilon)|$.

2.3.1 Notation

We require some additional notation for the description of the adaptive algorithm to supplement that of Sections 2.2.2 and 2.2.4.

- *nlevs* will denote the level of finest subdivision of any part of the interval $[-1, 1]$.
- For a fixed precision, ε, $m = |\log_2(\varepsilon)|$ will denote the maximum level of refinement of the interval $[-1, 1]$ (see Remark 2.15).
- I_l will denote the set of non-empty subintervals at level l, i.e. the set of subintervals at level l resulting from the bisection of a larger interval at level $l-1$.
- If subinterval *isub* contains more than s points, it is called a parent subinterval, and *ilchild*(*isub*) and *irchild*(*isub*) will denote its left child and its right child which are the subintervals resulting from its bisection. Otherwise, *isub* is a called a childless subinterval and *ilchild*(*isub*) and *irchild*(*isub*) are set to 0.
- *ilnbr*(*isub*) and *irnbr*(*isub*) will denote the left and right neighbors for subinterval *isub*, which are the smallest adjacent subintervals at the same level of refinement or a coarser one.
- Φ_i^L and Φ_i^R will be $\bar{p}$-vectors denoting respectively the left and right far-field expansions for subinterval i.
- Ψ_i^L and Ψ_i^R will be $\bar{p}$-vectors denoting respectively the left and right local expansions for subinterval i.

2.3.2 Description of the Algorithm

This section contains a detailed description and a complexity analysis of an adaptive version of Algorithm 2.2.

Algorithm 2.3
Initialization Step
Comment [Geometrical preprocessing]
Comment [Input problem size, N, and a real number $\varepsilon > 0$.]
Set size of Chebyshev expansions $p = \lceil -\log_5(\varepsilon) \rceil$, choose $\bar{p}$, choose s and set maximum level of refinement $m = \lceil \log_2(N/s) \rceil$.

$I_1 = \{[-1,0],[0,1]\}$
do $l = 1,\ldots,m$ **while** I_l is non-empty
  do $isub \in I_l$
    if $isub$ contains more than s points **then**
      Add $ilchild(isub)$ and $irchild(isub)$ to I_{l+1}.
    else
      $ilchild(isub) = 0$
      $irchild(isub) = 0$
    endif
  enddo
enddo
$nlevs = l$
do $l = 1,\ldots,nlevs$
  do $isub \in I_l$
    if $isub$ is not childless **then**
      $ilnbr(irchild(isub)) = ilchild(isub)$
      $irnbr(ilchild(isub)) = irchild(isub)$
      if $ilnbr(isub)$ is not childless **then**
        $ilnbr(ilchild(isub)) = irchild(ilnbr(isub))$
      else
        $ilnbr(ilchild(isub)) = ilnbr(isub)$
      endif
      if $irnbr(isub)$ is not childless **then**
        $irnbr(irchild(isub)) = ilchild(irnbr(isub))$
      else
        $irnbr(irchild(isub)) = irnbr(isub)$
      endif
    endif
  enddo
enddo

Step 1
Comment [Upward Pass]

do $l = nlevs,\ldots,1$
  do $isub \in I_l$
    if $isub$ is a childless subinterval **then**
      Form a $\bar{p}$-term far-field expansion Φ^L_{isub}.
      Form a $\bar{p}$-term far-field expansion Φ^R_{isub}.
    else
      $\Phi^L_{isub} = M^L_L \cdot \Phi^L_{ilchild(isub)} + M^L_R \cdot \Phi^L_{irchild(isub)}$
      $\Phi^R_{isub} = M^R_L \cdot \Phi^R_{ilchild(isub)} + M^R_R \cdot \Phi^R_{irchild(isub)}$
    endif

enddo
enddo

Step 2
Comment [Downward Pass]

do $l = 1, \ldots, nlevs$
do $isub \in I_l$
if $isub$ is a childless subinterval **then**
Evaluate and add $\bar{p}$-term local expansions $\Psi^L_{isub} + \Psi^R_{isub}$.
else
$\Psi^L_{ilchild(isub)} = S^R_L \cdot \Psi^L_{isub}$
$\Psi^L_{irchild(isub)} = S^R_R \cdot \Psi^L_{isub}$
$\Psi^R_{ilchild(isub)} = S^R_L \cdot \Psi^R_{isub}$
$\Psi^R_{irchild(isub)} = S^R_R \cdot \Psi^R_{isub}$
if $ilnbr(isub)$ is a childless subinterval **then**
Add contribution of $\Phi^L_{irchild(isub)}$ to each point in $ilnbr(isub)$
Add contribution of each point in $ilnbr(isub)$ to $\Psi^L_{irchild(isub)}$
else
$\Psi^L_{ilchild(isub)} = \Psi^L_{ilchild(isub)} + T^L_1 \cdot \Phi^R_{ilchild(ilnbr(isub))}$
$\Psi^L_{irchild(isub)} = \Psi^L_{irchild(isub)} + T^L_1 \cdot \Phi^R_{irchild(ilnbr(isub))} + T^L_2 \cdot \Phi^R_{ilchild(ilnbr(isub))}$
endif
if $irnbr(isub)$ is a childless subinterval **then**
Add contribution of $\Phi^R_{ilchild(isub)}$ to each point in $irnbr(isub)$
Add contribution of each point in $irnbr(isub)$ to $\Psi^R_{ilchild(isub)}$
else
$\Psi^R_{ilchild(isub)} = \Psi^R_{ilchild(isub)} + T^R_1 \cdot \Phi^R_{ilchild(irnbr(isub))} + T^R_2 \cdot \Phi^R_{irchild(irnbr(isub))}$
$\Psi^R_{irchild(isub)} = \Psi^R_{irchild(isub)} + T^R_1 \cdot \Phi^R_{irchild(irnbr(isub))}$
endif
endif
enddo
enddo

Step 3
Comment [Direct Interactions]

do $isub = 1, \ldots, nsub$
if $isub$ is childless **then**
For each point in subinterval $isub$, compute interactions with all other points in this subinterval and in adjacent childless subintervals, and add to far-field values.
endif
enddo

Following is a complexity analysis of Algorithm 2.3. Before presenting a step-by-step breakdown of the operation counts, we need two lemmas. These lemmas provide upper bounds on the numbers of subintervals which can be created by the process of selective subdivision.

Lemma 2.11 *For any subdivision of the computational domain produced by Algorithm 2.3, the number of childless intervals is bounded by*

$$2 \cdot \log_2\left(\frac{1}{\varepsilon}\right) \cdot \frac{N}{s}. \tag{2.91}$$

Proof. Each parent interval at level l contains more than s points (otherwise it would not be further subdivided). Therefore, the total number of parent intervals at level l is bounded above by N/s. Each parent interval has two children by definition, so the number of childless boxes at any level l is bounded above by $2N/s$. The bound (2.91) follows from the fact that the number of levels of subdivision is bounded above by $\log_2(1/\varepsilon)$ (see Remark 2.15). □

Lemma 2.12 *For any subdivision of the computational domain produced by Algorithm 2.3, the total number of intervals is bounded by*

$$3 \cdot \log_2\left(\frac{1}{\epsilon}\right) \cdot \frac{N}{s}. \tag{2.92}$$

Proof. The number of parent intervals at level l is bounded above by N/s. Each of these parent intervals has two children, so the number of childless boxes at any level l is bounded above by $2N/s$. Thus, the total number of intervals at all levels (childless and parent) is bounded by $\log_2(1/\varepsilon) \cdot (N/s + 2N/s)$. □

Following is a step-by-step breakdown of the operation counts of Algorithm 2.3.

Step	Complexity	Description
1	$O(2\bar{p}N)+$	Each point contributes to the left and right $\bar{p}$-term far-field expansions of the childless subinterval in which it lies.
	$O(4\bar{p}^2 mN/s)$	For each parent subinterval (there are at most mN/s) we perform 4 $\bar{p} \times \bar{p}$ matrix-vector products.
2	$O(2\bar{p}N)+$	We evaluate and add the left and right $\bar{p}$-term local expansions for each of the N points.

	$O(4\bar{p}^2mN/s)+$	For each parent subinterval (there are at most mN/s) we perform 4 $\bar{p} \times \bar{p}$ matrix-vector products.
	$O(6\bar{p}^2mN/s)+$	For each parent subinterval (there are at most mN/s) we perform at most 6 $\bar{p} \times \bar{p}$ matrix-vector products.
	$O(4\bar{p}mN)$	For each parent subinterval (there are at most mN/s) we perform at most 4 $\bar{p} \times s$ matrix-vector products.
3	$O(3Ns)$	Each of the y_j interacts directly with those x_k which lie in its own subinterval and the two nearest neighbors. There are at most $3s$ of these points x_k.
Total	$O(14\bar{p}^2mN/s + 4\bar{p}mN + 4\bar{p}N + 3Ns)$	

Remark 2.16 We choose $s \approx 2\bar{p}$, as in Algorithm 2.2 (see Remark 2.12). The above algorithm then requires order

$$N \cdot (11\bar{p}m + 10\bar{p}) \tag{2.93}$$

arithmetic operations.

2.4 FMM for Other Kernels

The algorithms described in Sections 2.2 and 2.3 are all designed for evaluating sums of the form

$$f(x) = \sum_{k=1}^{N} \frac{\alpha_k}{x - x_k}. \tag{2.94}$$

However, these algorithms are only mildly dependent on the choice of kernel, i.e. they can be modified to evaluate expressions of a more general form, given by the formula

$$f(x) = \sum_{k=1}^{N} \alpha_k \cdot \phi(x - x_k), \tag{2.95}$$

where ϕ is singular at 0 but smooth everywhere else. Examples of two closely related functions with this property are $\phi(x) = \log(x)$ and $\phi(x) = 1/x^2$, which are readily obtained by integrating and differentiating the expression (2.94). Another example of interest is $\phi(x) = 1/\tan(x)$ on the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$. This case is discussed in more detail in Chapter 3, where it arises in the formulation of the trigonometric interpolation problem.

Remark 2.17 While the algorithmic procedures for different kernels ϕ are virtually identical, different sets of formulae are needed for the creation and manipulation of the Chebyshev far-field and local expansions which are required by the algorithms. These formulae can be obtained by constructing analogs of Theorems 2.5, 2.7, 2.8, 2.9 and 2.10 for the particular function ϕ.

2.5 A Fast Algorithm for Polynomial Interpolation

In this section we return to the original problem of this chapter: given a set of points $\{x_1, \ldots, x_N\}$ and function values $\{f_1, \ldots, f_N\}$, evaluate the unique interpolating polynomial at the points $\{y_1, \ldots, y_N\}$. Recall from (2.2) that the Lagrange interpolating polynomial defined by the formula

$$P_N(x) = \sum_{j=1}^{N} f_j \cdot \prod_{\substack{k=1 \\ k \neq j}}^{N} \frac{x - x_k}{x_j - x_k} \tag{2.96}$$

can be rewritten in the form

$$P_N(x) = \prod_{k=1}^{N} (x - x_k) \cdot \sum_{j=1}^{N} \frac{f_j}{x - x_j} \cdot \prod_{\substack{k=1 \\ k \neq j}}^{N} \frac{1}{x_j - x_k}. \tag{2.97}$$

Furthermore,

$$P_N(y_l) = r_l \cdot \sum_{j=1}^{N} \frac{f_j \cdot s_j}{y_l - x_j} \tag{2.98}$$

for $l = 1, \ldots, n$, where r_l and s_j are defined by the formulae

$$r_l = \prod_{k=1}^{N} (y_l - x_k) = e^{\sum_{k=1}^{N} \ln(y_l - x_k)}, \tag{2.99}$$

and,

$$s_j = \prod_{\substack{k=1 \\ k \neq j}}^{N} \frac{1}{x_j - x_k} = e^{-\sum_{k=1, k \neq j}^{N} \ln(x_j - x_k)}. \tag{2.100}$$

Observation 2.18 *Sums of the form $\sum \ln(x - x_k)$ can also be evaluated using the algorithms of this chapter (see Section 2.4). The numbers $\{r_l\}$ and $\{s_j\}$ can therefore be computed in $O(N \log(\frac{1}{\varepsilon}))$ operations according to (2.99) and (2.100).*

Following is a description of an algorithm for the efficient evaluation of expressions of the form (2.98).

Algorithm 2.4

Step	Complexity	Description
Init	$O(N\log(\frac{1}{\varepsilon}))$	Compute the numbers $\{r_l\}$ and $\{s_j\}$.
1	$O(N)$	**do** $j = 1, n$
		$g_j = f_j \cdot s_j$
		end do
2	$O(N\log(\frac{1}{\varepsilon}))$	Compute $\tilde{p}_l = \sum_{j=1}^{n} g_j/(y_l - x_j)$ using Algorithm 2.3 (or Algorithm 2.2).
3	$O(N)$	**do** $l = 1, n$
		$\tilde{p}_l = \tilde{p}_l \cdot r_l$
		end do
Total	$O(N\log(\frac{1}{\varepsilon}))$	

Remark 2.19 It is well known that the polynomial interpolant of a function is spectrally accurate when the function is tabulated at Chebyshev or Legendre nodes (which are clustered near the interval ends), whereas the interpolation errors can be arbitrarily large when the function is tabulated at general distributions of points (see, for example, [9], [23]). It is expected that many practical applications of Algorithm 2.4 will assume nonuniformly spaced nodes which are clustered near the extremities of the interval.

2.6 Applications in Numerical Integration and Differentiation

The fast polynomial interpolation algorithm of this chapter can be applied to a variety of problems. One such example is discussed in this section. Here we will consider the folowing problem: given a set of points $\{x_1, \ldots, x_N\}$ and function values $\{f_1, \ldots, f_N\}$, evaluate the integrals and derivatives of the interpolating polynomial at the points $\{x_k\}$. In other words, we wish to compute

$$\int_{-1}^{x_k} P_N(x)dx \quad \text{and} \quad P_N'(x_k) \tag{2.101}$$

for $k = 1, \ldots, N$, where P_N is the interpolating polynomial for the function values $\{f_k\}$ at the points $\{x_k\}$, defined by the Lagrange formula

$$P_N(x) = \sum_{j=1}^{N} f_j \cdot \prod_{\substack{k=1 \\ k\neq j}}^{N} \frac{x - x_k}{x_j - x_k}. \tag{2.102}$$

We will make use of the following lemma, which may be found in the appendix to [13]. This lemma describes formulae for the integration and differentiation of Chebyshev expansions.

Lemma 2.13 *Let P_N be a polynomial given by a Chebyshev series*

$$P_N(x) = \sum_{k=0}^{N-1} a_k \cdot T_k(x). \tag{2.103}$$

Then, the integral of P_N has a series expansion of the form

$$\int_{-1}^{x} P_N(t)dt = \sum_{k=0}^{N} b_k \cdot T_k(x), \tag{2.104}$$

where

$$\begin{aligned} b_k &= \frac{1}{2k} \cdot (a_{k-1} - a_{k+1}) \quad \textit{for } 2 \le k \le N, \\ b_1 &= \frac{1}{2} \cdot (2a_0 - a_2), \\ b_0 &= -2 \cdot \sum_{j=1}^{N} (-1)^j \cdot b_j, \end{aligned} \tag{2.105}$$

and the derivative of P_N has a series expansion of the form

$$\frac{d}{dx} P_N(x) = \sum_{k=0}^{N-2} d_k \cdot T_k(x), \tag{2.106}$$

where

$$\begin{aligned} d_k &= \sum_{\substack{j=k+1 \\ j+k \ odd}}^{N} j \cdot a_j, \quad \textit{for } 1 \le k \le N-2, \\ d_0 &= \frac{1}{2} \cdot \sum_{\substack{j=1 \\ j \ odd}}^{N} j \cdot a_j. \end{aligned} \tag{2.107}$$

Remark 2.20 It can be shown that the process of numerical differentiation via Chebyshev series has a condition number proportional to N^2, whereas the process of numerical integration via Chebyshev series has a condition number bounded by 2. Thus, numerical differentiation of this type is not usually favored when large scale calculations are being performed. On the other hand, numerical integration is virtually insensitive to problem size, and is a powerful tool in the solution of certain classes of differential equations, for example (for a more detailed discussion, see [15]).

The two algorithms described below perform the spectral integration and differentiation of the Lagrange polynomial interpolant of a function which is tabulated at nodes other than Chebyshev. In these descriptions we will assume that $x_k \in [-1, 1]$ for $k = 1, \ldots, N$, and that $t_1, \ldots, t_N$ are Chebyshev nodes of order N on the interval $[-1, 1]$.

Algorithm 2.5

Step	Complexity	Description
1	$O(N \log(\frac{1}{\varepsilon}))$	Interpolate from $\{x_k\}$ to $\{t_k\}$ using Algorithm 2.4.
2	$O(N \log N)$	Compute Chebyshev coefficients using fast cosine transform.
3	$O(N)$	Integrate Chebyshev series using (2.105).
4	$O(N \log N)$	Evaluate new series at Chebyshev nodes using fast cosine transform.
5	$O(N \log(\frac{1}{\varepsilon}))$	Interpolate from $\{t_k\}$ to $\{x_k\}$ using Algorithm 2.4.
Total	$O(N \cdot \log N + N \cdot \log(\frac{1}{\varepsilon}))$	

Algorithm 2.6

Step	Complexity	Description
1	$O(N \log(\frac{1}{\varepsilon}))$	Interpolate from $\{x_k\}$ to $\{t_k\}$ using Algorithm 2.4.
2	$O(N \log N)$	Compute Chebyshev coefficients using fast cosine transform.
3	$O(N)$	Differentiate Chebyshev series using (2.107).
4	$O(N \log N)$	Evaluate new series at Chebyshev nodes using fast cosine transform.
5	$O(N \log(\frac{1}{\varepsilon}))$	Interpolate from $\{t_k\}$ to $\{x_k\}$ using Algorithm 2.4.
Total	$O(N \cdot \log N + N \cdot \log(\frac{1}{\varepsilon}))$	

Chapter 3

Trigonometric Interpolation and FFTs

In this chapter we will consider the transformation $F : \mathbf{C}^N \rightarrow \mathbf{C}^N$ defined by the formulae

$$F(\alpha)_j = f_j = \sum_{k=-N/2}^{N/2-1} \alpha_k \cdot e^{ikx_j}, \tag{3.1}$$

for $j = 1, \ldots, N$, where $x = \{x_1, \ldots, x_N\}$ is a sequence of real numbers in $[-\pi, \pi]$ and $\alpha = \{\alpha_{-N/2}, \ldots, \alpha_{N/2-1}\}$ and $f = \{f_1, \ldots, f_N\}$ are sequences of complex numbers. We are interested in the efficient application and inversion of the transformation F and its transpose. More precisely, we will consider the following four problems:

- **Problem 3.1:** Given α, find $f = F(\alpha)$.
- **Problem 3.2:** Given α, find $f = F^T(\alpha)$.
- **Problem 3.3:** Given f, find $\alpha = F^{-1}(f)$.
- **Problem 3.4:** Given f, find $\alpha = (F^T)^{-1}(f)$.

In this chapter we will describe a group of four efficient algorithms for Problems 3.1–3.4. These algorithms utilize the fact that a Fourier series is a trigonometric polynomial; when dealing with the values of this polynomial at equispaced nodes on the unit circle, the FFT can be applied. However, we are interested in the values at nonuniformly spaced nodes, which are values of the polynomial which interpolates the equispaced values. The algorithms we will describe rely for their efficiency on a combination of the FFT with a fast algorithm for evaluating trigonometric polynomial interpolants which uses a version of the Fast Multipole Method (FMM) specifically

designed for the geometry of the circle. This interpolation algorithm is closely related to Algorithm 2.4 described in Chapter 2 for the interpolation of polynomials tabulated on the line.

Remark 3.1 Throughout this chapter we will be using the well known Lagrange representation of polynomial interpolants. For a function $f : \mathbf{C} \rightarrow \mathbf{C}$ tabulated at nodes $z_1, \ldots, z_N$, this is defined by the formula

$$P_N(z) = \sum_{j=1}^{N} f(z_j) \cdot \prod_{\substack{k=1 \\ k \neq j}}^{N} \frac{z - z_k}{z_j - z_k}. \tag{3.2}$$

This chapter is divided into two sections. Section 3.1 contains a number of results from analysis and approximation theory, and in Section 3.2 we describe both formally and informally how these results are used, together with the FMM, in the construction of the fast algorithms of this chapter.

Remark 3.2 An alternative approach to the problems of this chapter is presented in Chapter 4, where an interpolation scheme based on the Fourier analysis of the Gaussian bell is used in place of the FMM-based interpolation scheme of this chapter. The two approaches are compared in Chapter 6.

3.1 Mathematical and Numerical Preliminaries

This section is divided into two parts. In Subsection 3.1.1 we present several identities which are employed in the development of the fast algorithms of this chapter. Subsection 3.1.2 contains a collection of error bounds which allow us to perform calculations to any prescribed accuracy.

3.1.1 Analytical Tools

The main results of this subsection are Theorems 3.3 and 3.4 which describe linear transformations connecting the values of a Fourier series at two distinct sets of points. Lemmas 3.1 and 3.2 provide intermediate results which are used in the proofs of these theorems.

Lemma 3.1 *Let $\{x_1, \ldots, x_N\}$ and $\{y_1, \ldots, y_N\}$ be sequences of real numbers on the interval $[-\pi, \pi]$, and let $\{w_1, \ldots, w_N\}$ and $\{z_1, \ldots, z_N\}$ be sequences of complex numbers defined by the formulae*

$$w_j = e^{ix_j} \tag{3.3}$$

$$z_j = e^{iy_j} \tag{3.4}$$

for $j = 1, \ldots, N$. *Then,*

$$\prod_{\substack{k=1 \\ k \neq j}}^{N} (w_l - z_k) = w_l^{(N-1)/2} \cdot \prod_{\substack{k=1 \\ k \neq j}}^{N} z_k^{1/2} \cdot 2i \cdot \sin((x_l - y_k)/2) \tag{3.5}$$

for $l = 1, \ldots, N$, *and*

$$\prod_{\substack{k=1 \\ k \neq j}}^{N} (z_j - z_k) = z_j^{(N-1)/2} \cdot \prod_{\substack{k=1 \\ k \neq j}}^{N} z_k^{1/2} \cdot 2i \cdot \sin((y_j - y_k)/2) \tag{3.6}$$

for $j = 1, \ldots, N$.

Proof. A sequence of simple algebraic manipulations and trigonometric identities yields

$$\begin{aligned}
\prod_{\substack{k=1 \\ k \neq j}}^{N} (w_l - z_k) &= \prod_{\substack{k=1 \\ k \neq j}}^{N} (e^{ix_l} - e^{iy_k}) \\
&= \prod_{\substack{k=1 \\ k \neq j}}^{N} e^{i(x_l + y_k)/2} \cdot (e^{i(x_l - y_k)/2} - e^{-i(x_l - y_k)/2}) \\
&= e^{i(N-1)x_l/2} \cdot \prod_{\substack{k=1 \\ k \neq j}}^{N} e^{iy_k/2} \cdot 2i \cdot \sin((x_l - y_k)/2) \qquad (3.7) \\
&= w_l^{(N-1)/2} \cdot \prod_{\substack{k=1 \\ k \neq j}}^{N} z_k^{1/2} \cdot 2i \cdot \sin((x_l - y_k)/2).
\end{aligned}$$

Substituting z_j for w_l and y_j for x_l in (3.7), we also obtain

$$\prod_{\substack{k=1 \\ k \neq j}}^{N} (z_j - z_k) = z_j^{(N-1)/2} \cdot \prod_{\substack{k=1 \\ k \neq j}}^{N} z_k^{1/2} \cdot 2i \cdot \sin((y_j - y_k)/2). \tag{3.8}$$

□

The following lemma describes an alternative representation of the well known Lagrange interpolation formula for polynomials in the case when the interpolation points lie on the circle.

Lemma 3.2 *Let $\{x_1,\ldots,x_N\}$ and $\{y_1,\ldots,y_N\}$ be sequences of real numbers in the interval $[-\pi,\pi]$, and let $\{f_1,\ldots,f_N\}$ be a sequence of complex numbers. Further, let $\{w_1,\ldots,w_N\}$ and $\{z_1,\ldots,z_N\}$ be sequences of complex numbers defined by the formulae*

$$w_j = e^{ix_j} \tag{3.9}$$

$$z_j = e^{iy_j} \tag{3.10}$$

for $j=1,\ldots,N$. Then,

$$\sum_{j=1}^{N} f_j \cdot \prod_{\substack{k=1\\k\neq j}}^{N} \frac{w_l - z_k}{z_j - z_k} = w_l^{N/2} \cdot c_l \cdot \sum_{j=1}^{N} f_j \cdot z_j^{-N/2} \cdot d_j \cdot \left(\frac{1}{\tan((x_l - y_j)/2)} - i\right) \tag{3.11}$$

where $\{c_l\}$ and $\{d_j\}$ are defined by the formulae

$$c_l = \prod_{k=1}^{N} \sin((x_l - y_k)/2), \tag{3.12}$$

$$d_j = \prod_{\substack{k=1\\k\neq j}}^{N} \frac{1}{\sin((y_j - y_k)/2)} \tag{3.13}$$

for $j,l=1,\ldots,N$.

Proof. Dividing (3.5) by (3.6) we obtain

$$\begin{aligned}\prod_{\substack{k=1\\k\neq j}}^{N} \frac{w_l - z_k}{z_j - z_k} &= \frac{w_l^{(N-1)/2}}{z_j^{(N-1)/2}} \cdot \prod_{\substack{k=1\\k\neq j}}^{N} \frac{z_k^{1/2} \cdot 2i \cdot \sin((x_l - y_k)/2)}{z_k^{1/2} \cdot 2i \cdot \sin((y_j - y_k)/2)} \\ &= \frac{e^{-i(x_l - y_j)/2}}{\sin((x_l - y_j)/2)} \cdot \frac{w_l^{N/2} \prod_{k=1}^{N} \sin((x_l - y_k)/2)}{z_j^{N/2} \prod_{k\neq j} \sin((y_j - y_k)/2)},\end{aligned} \tag{3.14}$$

and the combination of (3.14) with the fact that

$$\begin{aligned}\frac{e^{-i(x_l - y_j)/2}}{\sin((x_l - y_j)/2)} &= \frac{\cos((x_l - y_j)/2) + i\sin((x_l - y_j)/2)}{\sin((x_l - y_j)/2)} \\ &= \frac{1}{\tan((x_l - y_j)/2)} - i,\end{aligned} \tag{3.15}$$

gives us

$$\sum_{j=1}^{N} f_j \cdot \prod_{\substack{k=1\\k\neq j}}^{N} \frac{w_l - z_k}{z_j - z_k} = w_l^{N/2} \cdot c_l \cdot \sum_{j=1}^{N} f_j \cdot z_j^{-N/2} \cdot d_j \cdot \left(\frac{1}{\tan((x_l - y_j)/2)} - i\right) \tag{3.16}$$

where $\{c_l\}$ and $\{d_j\}$ are defined by (3.12) and (3.13). □

The following theorem provides a formula for determining the values of a Fourier series at a set of points in terms of the values of this series at another set of points.

Theorem 3.3 *Let $\{x_1, \ldots, x_N\}$ and $\{y_1, \ldots, y_N\}$ be sequences of real numbers in the interval $[-\pi, \pi]$, and let $\{\alpha_{-N/2}, \ldots, \alpha_{N/2-1}\}$ be a sequence of complex numbers. Further, let $\{w_1, \ldots, w_N\}$, $\{z_1, \ldots, z_N\}$, $\{f_1, \ldots, f_N\}$ and $\{g_1, \ldots, g_N\}$ be sequences of complex numbers defined by the formulae*

$$w_j = e^{ix_j} \tag{3.17}$$

$$z_j = e^{iy_j} \tag{3.18}$$

$$f_j = \sum_{k=-N/2}^{N/2-1} \alpha_k \cdot e^{iky_j} \tag{3.19}$$

$$g_j = \sum_{k=-N/2}^{N/2-1} \alpha_k \cdot e^{ikx_j} \tag{3.20}$$

for $j = 1, \ldots, N$. Then,

$$g_l = c_l \cdot \sum_{j=1}^{N} f_j \cdot d_j \cdot \left(\frac{1}{\tan((x_l - y_j)/2)} - i \right), \tag{3.21}$$

where $\{c_l\}$ are defined by (3.12) and $\{d_j\}$ are defined by (3.13).

Proof. Let the polynomial P_α be defined by the formula

$$P_\alpha(z) = \sum_{k=0}^{N-1} \alpha_{k-N/2} \cdot z^k. \tag{3.22}$$

The Lagrange interpolation formula relates the values of P_α at the points $\{w_l\}$ to the values at the points $\{z_k\}$ via the expressions

$$P_\alpha(w_l) = \sum_{j=1}^{N} P_\alpha(z_j) \cdot \prod_{\substack{k=1 \\ k \neq j}}^{N} \frac{w_l - z_k}{z_j - z_k} \tag{3.23}$$

for $l = 1, \ldots, N$, and applying Lemma 3.2 to (3.23) we obtain

$$P_\alpha(w_l) = w_l^{N/2} \cdot c_l \cdot \sum_{j=1}^{N} P_\alpha(z_j) \cdot z_j^{-N/2} \cdot d_j \cdot \left(\frac{1}{\tan((x_l - y_j)/2)} - i \right). \tag{3.24}$$

From the combination of (3.22) and (3.17)–(3.20) we see that

$$P_\alpha(z_j) = z_j^{N/2} \cdot \sum_{k=-N/2}^{N/2-1} \alpha_k \cdot z_j^k = z_j^{N/2} \cdot f_j \tag{3.25}$$

and

$$P_\alpha(w_l) = w_l^{N/2} \cdot \sum_{k=-N/2}^{N/2-1} \alpha_k \cdot w_l^k = w_l^{N/2} \cdot g_l, \tag{3.26}$$

and finally substituting (3.25) and (3.26) into (3.24) we obtain

$$g_l = c_l \cdot \sum_{j=1}^{N} f_j \cdot d_j \cdot \left(\frac{1}{\tan((x_l - y_j)/2)} - i \right) \tag{3.27}$$

for $l = 1, \ldots, N$. □

In the case when the points $\{y_j\}$ are equispaced in $[-\pi, \pi]$, the interpolation formula of Theorem 3.3 has a simpler form, which is described in the following theorem. The result of this theorem can be found in a slightly different form in [12].

Theorem 3.4 *Let $\{x_1, \ldots, x_N\}$ be a sequence of real numbers on the interval $[-\pi, \pi]$ and let $\{\alpha_{-N/2}, \ldots, \alpha_{N/2-1}\}$ be a sequence of complex numbers. Further, let $\{y_1, \ldots, y_N\}$ be a sequence of real numbers defined by the formulae*

$$y_j = (j - 1 - N/2)\pi/N \tag{3.28}$$

for $j = 1, \ldots, N$, and let $\{w_1, \ldots, w_N\}$, $\{z_1, \ldots, z_N\}$, $\{f_1, \ldots, f_N\}$ and $\{g_1, \ldots, g_N\}$ be sequences of complex numbers defined by the formulae

$$w_j = e^{ix_j} \tag{3.29}$$

$$z_j = e^{iy_j} \tag{3.30}$$

$$f_j = \sum_{k=-N/2}^{N/2-1} \alpha_k \cdot e^{iky_j} \tag{3.31}$$

$$g_j = \sum_{k=-N/2}^{N/2-1} \alpha_k \cdot e^{ikx_j} \tag{3.32}$$

for $j = 1, \ldots, N$. Then,

$$g_l = \sin\left(\frac{Nx_l}{2}\right) \cdot \sum_{j=1}^{N} f_j \cdot \frac{(-1)^j}{N} \cdot \left(\frac{1}{\tan((x_l - y_j)/2)} - i \right). \tag{3.33}$$

Proof. From the combination of equations (3.28) and (3.31), we see that the sequence $\{\alpha_{-N/2}, \ldots, \alpha_{N/2-1}\}$ is the discrete Fourier transform of the sequence $\{f_1, \ldots, f_N\}$. In other words,

$$\alpha_k = \frac{1}{N} \cdot \sum_{j=1}^{N} f_j \cdot e^{-iky_j} \tag{3.34}$$

for $k = -N/2, \ldots, N/2 - 1$. Let us now define the function $f : [-\pi, \pi] \to \mathbf{C}$ by the formula

$$f(x) = \sum_{k=-N/2}^{N/2-1} \alpha_k \cdot e^{ikx}. \tag{3.35}$$

Substituting (3.34) into (3.35) and changing the order of summation, we obtain

$$\begin{aligned} f(x) &= \sum_{k=-N/2}^{N/2-1} \frac{1}{N} \cdot \sum_{j=1}^{N} f_j \cdot e^{-iky_j} \cdot e^{ikx} \\ &= \sum_{j=1}^{N} f_j \cdot \frac{1}{N} \cdot \sum_{k=-N/2}^{N/2-1} e^{ik(x-y_j)}. \end{aligned} \tag{3.36}$$

Observing that the second sum in the expression (3.36) is a geometric series we have

$$\begin{aligned} \sum_{k=-N/2}^{N/2-1} e^{ik(x-y_j)} &= \frac{e^{-iN(x-y_j)/2} - e^{iN(x-y_j)/2}}{1 - e^{i(x-y_j)}} \\ &= \frac{\sin(N(x-y_j)/2)}{e^{i(x-y_j)/2} \cdot \sin((x-y_j)/2)} \\ &= (\sin(N(x-y_j)/2)) \cdot (\cot((x-y_j)/2) - i) \end{aligned} \tag{3.37}$$

for any $x \in [-\pi, \pi]$. The definition of $\{y_j\}$ now yields

$$\begin{aligned} \sin(N(x-y_j)/2) &= \sin(Nx/2)\cos(Ny_j/2) - \cos(Nx/2)\sin(Ny_j/2) \\ &= \sin(Nx/2) \cdot (-1)^j, \end{aligned} \tag{3.38}$$

and finally, using the fact that $g_l = f(x_l)$ and combining (3.36), (3.37) and (3.38) we obtain

$$g_l = \sin\left(\frac{Nx_l}{2}\right) \cdot \sum_{j=1}^{N} f_j \cdot \frac{(-1)^j}{N} \cdot \left(\frac{1}{\tan((x_l - y_j)/2)} - i\right). \tag{3.39}$$

□

3.1.2 Relevant Facts from Approximation Theory

The algorithms of this chapter are based on several results from the Chebyshev approximation theory of the function $1/\tan(x)$. These results are contained in the lemmas and theorems of this subsection, numbered 3.8–3.14. Analogs of these results for the function $1/x$ can be found in Section 2.1.

The main results of this section fall into two categories. Theorems 3.11 and 3.14 describe how the function $1/\tan(x)$ can be approximated on different regions of the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$ using Chebyshev expansions. Theorems 3.15, 3.16 and 3.17 provide three ways of manipulating these expansions which are needed by the fast algorithms of this chapter.

We begin with three classical definitions which can be found, for example, in [14], [23].

Definition 3.1 *The n-th degree Chebyshev polynomial $T_n(x)$ is defined by the following equivalent formulae:*

$$T_n(x) = \cos(n \arccos x) \tag{3.40}$$

$$T_n(x) = \frac{1}{2} \cdot \left((x + \sqrt{x^2 - 1})^n + (x - \sqrt{x^2 - 1})^n \right). \tag{3.41}$$

Definition 3.2 *The roots $t_1, \ldots, t_n$ of the n-th degree Chebyshev polynomial T_n lie in the interval $[-1, 1]$ and are defined by the formulae*

$$t_k = -\cos\left(\frac{2k-1}{n} \cdot \frac{\pi}{2} \right) \tag{3.42}$$

for $k = 1, \ldots, n$. They are referred to as Chebyshev nodes of order n.

Definition 3.3 *We will define the polynomials $u_1, \ldots, u_n$ of order $n-1$ by the formulae*

$$u_j(t) = \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{t - t_k}{t_j - t_k} \tag{3.43}$$

for $j = 1, \ldots, n$, where t_k are defined by (3.42).

The order $n-1$ Chebyshev approximation for a function $f : [-1, 1] \to \mathbf{C}$ is defined as the unique polynomial of order $n-1$ which agrees with f at the nodes $t_1, \ldots, t_n$. There exist several standard representations for this polynomial, and the one we will use in this chapter is given by the expression

$$\sum_{j=1}^{n} f(t_j) \cdot u_j(t). \tag{3.44}$$

For the purposes of this chapter, Chebyshev expansions for any function will be characterized by values of this function tabulated at Chebyshev nodes.

Lemmas 3.5–3.7 provide estimates involving Chebyshev expansions which are used in the remainder of this section. The proof of Lemma 3.5 is obvious from (3.40).

Lemma 3.5 *Let $T_n(x)$ be the Chebyshev polynomial of degree n. Then,*

$$|T_n(x)| \leq 1 \tag{3.45}$$

for any $x \in [-1, 1]$.

Lemma 3.6 *Let $T_n(x)$ be the Chebyshev polynomial of degree n. Then,*

$$|T_n(x)| > \frac{1}{2} \cdot \left|\frac{5x}{3}\right|^n \tag{3.46}$$

for any x such that $|x| \geq 3$.

Proof. From Definition 3.1, we have

$$\begin{aligned} |T_n(x)| &= \frac{1}{2} \cdot \left|(x + \sqrt{x^2 - 1})^n + (x - \sqrt{x^2 - 1})^n\right| \\ &> \frac{1}{2} \cdot |x + \sqrt{x^2 - (x/3)^2}|^n = \frac{1}{2} \cdot |x \cdot (1 + \sqrt{8/9})|^n \\ &> \frac{1}{2} \cdot \left|\frac{5x}{3}\right|^n \end{aligned} \tag{3.47}$$

for any x such that $|x| \geq 3$. □

Lemma 3.7 *Let $u_j(x)$ be defined by (3.43). Then, for any $x \in [-1, 1]$,*

$$|u_j(x)| \leq 1. \tag{3.48}$$

Proof. It is obvious from (3.43) that $u_j(t_j) = 1$, and that $u_j(t_k) = 0$ when $k \neq j$. In addition, the expression

$$\frac{1}{n} \sum_{k=1}^{n} T_k(t_j) \cdot T_k(x) \tag{3.49}$$

is also equal to 1 at t_j and equal to 0 at all other t_k. Since both u_j and (3.49) are polynomials of order $n - 1$, we have

$$u_j(x) = \frac{1}{n} \sum_{k=1}^{n} T_k(t_j) \cdot T_k(x) \tag{3.50}$$

for $j = 1, \ldots, n$. Furthermore, due to the combination of (3.50) and the triangle inequality, we obtain

$$|u_j(x)| = \left| \frac{1}{n} \sum_{k=1}^{n} T_k(t_j) \cdot T_k(x) \right| = \frac{1}{n} \sum_{k=1}^{n} |T_k(t_j)| \cdot |T_k(x)| \leq 1 \tag{3.51}$$

for any $x \in [-1, 1]$. □

The next lemma is obvious.

Lemma 3.8 *For any $a \in [0, \frac{\pi}{8}]$,*

$$\tan 3a \geq 3 \cdot \tan a. \tag{3.52}$$

The following two lemmas provide preliminary results which are used in the proof of Theorem 3.11.

Lemma 3.9 *Suppose that $n \geq 2$, and that $b > 0$ and x_0 are real numbers with $|x_0| \geq 3b$. Then, for any x,*

$$1 + xx_0 - (x - x_0) \cdot \sum_{j=1}^{n} \frac{1 + bt_j x_0}{bt_j - x_0} \cdot u_j\left(\frac{x}{b}\right) = (1 + x_0^2) \cdot \frac{T_n(x/b)}{T_n(x_0/b)}. \tag{3.53}$$

Proof. Let $Q(x)$ by the polynomial of degree n defined by the formula

$$Q(x) = 1 + xx_0 - (x - x_0) \cdot \sum_{j=1}^{n} \frac{1 + bt_j x_0}{bt_j - x_0} \cdot u_j\left(\frac{x}{b}\right). \tag{3.54}$$

It follows from the combination of (3.43) and (3.54) that

$$\begin{aligned} Q(bt_k) &= 1 + bt_k x_0 - (bt_k - x_0) \cdot \sum_{j=1}^{n} \frac{1 + bt_j x_0}{bt_j - x_0} \cdot u_j(t_k) \\ &= 1 + bt_k x_0 - (bt_k - x_0) \cdot \frac{1 + bt_k x_0}{bt_k - x_0} = 0, \end{aligned} \tag{3.55}$$

for $k = 1, \ldots, n$. Clearly, then, $Q(x)$ satisfies the conditions

$$\begin{aligned} Q(x_0) &= 1 + x_0^2 \\ Q(bt_1) &= 0 \\ &\vdots \\ Q(bt_n) &= 0. \end{aligned} \tag{3.56}$$

It is clear that the function

$$(1 + x_0^2) \cdot \frac{T_n(x/b)}{T_n(x_0/b)} \tag{3.57}$$

is also a polynomial of degree n which satisfies the $n+1$ conditions (3.56). Therefore,

$$Q(x) \equiv (1 + x_0^2) \cdot \frac{T_n(x/b)}{T_n(x_0/b)}, \tag{3.58}$$

and (3.53) follows as an immediate consequence of (3.54) and (3.58). □

Lemma 3.10 *Suppose that $n \geq 2$, and that $b > 0$ and x_0 are real numbers with $|x_0| \geq 3b$. Then,*

$$\left| \frac{1 + xx_0}{x - x_0} - \sum_{j=1}^{n} \frac{1 + bt_j x_0}{bt_j - x_0} \cdot u_j\left(\frac{x}{b}\right) \right| < \frac{1 + 9b^2}{b \cdot 5^n} \tag{3.59}$$

for any $x \in [-b, b]$.

Proof. Dividing (3.53) by $(x - x_0)$ and taking absolute values, we obtain

$$\left| \frac{1 + xx_0}{x - x_0} - \sum_{j=1}^{n} \frac{1 + bt_j x_0}{bt_j - x_0} \cdot u_j\left(\frac{x}{b}\right) \right| = \frac{1 + x_0^2}{|x - x_0|} \cdot \frac{|T_n(x/b)|}{|T_n(x_0/b)|}. \tag{3.60}$$

Due to Lemmas 3.5 and 3.6 we have

$$|T_n(x/b)| \leq 1 \tag{3.61}$$

for any $x \in [-b, b]$, and

$$\left| \frac{1 + x_0^2}{T_n(x_0/b)} \right| < (1 + x_0^2) \cdot 2 \cdot \left| \frac{3b}{5x_0} \right|^n < \frac{2}{5^n} \cdot (1 + (3b)^2) \tag{3.62}$$

for any $|x_0| \geq 3b$. Finally, substituting (3.61) and (3.62) into (3.60), we obtain

$$\left| \frac{1 + xx_0}{x - x_0} - \sum_{j=1}^{n} \frac{1 + bt_j x_0}{bt_j - x_0} \cdot u_j\left(\frac{x}{b}\right) \right| < \frac{1 + 9b^2}{b \cdot 5^n} \tag{3.63}$$

for any $x \in [-b, b]$. □

Theorem 3.11 *Suppose that $n \geq 2$, and that a and θ_0 are real numbers with $0 < a \leq \frac{\pi}{8}$ and $3a \leq |\theta_0| \leq \frac{\pi}{2}$. Then,*

$$\left| \frac{1}{\tan(\theta - \theta_0)} - \sum_{j=1}^{n} \frac{1 + t_j \tan a \tan \theta_0}{t_j \tan a - \tan \theta_0} \cdot u_j \left(\frac{\tan \theta}{\tan a} \right) \right| < \frac{1 + 9 \tan^2 a}{\tan a \cdot 5^n} \tag{3.64}$$

for any $\theta \in [-a, a]$.

Proof. Let $\theta \in [-a, a]$. Then, defining the real numbers b, x, and x_0 by the formulae $b = \tan a$, $x = \tan \theta$ and $x_0 = \tan \theta_0$, we observe that $|x| \leq b$, and, due to Lemma 3.8, $|x_0| \geq \tan 3a \geq 3b$. We also observe that

$$\frac{1}{\tan(\theta - \theta_0)} = \frac{1 + \tan \theta \tan \theta_0}{\tan \theta - \tan \theta_0} = \frac{1 + x x_0}{x - x_0}, \tag{3.65}$$

and

$$\sum_{j=1}^{n} \frac{1 + t_j \tan a \tan \theta_0}{t_j \tan a - \tan \theta_0} \cdot u_j \left(\frac{\tan \theta}{\tan a} \right) = \sum_{j=1}^{n} \frac{1 + t_j b x_0}{t_j b - x_0} \cdot u_j \left(\frac{x}{b} \right). \tag{3.66}$$

It follows from the combination of equations (3.65) and (3.66) and Lemma 3.10 that

$$\begin{aligned} & \left| \frac{1}{\tan(\theta - \theta_0)} - \sum_{j=1}^{n} \frac{1 + t_j \tan a \tan \theta_0}{t_j \tan a - \tan \theta_0} \cdot u_j \left(\frac{\tan \theta}{\tan a} \right) \right| \\ = & \left| \frac{1 + x x_0}{x - x_0} - \sum_{j=1}^{n} \frac{1 + t_j b x_0}{t_j b - x_0} \cdot u_j \left(\frac{x}{b} \right) \right| \\ < & \frac{1 + 9b^2}{b \cdot 5^n} = \frac{1 + 9 \tan^2 a}{\tan a \cdot 5^n} \end{aligned} \tag{3.67}$$

for any $\theta \in [-a, a]$. □

The following two lemmas provide preliminary results which are used in the proof of Theorem 3.14.

Lemma 3.12 *Suppose that $n \geq 2$, and that $b > 0$ and x_0 are real numbers with $|x_0| \leq b$. Then, for any x,*

$$x + 3bx_0 - (3b - x x_0) \cdot \sum_{j=1}^{n} \frac{t_j + 3b x_0}{3b - t_j x_0} \cdot u_j(x) = \left(\frac{3b}{x_0} + 3b x_0 \right) \cdot \frac{T_n(x)}{T_n(3b/x_0)}. \tag{3.68}$$

Proof. Let $Q(x)$ be the poly. omial of degree n defined by the formula

$$Q(x) = x + 3bx_0 - (3b - xx_0) \cdot \sum_{j=1}^{n} \frac{t_j + 3bx_0}{3b - t_j x_0} \cdot u_j(x). \tag{3.69}$$

It follows from the combination of (3.43) and (3.69) that

$$\begin{aligned} Q(t_k) &= t_k + 3bx_0 - (3b - t_k x_0) \cdot \sum_{j=1}^{n} \frac{t_j + 3bx_0}{3b - t_j x_0} \cdot u_j(t_k) \\ &= t_k + 3bx_0 - (3b - t_k x_0) \cdot \frac{t_k + 3bx_0}{3b - t_k x_0} = 0, \end{aligned} \tag{3.70}$$

for $k = 1, \ldots, n$. Clearly, then, $Q(x)$ satisfies the conditions

$$\begin{aligned} Q(3b/x_0) &= 3b/x_0 + 3bx_0 \\ Q(t_1) &= 0 \\ &\vdots \\ Q(t_n) &= 0. \end{aligned} \tag{3.71}$$

It is clear that the function

$$\left(\frac{3b}{x_0} + 3bx_0\right) \cdot \frac{T_n(x)}{T_n(3b/x_0)} \tag{3.72}$$

is also a polynomial of degree n which satisfies the $n+1$ conditions (3.71). Therefore,

$$Q(x) \equiv \left(\frac{3b}{x_0} + 3bx_0\right) \cdot \frac{T_n(x)}{T_n(3b/x_0)}, \tag{3.73}$$

and (3.68) follows as an immediate consequence of (3.69) and (3.73). □

Lemma 3.13 *Suppose that $n \geq 2$, and that $b > 0$ and x_0 are real numbers with $|x_0| \leq b$. Then,*

$$\left| \frac{x + 3bx_0}{3b - xx_0} - \sum_{j=1}^{n} \frac{t_j + 3bx_0}{3b - t_j x_0} \cdot u_j(x) \right| < \frac{3(1 + b^2)}{b \cdot 5^n} \tag{3.74}$$

for any $x \in [-1, 1]$.

Proof. Dividing (3.68) by $(3b - xx_0)$ and taking absolute values, we obtain

$$\left|\frac{x+3bx_0}{3b-xx_0} - \sum_{j=1}^{n} \frac{t_j+3bx_0}{3b-t_jx_0}\cdot u_j(x)\right| = \frac{1}{|3b-xx_0|}\cdot\left|\frac{3b}{x_0}+3bx_0\right|\cdot\frac{|T_n(x)|}{|T_n(3b/x_0)|}. \tag{3.75}$$

In addition, due to Lemmas 3.5 and 3.6 we have

$$|T_n(x)| \le 1 \tag{3.76}$$

for any $x \in [-1,1]$, and

$$\left|\frac{3b/x_0+3bx_0}{T_n(3b/x_0)}\right| < 3b\cdot(1/x_0+x_0)\cdot 2\cdot\left|\frac{3x_0}{5\cdot 3b}\right|^n < \frac{6b}{5^n}\cdot(1/b+b) \tag{3.77}$$

for $|x_0| \le b$. Substituting (3.76) and (3.77) into (3.75), we obtain

$$\left|\frac{x+3bx_0}{3b-xx_0} - \sum_{j=1}^{n} \frac{t_j+3bx_0}{3b-t_jx_0}\cdot u_j(x)\right| \le \frac{1}{2b}\cdot\frac{6b}{5^n}\cdot(1/b+b) = \frac{3(1+b^2)}{b\cdot 5^n} \tag{3.78}$$

for any $x \in [-1,1]$. □

Theorem 3.14 *Suppose that $n \ge 2$, and that a and θ_0 are real numbers with $0 < a \le \frac{\pi}{8}$ and $|\theta_0| \le a$. Then,*

$$\left|\frac{1}{\tan(\theta-\theta_0)} - \sum_{j=1}^{n} \frac{t_j+3\tan a\tan\theta_0}{3\tan a - t_j\tan\theta_0}\cdot u_j\left(\frac{3\tan a}{\tan\theta}\right)\right| < \frac{6}{\sin 2a\cdot 5^n} \tag{3.79}$$

for any θ such that $3a \le |\theta| \le \frac{\pi}{2}$.

Proof. Let θ be any real number such that $3a \le |\theta| \le \frac{\pi}{2}$. Then, defining the real numbers b, x and x_0 by the formulae $b = \tan a$, $x = 3\tan a/\tan\theta$ and $x_0 = \tan\theta_0$, we observe that $|x_0| \le b$, and, due to Lemma 3.8, $|x| \le 1$. We also observe that

$$\frac{1}{\tan(\theta-\theta_0)} = \frac{1+\tan\theta\tan\theta_0}{\tan\theta-\tan\theta_0} = \frac{1+3bx_0/x}{3b/x-x_0} = \frac{x+3bx_0}{3b-xx_0}, \tag{3.80}$$

and

$$\sum_{j=1}^{n} \frac{t_j+3\tan a\tan\theta_0}{3\tan a - t_j\tan\theta_0}\cdot u_j\left(\frac{3\tan a}{\tan\theta}\right) = \sum_{j=1}^{n} \frac{t_j+3bx_0}{3b-t_jx_0}\cdot u_j(x). \tag{3.81}$$

It follows from the combination of equations (3.80) and (3.80) and Lemma 3.12 that

$$\left|\frac{1}{\tan(\theta-\theta_0)}-\sum_{j=1}^{n}\frac{t_j+3\tan a\tan\theta_0}{3\tan a-t_j\tan\theta_0}\cdot u_j\left(\frac{3\tan a}{\tan\theta}\right)\right|$$

$$=\left|\frac{x+3bx_0}{3b-xx_0}-\sum_{j=1}^{n}\frac{t_j+3bx_0}{3b-t_jx_0}\cdot u_j(x)\right| \tag{3.82}$$

$$<\ 3\cdot\frac{1+b^2}{b\cdot 5^n}=\frac{3\sec^2 a}{\tan a\cdot 5^n}=\frac{6}{\sin 2a\cdot 5^n} \tag{3.83}$$

for any θ such that $3a\le|\theta|\le\frac{\pi}{2}$. □

The following three theorems provide formulae for translating along the interval $[-\frac{\pi}{2},\frac{\pi}{2}]$ Chebyshev expansions of the type described in the previous two theorems. Theorem 3.15 provides a formula for translating expansions described in Theorems 3.11, Theorem 3.16 describes a mechanism of converting the expansion of Theorem 3.14 to the expansion of Theorem 3.11, and Theorem 3.17 provides a way of translating the expansion of Theorem 3.14.

Theorem 3.15 *Suppose that $n,N\ge 2$, and let a,c,d be real numbers such that $0<a\le\pi/8$ and $[c-d,c+d]\subset[-a,a]$. Let the function $f:[-\frac{\pi}{2},\frac{\pi}{2}]\to\mathbf{C}$ be defined by the formula*

$$f(\theta)=\sum_{k=1}^{N}\frac{\alpha_k}{\tan(\theta-\theta_k)} \tag{3.84}$$

where $3a\le|\theta_k|\le\frac{\pi}{2}$ for $k=1,\ldots,N$, and $\alpha_1,\ldots,\alpha_N$ is a set of complex numbers. Further, let $\Psi_1,\ldots,\Psi_n$ be a set of complex numbers defined by the formula

$$\Psi_k=f(\arctan(t_k\tan a)) \tag{3.85}$$

for $k=1,\ldots,n$, and let $\tilde{\Psi}_1,\ldots,\tilde{\Psi}_n$ be a set of complex numbers defined by the formula

$$\tilde{\Psi}_k=\sum_{j=1}^{n}\Psi_j\cdot u_j\left(\frac{\tan(c+\arctan(t_k\tan d))}{\tan a}\right) \tag{3.86}$$

for $k=1,\ldots,n$. Then, for any $\theta\in[c-d,c+d]$,

$$\left|f(\theta)-\sum_{k=1}^{n}\tilde{\Psi}_k\cdot u_j\left(\frac{\tan(\theta-c)}{\tan d}\right)\right|<A\cdot\frac{(n+1)(1+9\tan^2 a)}{\tan a\cdot 5^n}, \tag{3.87}$$

where $A=\sum_{k=1}^{N}|\alpha_k|$.

Proof. It follows from the triangle inequality that

$$\left| f(\theta) - \sum_{k=1}^{n} \tilde{\Psi}_k \cdot u_j \left(\frac{\tan(\theta - c)}{\tan d} \right) \right| \leq S_1 + S_2 \tag{3.88}$$

where

$$S_1 = \left| f(\theta) - \sum_{k=1}^{n} f(c + \arctan(t_k \tan d)) \cdot u_j \left(\frac{\tan(\theta - c)}{\tan d} \right) \right|, \tag{3.89}$$

and

$$S_2 = \left| \sum_{k=1}^{n} (f(c + \arctan(t_k \tan d)) - \tilde{\Psi}_k) \cdot u_j \left(\frac{\tan(\theta - c)}{\tan d} \right) \right|. \tag{3.90}$$

Combining Theorem 3.11 with Lemma 3.7 and the triangle inequality, we have

$$S_1 < A \cdot \frac{1 + 9\tan^2 a}{\tan a \cdot 5^n}, \tag{3.91}$$

and

$$\begin{aligned} S_2 &\leq \sum_{k=1}^{n} \left| f(c + \arctan(t_k \tan d)) - \sum_{j=1}^{n} \Psi_j \cdot u_j \left(\frac{\tan(c + \arctan(t_k \tan d))}{\tan a} \right) \right| \\ &< An \cdot \frac{1 + 9\tan^2 a}{\tan a \cdot 5^n}, \end{aligned} \tag{3.92}$$

where $A = \sum_{k=1}^{N} |\alpha_k|$. Finally, substituting (3.91) and (3.92) into (3.88) we obtain

$$\left| f(\theta) - \sum_{k=1}^{n} \tilde{\Psi}_k \cdot u_j \left(\frac{\tan(\theta - c)}{\tan d} \right) \right| < A \cdot (n+1) \cdot \frac{1 + 9\tan^2 a}{\tan a \cdot 5^n} \tag{3.93}$$

for any $\theta \in [c - d, c + d]$. □

Theorem 3.16 *Suppose that $n, N \geq 2$. and let a, c, d be real numbers such that $0 < a \leq \pi/8$ and $|c| - d > 3a$. Let the function $f : [-\frac{\pi}{2}, \frac{\pi}{2}] \to \mathbf{C}$ be defined by the formula*

$$f(\theta) = \sum_{k=1}^{N} \frac{\alpha_k}{\tan(\theta - \theta_k)} \tag{3.94}$$

where $\theta_k \in [-a, a]$ for $k = 1, \ldots, N$, and $\alpha_1, \ldots, \alpha_N$ is a set of complex numbers. Further, let $\Phi_1, \ldots, \Phi_n$ be a set of complex numbers defined by the formula

$$\Phi_k = f(\arctan(3\tan(a)/t_k)) \tag{3.95}$$

for $k = 1, \ldots, n$, and let $\Psi_1, \ldots, \Psi_n$ be a set of complex numbers defined by the formula

$$\Psi_k = \sum_{j=1}^{n} \Phi_j \cdot u_j \left(\frac{3 \tan a}{\tan(c + \arctan(t_k \tan d))} \right) \tag{3.96}$$

for $k = 1, \ldots, n$. Then, for any $\theta \in [c - d, c + d]$,

$$\left| f(\theta) - \sum_{k=1}^{n} \Psi_k \cdot u_j \left(\frac{\tan(\theta - c)}{\tan d} \right) \right| < A \cdot \frac{3n \sec^2 a + 1 + 9 \tan^2 a}{\tan a \cdot 5^n} \tag{3.97}$$

where $A = \sum_{k=1}^{N} |\alpha_k|$.

Proof. It follows from the triangle inequality that

$$\left| f(\theta) - \sum_{k=1}^{n} \Psi_k \cdot u_j \left(\frac{\tan(\theta - c)}{\tan d} \right) \right| \leq S_1 + S_2 \tag{3.98}$$

where

$$S_1 = \left| f(\theta) - \sum_{k=1}^{n} f(c + \arctan(t_k \tan d)) \cdot u_j \left(\frac{\tan(\theta - c)}{\tan d} \right) \right|, \tag{3.99}$$

and

$$S_2 = \left| \sum_{k=1}^{n} (f(c + \arctan(t_k \tan d)) - \Psi_k) \cdot u_j \left(\frac{\tan(\theta - c)}{\tan d} \right) \right|. \tag{3.100}$$

Combining Theorem 3.11 with the triangle inequality gives us

$$S_1 < A \cdot \frac{1 + 9 \tan^2 a}{\tan a \cdot 5^n}, \tag{3.101}$$

and from the combination of Theorem 3.14, Lemma 3.7 and the triangle inequality, we have

$$\begin{aligned} S_2 &\leq \sum_{k=1}^{n} \left| f(c + \arctan(t_k \tan d)) - \sum_{j=1}^{n} \Phi_j \cdot u_j \left(\frac{3 \tan a}{\tan(c + \arctan(t_k \tan d))} \right) \right| \\ &< An \cdot \frac{3 \sec^2 a}{\tan a \cdot 5^n}, \end{aligned} \tag{3.102}$$

where $A = \sum_{k=1}^{N} |\alpha_k|$. Finally, substituting (3.101) and (3.102) into (3.98) we obtain

$$\left| f(\theta) - \sum_{k=1}^{n} \Psi_k \cdot u_j \left(\frac{\tan(\theta - c)}{\tan d} \right) \right| < A \cdot \frac{3n \sec^2 a + 1 + 9 \tan^2 a}{\tan a \cdot 5^n} \tag{3.103}$$

for any $\theta \in [c - d, c + d]$. □

Theorem 3.17 *Suppose that $n, N \geq 2$, and let a, c, d be real numbers such that $0 < a \leq \pi/8$ and $[c-d, c+d] \supset [-a, a]$. Let the function $f : [-\frac{\pi}{2}, \frac{\pi}{2}] \to \mathbf{C}$ be defined by the formula*

$$f(\theta) = \sum_{k=1}^{N} \frac{\alpha_k}{\tan(\theta - \theta_k)} \tag{3.104}$$

where $\theta_k \in [-a, a]$ for $k = 1, \ldots, N$, and $\alpha_1, \ldots, \alpha_N$ is a set of complex numbers. Further, let $\Phi_1, \ldots, \Phi_n$ be a set of complex numbers defined by the formula

$$\Phi_k = f(\arctan(3\tan(a)/t_k)) \tag{3.105}$$

for $k = 1, \ldots, n$, and let $\tilde{\Phi}_1, \ldots, \tilde{\Phi}_n$ be a set of complex numbers defined by the formula

$$\tilde{\Phi}_k = \sum_{j=1}^{n} \Phi_j \cdot u_j \left(\frac{3\tan a}{\tan(c + \arctan(3\tan(d)/t_k))} \right) \tag{3.106}$$

for $k = 1, \ldots, n$. Then, for any θ such that $|\theta - c| \geq 3d$,

$$\left| f(\theta) - \sum_{k=1}^{n} \tilde{\Phi}_k \cdot u_j \left(\frac{3\tan d}{\tan(\theta - c)} \right) \right| < A \cdot \frac{3(n+1)\sec^2 a}{\tan a \cdot 5^n}, \tag{3.107}$$

where $A = \sum_{k=1}^{N} |\alpha_k|$.

Proof. It follows from the triangle inequality that

$$\left| f(\theta) - \sum_{k=1}^{n} \tilde{\Phi}_k \cdot u_j \left(\frac{3\tan d}{\tan(\theta - c)} \right) \right| \leq S_1 + S_2 \tag{3.108}$$

where

$$S_1 = \left| f(\theta) - \sum_{k=1}^{n} f(c + \arctan(3\tan(d)/t_k)) \cdot u_j \left(\frac{3\tan d}{\tan(\theta - c)} \right) \right|, \tag{3.109}$$

and

$$S_2 = \left| \sum_{k=1}^{n} \left(f(c + \arctan(3\tan(d)/t_k)) - \tilde{\Phi}_k \right) \cdot u_j \left(\frac{3\tan d}{\tan(\theta - c)} \right) \right|. \tag{3.110}$$

Combining Theorem 3.14 with Lemma 3.7 and the triangle inequality, we have

$$S_1 < A \cdot \frac{3\sec^2 a}{\tan a \cdot 5^n}, \tag{3.111}$$

and

$$S_2 \leq \sum_{k=1}^{n} \left| f(c + \arctan(3\tan(d)/t_k)) - \sum_{j=1}^{n} \Phi_j \cdot u_j \left(\frac{3\tan a}{\tan(c + \arctan(3\tan(d)/t_k))} \right) \right|$$
$$< An \cdot \frac{3\sec^2 a}{\tan a \cdot 5^n}, \tag{3.112}$$

where $A = \sum_{k=1}^{N} |\alpha_k|$. Finally, substituting (3.111) and (3.112) into (3.108) we obtain

$$\left| f(\theta) - \sum_{k=1}^{n} \tilde{\Phi}_k \cdot u_j \left(\frac{3\tan d}{\tan(\theta - c)} \right) \right| < A \cdot \frac{3(n+1)\sec^2 a}{\tan a \cdot 5^n}, \tag{3.113}$$

for any θ such that $|\theta - c| \geq 3d$. □

3.2 Application of the FMM to Nonequispaced FFTs

This section consists of four parts. In Subsection 3.2.1 we describe briefly how the one dimensional Fast Multipole algorithm of Chapter 2 can be applied to the problems of this chapter, in Subsection 3.2.2 we outline a set of four algorithms for these problems, Subsection 3.2.3 contains more formal descriptions of these algorithms, and finally in Subsection 3.2.4 we discuss a generalization of Problems 3.1–3.4.

3.2.1 FMM and Trigonometric Interpolation

There exist a number of different formulations of the trigonometric interpolation problem (see [23]). The version we will use for the purposes of this chapter is described as follows: given a set of points $\{y_1, \ldots, y_N\}$ and function values $\{f_1, \ldots, f_N\}$, evaluate the interpolating Fourier series at the points $\{x_1, \ldots, x_N\}$. According to Theorem 3.3, these values are given by the formulae

$$g_l = c_l \cdot \sum_{j=1}^{N} f_j \cdot d_j \cdot \left(\frac{1}{\tan((x_l - y_j)/2)} - i \right) \tag{3.114}$$

for $l = 1, \ldots, N$, where $\{c_l\}$ and $\{d_j\}$ are defined by the formulae

$$c_l = \prod_{k=1}^{N} \sin((x_l - y_k)/2) = e^{\sum_{k=1}^{N} \ln(\sin((x_l - y_k)/2))} \tag{3.115}$$

for $l = 1, \ldots, N$, and

$$d_j = \prod_{\substack{k=1 \\ k \neq j}}^{N} \frac{1}{\sin((y_j - y_k)/2)} = e^{-\sum_{k=1, k\neq j}^{N} \ln(\sin((y_j - y_k)/2))} \tag{3.116}$$

for $j = 1, \ldots, N$.

Remark 3.3 The FMM algorithms of Chapter 2 are designed to evaluate expressions of the form

$$\sum_{k=1}^{N} \frac{\alpha_k}{x - x_k} \tag{3.117}$$

in $O\left(N \log\left(\frac{1}{\varepsilon}\right)\right)$ arithmetic operations, where ε is the desired accuracy. With a few minor modifications they can also be used to evaluate expressions of the form

$$\sum_{k=1}^{N} \frac{\alpha_k}{\tan(x - x_k)} \tag{3.118}$$

and

$$\sum_{k=1}^{N} \ln(\sin(x - x_k)), \tag{3.119}$$

and hence expressions of the form (3.114) for the same computational cost. Moreover, the algorithmic procedure for the kernel $1/\tan x$ is virtually identical to that for $1/x$, and the various expansions required by the algorithms are manipulated via Theorems 3.15, 3.16 and 3.17 (see Chapter 2 for detailed descriptions of these algorithms).

3.2.2 Informal Descriptions of the Algorithms

In this subsection we outline how a fast trigonometric interpolation scheme can be used to construct efficient algorithms for Problems 3.1–3.4 of this chapter.

We begin with some notation.

$\mathcal{F} : \mathbf{C}^N \rightarrow \mathbf{C}^N$ will denote the matrix which maps a sequence of N complex numbers to its discrete Fourier transform. $\mathcal{F}$ is defined by the formulae

$$\mathcal{F}_{jk} = e^{2\pi i \cdot (j - N/2 - 1) \cdot (k - N/2 - 1)/N} \tag{3.120}$$

for $j, k = 1, \ldots, N$, and it is well known that $\mathcal{F}^T = \mathcal{F}$, and that $\mathcal{F}^{-1} = \frac{1}{N}\bar{\mathcal{F}}$.

Remark 3.4 $\mathcal{F}$ and $\mathcal{F}^{-1}$ can each be applied in $O(N \log N)$ operations via the FFT.

$\mathcal{P} : \mathbf{C}^N \rightarrow \mathbf{C}^N$ will denote the matrix which maps the values of an N-term Fourier series at N equispaced points $\{y_1, \ldots, y_N\}$ on $[-\pi, \pi]$ to the values of this series at the arbitrarily spaced points $\{x_1, \ldots, x_N\}$. According to Theorem 3.4, $\mathcal{P}$ is defined by the formulae

$$\mathcal{P}_{jk} = \sin\left(\frac{Nx_j}{2}\right) \cdot \frac{(-1)^k}{N} \cdot \left(\frac{1}{\tan((x_j - y_k)/2)} - i\right) \tag{3.121}$$

for $j, k = 1, \ldots, N$. It follows directly from (3.121) that

$$\mathcal{P}^T_{jk} = \frac{(-1)^j}{N} \cdot \sin\left(\frac{Nx_k}{2}\right) \cdot \left(\frac{1}{\tan((x_k - y_j)/2)} - i\right) \tag{3.122}$$

for $j, k = 1, \ldots, N$. The inverse of the mapping $\mathcal{P}$ converts the values of an N-term Fourier series at the points $\{x_1, \ldots, x_N\}$ to the values of this series at the equispaced points $\{y_1, \ldots, y_N\}$. $\mathcal{P}^{-1}$ is therefore given analytically, and according to Theorem 3.4 it is defined by the formulae

$$\mathcal{P}^{-1}_{jk} = c_j \cdot d_k \cdot \left(\frac{1}{\tan((y_j - x_k)/2)} - i\right) \tag{3.123}$$

for $j, k = 1, \ldots, N$, where $c_1, \ldots, c_N$ and $d_1, \ldots, d_N$ are sequences of real numbers defined by the formulae (3.115) and (3.116). It follows directly from (3.123) that

$$(\mathcal{P}^T)^{-1}_{jk} = d_j \cdot c_k \cdot \left(\frac{1}{\tan((y_k - x_j)/2)} - i\right) \tag{3.124}$$

for $j, k = 1, \ldots, N$.

Remark 3.5 $\mathcal{P}$, $\mathcal{P}^T$, $\mathcal{P}^{-1}$ and $(\mathcal{P}^T)^{-1}$ are all of the same form, and each can be applied with a relative precision ε in $O\left(N \log\left(\frac{1}{\varepsilon}\right)\right)$ operations via the FMM (see Section 3.2.1).

Observation 3.6 *From the combination of (3.1), Theorems 3.3 and 3.4, and several elementary matrix identities, we see that*

$$\begin{aligned} F &= \mathcal{P} \cdot \mathcal{F} \\ F^T &= \mathcal{F} \cdot \mathcal{P}^T \\ F^{-1} &= \mathcal{F}^{-1} \cdot \mathcal{P}^{-1} \\ (F^T)^{-1} &= (\mathcal{P}^T)^{-1} \cdot \mathcal{F}^{-1}. \end{aligned} \tag{3.125}$$

Furthermore, due to Remarks 3.4 and 3.5, F, F^T, F^{-1} and $(F^T)^{-1}$ can each be applied in $O\left(N \log\left(\frac{1}{\varepsilon}\right)\right)$ arithmetic operations.

3.2.3 Formal Descriptions of the Algorithms

Following are detailed descriptions of the four algorithms of this chapter.

Algorithm 3.1

Step	Complexity	Description
1	$O(N \log N)$	**Comment** [Evaluate Fourier series at equispaced points using FFT.] Compute $g_j = \sum_{k=-N/2}^{N/2-1} \alpha_k \cdot e^{iky_j}$ for $j = 1, \ldots, N$.
2	$O(N \log(\frac{1}{\epsilon}))$	**Comment** [Interpolate in space domain.] **do** $j = 1, N$ $\quad g_j = g_j \cdot (-1)^j / N$ **end do** Compute $f_l = \sum_{j=1}^{N} g_j / \tan((x_l - y_j)/2)$ for $l = 1, \ldots, N$ using FMM. **do** $l = 1, N$ $\quad f_l = f_l - i \cdot \sum_{j=1}^{N} g_j$ **end do** **do** $l = 1, N$ $\quad f_l = f_l \cdot \sin(N x_l / 2)$ **end do**
Total	$O(N \cdot \log N + N \cdot \log(\frac{1}{\epsilon}))$	

Algorithm 3.2

Step	Complexity	Description
1	$O(N \log(\frac{1}{\epsilon}))$	**Comment** [Interpolate in frequency domain.] **do** $j = 1, N$ $\quad \alpha_j = \alpha_j \cdot \sin(N x_j / 2)$ **end do** Compute $a_l = -\sum_{j=1}^{N} \alpha_j / \tan((y_l - x_j)/2)$ for $l = 1, \ldots, N$ using FMM. **do** $l = 1, N$ $\quad a_l = a_l - i \cdot \sum_{j=1}^{N} \alpha_j$ **end do** **do** $l = 1, N$ $\quad a_l = a_l \cdot (-1)^l / N$ **end do**
2	$O(N \log N)$	**Comment** [Evaluate Fourier series at equispaced points using FFT.] Compute $f_j = \sum_{k=-N/2}^{N/2-1} a_k \cdot e^{iky_j}$ for $j = 1, \ldots, N$.
Total	$O(N \cdot \log N + N \cdot \log(\frac{1}{\epsilon}))$	

Algorithm 3.3

Step	Complexity	Description
1	$O(N \log(\frac{1}{\epsilon}))$	**Comment** [Interpolate in space domain.] **do** $j = 1, N$ $f_j = f_j \cdot d_j$ **end do** Compute $a_l = \sum_{j=1}^{N} f_j / \tan((y_l - x_j)/2)$ for $l = 1, \ldots, N$ using FMM. **do** $l = 1, N$ $a_l = a_l - i \cdot \sum_{j=1}^{N} f_j$ **end do** **do** $l = 1, N$ $a_l = a_l \cdot c_l$ **end do**
2	$O(N \log N)$	**Comment** [Obtain Fourier coeffients using FFT.] Compute $\alpha_j = \frac{1}{N} \cdot \sum_{k=1}^{N} a_k \cdot e^{-iky_j}$ for $j = -N/2, \ldots, N/2 - 1$.
Total	$O(N \cdot \log N + N \cdot \log(\frac{1}{\epsilon}))$	

Algorithm 3.4

Step	Complexity	Description
1	$O(N \log N)$	**Comment** [Obtain Fourier coeffients using FFT.] Compute $a_j = \frac{1}{N} \cdot \sum_{k=1}^{N} f_k \cdot e^{-iky_j}$ for $j = 1, \ldots, N$.
2	$O(N \log(\frac{1}{\epsilon}))$	**Comment** [Interpolate in frequency domain.] **do** $j = 1, N$ $a_j = a_j \cdot c_j$ **end do** Compute $\alpha_l = -\sum_{j=1}^{N} a_j / \tan((x_l - y_j)/2)$ for $l = 1, \ldots, N$ using FMM. **do** $l = 1, N$ $\alpha_l = \alpha_l - i \cdot \sum_{j=1}^{N} a_j$ **end do** **do** $l = 1, N$ $\alpha_l = \alpha_l \cdot d_l$ **end do**
Total	$O(N \cdot \log N + N \cdot \log(\frac{1}{\epsilon}))$	

3.2.4 FFTs for Complex Data Points

Various generalizations of the problems addressed in this chapter are mentioned briefly in Section 6. One of the generalizations of Problems 3.1–3.4 merits special attention,

and is discussed in this section: this is the case when the points $\{x_j\}$ are complex, and lie slightly off the real axis.

We are interested here in the transformations described by the formulae

$$f_j = \sum_{k=-N/2}^{N/2-1} \alpha_k \cdot e^{ikr_j} \cdot e^{-ks_j}, \tag{3.126}$$

for $j = 1, \ldots, N$, which is a generalization of (3.1) with

$$x_j = r_j + is_j. \tag{3.127}$$

Algorithms 3.1–3.4 can be modified to evaluate expressions of the form (3.126), provided that the s_j are small (on the order of $\frac{1}{N}$).

Problems of this type are frequently encountered in signal analysis, computational complex analysis and several other areas.

Chapter 4

Nonequispaced FFTs: An Alternative Approach

In this chapter, we present another set of algorithms for evaluating series of the form (1.1). This approach uses an interpolation scheme based on the Fourier analysis of Gaussian bells in place of the FMM-based interpolation scheme of Chapter 3. The two approaches are compared in Chapter 6.

For the remainder of the chapter 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 the Fourier matrix and its transpose, i.e. we are interested in the transformations $F, G : \mathbf{C}^{N+1} \to \mathbf{C}^{N+1}$ defined by the formulae

$$f_j = F(\alpha)_j = \sum_{k=0}^{N} \alpha_k \cdot e^{i\omega_k \cdot 2\pi j/N} \tag{4.1}$$

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

$$g_j = G(\beta)_j = \sum_{k=-N/2}^{N/2} \beta_k \cdot e^{ikx_j} \tag{4.2}$$

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

Remark 4.1 If $x_k = -\omega_k \cdot 2\pi/N$ for $k = 0, \ldots, N$, then (4.1) can be rewritten as

$$f_j = \sum_{k=0}^{N} \alpha_k \cdot e^{-ijx_k}, \tag{4.3}$$

or alternatively as $F = G^*$.

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

$$h_j = H(\gamma)_j = \sum_{k=0}^{N} \gamma_k \cdot e^{i\omega_k x_j}. \tag{4.4}$$

More formally, we consider the following problems

- **Problem 4.1:** Given α, find $f = F(\alpha)$.
- **Problem 4.2:** Given β, find $g = G(\beta)$.
- **Problem 4.3:** Given γ, find $h = H(\gamma)$.

The plan of this chapter is as follows. We start in Section 4.1 with some results from analysis and approximation theory which are used in the design of the algorithms. This is followed by an informal description of the algorithms in Section 4.2. In Section 4.3 we introduce some notation which is used in a set of more detailed algorithm descriptions in Section 4.4.

4.1 Mathematical and Numerical Preliminaries

4.1.1 Elementary Analytical Tools

In this subsection we summarize some well-known results to be used in the remainder of this chapter. Lemmas 4.1 and 4.2 are obvious, and Lemmas 4.3 and 4.4 can be found, for example, in [14].

Lemma 4.1 *For any real c,*

$$\int_{-\pi}^{\pi} e^{icx} dx = \frac{2}{c} \sin(c\pi). \tag{4.5}$$

Lemma 4.2 *For any integer k,*

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} e^{ikx} dx = \begin{cases} 1 & \text{if } k = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

Lemma 4.3 *For any real $b > 0$ and complex z,*

$$\int_{-\infty}^{\infty} e^{-bx^2} \cdot e^{zx} dx = \sqrt{\frac{\pi}{b}} \cdot e^{z^2/4b}. \tag{4.7}$$

Lemma 4.4 *For any real $b > 0$ and $a > 0$,*

$$\int_{a}^{\infty} e^{-bx^2} dx < \frac{e^{-ba^2}}{2ba}. \tag{4.8}$$

Proof.

$$\int_{a}^{\infty} e^{-bx^2} dx = \int_{0}^{\infty} e^{-b(x+a)^2} dx < e^{-ba^2} \int_{0}^{\infty} e^{-2bax} dx = \frac{e^{-ba^2}}{2ba}. \tag{4.9}$$

□

4.1.2 Relevant Facts from Approximation Theory

The principal tool of this chapter is a somewhat detailed analysis of Fourier series of functions $\phi : [-\pi, \pi] \to \mathbf{C}$ given by the formula

$$\phi(x) = e^{-bx^2} \cdot e^{icx} \tag{4.10}$$

where $b > \frac{1}{2}$ and c are real numbers. We present this analysis in the lemmas and theorems of this subsection, numbered 4.5–4.10.

Lemmas 4.5 and 4.6 provide two inequalities which are used in Theorem 4.7, and Theorems 4.7–4.9 are intermediate results leading to Theorem 4.10. This final theorem explains how to approximate functions of the form e^{icx} using a small number of terms, and the algorithms of this chapter are based upon this result. We derive error bounds for all approximations which allow us to perform numerical computations to any specified accuracy.

Lemma 4.5 *For any real $b > \frac{1}{2}, c$ and any integer k,*

$$\left| 2 \int_{\pi}^{\infty} e^{-bx^2} \cos((c-k)x) dx + e^{-b\pi^2} \cdot \int_{-\pi}^{\pi} e^{i(c-k)x} dx \right| < 2\pi e^{-b\pi^2} \cdot \left(1 + \frac{1}{\pi^2}\right). \tag{4.11}$$

Proof. Using the triangle inequality and Lemma 4.4 we have

$$\begin{aligned} & \left| 2 \int_{\pi}^{\infty} e^{-bx^2} \cdot \cos((c-k)x) dx + e^{-b\pi^2} \cdot \int_{-\pi}^{\pi} e^{i(c-k)x} dx \right| \\ & \leq 2 \int_{\pi}^{\infty} e^{-bx^2} dx + 2\pi e^{-b\pi^2} < 2\pi e^{-b\pi^2} \cdot \left(\frac{1}{2b\pi^2} + 1\right) \\ & < 2\pi e^{-b\pi^2} \cdot \left(\frac{1}{\pi^2} + 1\right). \end{aligned} \tag{4.12}$$

□

Lemma 4.6 *For any real $b > \frac{1}{2}, c$ and any integer k,*

$$\left|2\int_{\pi}^{\infty} e^{-bx^2}\cos((c-k)x)dx + e^{-b\pi^2}\cdot\int_{-\pi}^{\pi} e^{i(c-k)x}dx\right| < \frac{8b\pi e^{-b\pi^2}}{(c-k)^2}\cdot\left(1+\frac{1}{\pi^2}\right). \quad (4.13)$$

Proof. Integrating by parts we have

$$\begin{aligned} & 2\int_{\pi}^{\infty} e^{-bx^2}\cdot\cos((c-k)x)dx \\ &= \frac{2}{c-k}\left[e^{-bx^2}\sin((c-k)x)\right]_{\pi}^{\infty} + \frac{4b}{c-k}\int_{\pi}^{\infty} xe^{-bx^2}\sin((c-k)x)dx \quad (4.14) \\ &= -\frac{2}{c-k}e^{-b\pi^2}\sin((c-k)\pi) + \frac{4b}{c-k}\int_{\pi}^{\infty} xe^{-bx^2}\sin((c-k)x)dx. \end{aligned}$$

After rearranging the terms in (4.14) and integrating by parts again we obtain

$$\begin{aligned} & \left|2\int_{\pi}^{\infty} e^{-bx^2}\cos((c-k)x)dx + \frac{2e^{-b\pi^2}}{c-k}\sin((c-k)\pi)\right| \\ &= \left|\frac{4b}{c-k}\int_{\pi}^{\infty} xe^{-bx^2}\sin((c-k)x)dx\right| \\ &= \left|-\frac{4b}{(c-k)^2}\left(\left[xe^{-bx^2}\cos((c-k)x)\right]_{\pi}^{\infty} - \int_{\pi}^{\infty}(1-2bx^2)e^{-bx^2}\cos((c-k)x)dx\right)\right| \quad (4.15) \\ &\le \frac{4b}{(c-k)^2}\left(\pi e^{-b\pi^2} + \int_{\pi}^{\infty} e^{-bx^2}dx + \int_{\pi}^{\infty} x\cdot 2bxe^{-bx^2}dx\right) \\ &< \frac{4b}{(c-k)^2}\left(\pi e^{-b\pi^2} + \int_{\pi}^{\infty} e^{-bx^2}dx + \left[-x\epsilon^{-bx^2}\right]_{\pi}^{\infty} + \int_{\pi}^{\infty} e^{-bx^2}dx\right). \end{aligned}$$

Finally, due to (4.15) and Lemmas 4.1 and 4.4 we have

$$\begin{aligned} \left|2\int_{\pi}^{\infty} e^{-bx^2}\cos((c-k)x)dx + e^{-b\pi^2}\cdot\int_{-\pi}^{\pi} e^{i(c-k)x}dx\right| &< \frac{4be^{-b\pi^2}}{(c-k)^2}\cdot\left(2\pi+\frac{2}{2b\pi}\right) \\ &< \frac{8b\pi e^{-b\pi^2}}{(c-k)^2}\cdot\left(1+\frac{1}{\pi^2}\right) \quad (4.16) \end{aligned}$$

□

The following theorem provides an explicit expression for the coefficients of a Fourier series which approximates functions of the form (4.10).

Theorem 4.7 *Let $\phi(x) = e^{-bx^2}e^{icx}$ for any real $b > \frac{1}{2}, c$. Then, for any $x \in (-\pi, \pi)$,*

$$\left|\phi(x) - \sum_{k=-\infty}^{\infty}\rho_k e^{ikx}\right| < e^{-b\pi^2}\cdot\left(4b+\frac{70}{9}\right), \quad (4.17)$$

where

$$\rho_k = \frac{1}{2\sqrt{b\pi}} e^{-(c-k)^2/4b} \tag{4.18}$$

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

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

$$\phi(x) = \sum_{k=-\infty}^{\infty} \sigma_k e^{ikx}, \tag{4.19}$$

and due to Lemma 4.3 and equation (4.18) we have

$$\begin{aligned}
\sigma_k &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \phi(x) e^{-ikx} dx \\
&= \frac{1}{2\pi} \left(\int_{-\infty}^{\infty} e^{-bx^2} e^{icx} e^{-ikx} dx - \int_{-\infty}^{-\pi} e^{-bx^2} e^{icx} e^{-ikx} dx - \int_{\pi}^{\infty} e^{-bx^2} e^{icx} e^{-ikx} dx \right) \\
&= \frac{1}{2\pi} \left(\sqrt{\frac{\pi}{b}} \cdot e^{-(c-k)^2/4b} + \int_{\infty}^{\pi} e^{-bx^2 - icx + ikx} dx - \int_{\pi}^{\infty} e^{-bx^2 + icx - ikx} dx \right) \qquad (4.20) \\
&= \rho_k - \frac{1}{\pi} \int_{\pi}^{\infty} e^{-bx^2} \cos((c-k)x) dx.
\end{aligned}$$

Rearranging equation (4.20) and applying Lemmas 4.5 and 4.6 we obtain the inequalities

$$\left| \sigma_k - \rho_k - \frac{e^{-b\pi^2}}{2\pi} \int_{-\pi}^{\pi} e^{icx} e^{-ikx} dx \right| < e^{-b\pi^2} \cdot \left(1 + \frac{1}{\pi^2} \right), \tag{4.21}$$

$$\left| \sigma_k - \rho_k - \frac{e^{-b\pi^2}}{2\pi} \int_{-\pi}^{\pi} e^{icx} e^{-ikx} dx \right| < \frac{4be^{-b\pi^2}}{(c-k)^2} \cdot \left(1 + \frac{1}{\pi^2} \right), \tag{4.22}$$

and it now follows from the combination of (4.19), (4.21) and (4.22) that, for any $x \in (-\pi, \pi)$,

$$\begin{aligned}
&\left| \phi(x) - \sum_{k=-\infty}^{\infty} \rho_k e^{ikx} - e^{-b\pi^2} \cdot e^{icx} \right| \\
&= \left| \sum_{k=-\infty}^{\infty} e^{ikx} \cdot \left(\sigma_k - \rho_k - \frac{e^{-b\pi^2}}{2\pi} \int_{-\pi}^{\pi} e^{icx} e^{-ikx} dx \right) \right| \\
&< \sum_{k, |c-k| \geq 3} \frac{4be^{-b\pi^2}}{(c-k)^2} \cdot \left(1 + \frac{1}{\pi^2} \right) + \sum_{k, |c-k| < 3} e^{-b\pi^2} \cdot \left(1 + \frac{1}{\pi^2} \right) \qquad (4.23) \\
&< 4be^{-b\pi^2} \cdot \frac{9}{8} \cdot 2 \cdot \sum_{k=3}^{\infty} \frac{1}{k^2} + 6e^{-b\pi^2} \cdot \frac{10}{9}.
\end{aligned}$$

Some elementary analysis yields

$$\sum_{k=3}^{\infty} \frac{1}{k^2} < \frac{1}{9} + \int_3^{\infty} \frac{dx}{x^2} = \frac{1}{9} + \frac{1}{3} = \frac{4}{9} \tag{4.24}$$

and substituting (4.24) into (4.23) we have

$$\left| \phi(x) - \sum_{k=-\infty}^{\infty} \rho_k e^{ikx} - e^{-b\pi^2} \cdot e^{icx} \right| < e^{-b\pi^2} \cdot \left(4b + \frac{60}{9} \right). \tag{4.25}$$

To complete the proof we make use of the triangle inequality and (4.25) to obtain

$$\begin{aligned} \left| \phi(x) - \sum_{k=-\infty}^{\infty} \rho_k e^{ikx} \right| &\leq \left| \phi(x) - \sum_{k=-\infty}^{\infty} \rho_k e^{ikx} - e^{-b\pi^2} \cdot e^{icx} \right| + \left| e^{-b\pi^2} \cdot e^{icx} \right| \\ &< e^{-b\pi^2} \cdot \left(4b + \frac{70}{9} \right). \end{aligned} \tag{4.26}$$

□

Remark 4.2 According to Theorem 4.7, 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 4.3 The coefficients ρ_k as defined by (4.18) 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

$$q \geq 4b\pi, \tag{4.27}$$

so as to satisfy the inequality

$$e^{-(q/2)^2/4b} \leq e^{-b\pi^2}. \tag{4.28}$$

The following theorem estimates the truncation error under the conditions of Remark 4.3 and thus provides a way of approximating functions of the form (4.10) by a $(q+1)$-term series.

Theorem 4.8 *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 $x \in (-\pi, \pi)$,*

$$\left| \phi(x) - \sum_{k=[c]-q/2}^{[c]+q/2} \rho_k e^{ikx} \right| < e^{-b\pi^2} \cdot (4b + 9), \tag{4.29}$$

where $\{\rho_k\}$ are defined by (4.18).

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

$$\left|\phi(x) - \sum_{k=[c]-q/2}^{[c]+q/2} \rho_k e^{ikx}\right| \leq \left|\phi(x) - \sum_{k=-\infty}^{\infty} \rho_k e^{ikx}\right| + \left|\sum_{k>[c]+q/2} \rho_k e^{ikx}\right| + \left|\sum_{k<[c]-q/2} \rho_k e^{ikx}\right|. \tag{4.30}$$

Due to (4.18) and the triangle inequality we have the inequalities

$$\left|\sum_{k>[c]+q/2} \rho_k e^{ikx}\right| \leq \sum_{k=[c]+q/2+1}^{\infty} \frac{e^{-(c-k)^2/4b}}{2\sqrt{b\pi}} < \sum_{k=q/2}^{\infty} \frac{e^{-k^2/4b}}{\sqrt{2\pi}}, \tag{4.31}$$

$$\left|\sum_{k<[c]-q/2} \rho_k e^{ikx}\right| \leq \sum_{k=-\infty}^{[c]-q/2-1} \frac{e^{-(c-k)^2/4b}}{2\sqrt{b\pi}} < \sum_{k=q/2}^{\infty} \frac{e^{-k^2/4b}}{\sqrt{2\pi}}. \tag{4.32}$$

Some elementary analysis and an application of Lemma 4.4 yields

$$\sum_{k=q/2}^{\infty} e^{-k^2/4b} < e^{-(q/2)^2/4b} + \int_{q/2}^{\infty} e^{-x^2/4b} dx < e^{-(q/2)^2/4b} \cdot \left(1 + \frac{4b}{2q/2}\right), \tag{4.33}$$

and it follows from the combination of (4.27), (4.28) and (4.33) that

$$\sum_{k=q/2}^{\infty} e^{-k^2/4b} < e^{-b\pi^2} \cdot \left(1 + \frac{1}{\pi}\right). \tag{4.34}$$

Substituting (4.34) into (4.31) and (4.32) we have

$$\left|\sum_{k>[c]+q/2} \rho_k e^{ikx}\right| + \left|\sum_{k<[c]-q/2} \rho_k e^{ikx}\right| < \frac{2e^{-b\pi^2}}{\sqrt{2\pi}} \cdot \left(1 + \frac{1}{\pi}\right) < e^{-b\pi^2} \cdot \frac{10}{9}, \tag{4.35}$$

and finally, substituting (4.26) and (4.35) into (4.30), we obtain

$$\left|\phi(x) - \sum_{k=[c]-q/2}^{[c]+q/2} \rho_k e^{ikx}\right| < e^{-b\pi^2} \cdot \left(4b + \frac{70}{9} + \frac{10}{9}\right) < e^{-b\pi^2} \cdot (4b + 9). \tag{4.36}$$

□

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

Corollary 4.9 *Suppose that $m \geq 2$ is an integer and that the conditions of Theorem 4.8 are satisfied. Then, multiplying both sides of (4.29) by e^{bx^2}, we obtain*

$$\begin{aligned} \left| e^{icx} - e^{bx^2} \cdot \sum_{k=[c]-q/2}^{[c]+q/2} \rho_k e^{ikx} \right| &< e^{bx^2} \cdot e^{-b\pi^2} \cdot (4b+9) \\ &< e^{b\pi^2/m^2} \cdot e^{-b\pi^2} \cdot (4b+9) \end{aligned} \tag{4.37}$$

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

Finally, Theorem 4.10 makes use of a simple linear scaling to generalize the inequality (4.37) from $[-\frac{\pi}{m}, \frac{\pi}{m}]$ to any interval $[-d, d]$. This is the principal result of the section.

Theorem 4.10 *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 $x \in [-d, d]$,*

$$\left| e^{icx} - e^{b(x\pi/md)^2} \cdot \sum_{k=[cmd/\pi]-q/2}^{[cmd/\pi]+q/2} \rho_k e^{ikx\pi/md} \right| < e^{-b\pi^2(1-1/m^2)} \cdot (4b+9) \tag{4.38}$$

where $\{\rho_k\}$ are defined by (4.18).

Remark 4.4 The error bounds obtained in the above theorems are rather pessimistic. Numerical estimates for the actual errors can be found in Section 4.5.

4.2 Informal Descriptions of the Algorithms

In this section we give informal outlines of algorithms for Problems 4.1–4.3 of this chapter. More formal descriptions of these algorithms are presented in Section 4.4.

The algorithms for these problems are based on the following principal observation.

Observation 4.5 *According to Theorem 4.10, 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 (4.1), (4.2) and (4.4), we relate these more general cases to the equispaced case of the FFT. Observation 4.5 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 4.6 *The overall complexity of each such algorithm which couples the FFT with the interpolation scheme will be $O(N \log N + Nq)$ operations.*

4.3 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

$$\varepsilon = e^{-b\pi^2(1-1/m^2)} \cdot (4b + 9), \tag{4.39}$$

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

$$q \geq 4b\pi. \tag{4.40}$$

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

$$P_{jk} = \frac{1}{2\sqrt{b\pi}} \cdot e^{-(m\omega_k - (\mu_k + j))^2/4b} \tag{4.41}$$

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

Observation 4.7 *Setting $d = \pi$ in Theorem 4.10 we see that*

$$\left| e^{i\omega_k x} - e^{b(x/m)^2} \cdot \sum_{j=-q/2}^{q/2} P_{jk} \cdot e^{i(\mu_k + j)x/m} \right| < \varepsilon \tag{4.42}$$

for any $k = 0, \ldots, N$ and any $x \in [-\pi, \pi]$, where ε is defined by (4.39).

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

$$\sum_{k=1}^{N} \alpha_k \cdot \sum_{j=-q/2}^{q/2} P_{jk} \cdot e^{i(\mu_k + j)x/m} = \sum_{j=-mN/2}^{mN/2-1} \tau_j e^{ijx/m}, \tag{4.43}$$

so that

$$\tau_l = \sum_{j,k,\mu_k+j=l} \alpha_k \cdot P_{jk}. \tag{4.44}$$

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

$$T_j = \sum_{k=-mN/2}^{mN/2-1} \tau_k \cdot e^{2\pi ikj/mN} \tag{4.45}$$

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

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

$$\tilde{f}_j = e^{b(2\pi j/n_c N)^2} \cdot T_j \tag{4.46}$$

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

Observation 4.8 *Combining (4.42) - (4.46) with the triangle inequality, we see that*

$$|f_j - \tilde{f}_j| < \varepsilon \cdot \sum_{k=0}^{N} |\alpha_k| \tag{4.47}$$

for $j = -N/2, \ldots, N/2$, where $\{f_j = F(\alpha)_j\}$ are defined by (4.1).

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

$$Q_{jk} = \frac{1}{2\sqrt{b\pi}} \cdot e^{-(x_j mN/2\pi - (\nu_j + k))^2/4b} \tag{4.48}$$

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

Observation 4.9 *Setting $d = N/2$ in Theorem 4.10 we see that*

$$\left| e^{ikx_j} - e^{b(2\pi k/mN)^2} \cdot \sum_{l=-q/2}^{q/2} Q_{jk} \cdot e^{i(\nu_j + l)2\pi k/mN} \right| < \varepsilon \tag{4.49}$$

for any $j = 0, \ldots, N$ and any $k \in [-N/2, N/2]$, where ε is defined by (4.39).

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

$$u_k = \beta_k \cdot e^{b(2\pi k/mN)^2} \tag{4.50}$$

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

$$U_l = \sum_{k=-N/2}^{N/2} u_k \cdot e^{2\pi i k l/mN} \tag{4.51}$$

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

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

$$\tilde{g}_j = \sum_{l=-q/2}^{q/2} Q_{jl} \cdot U_{\nu_j+l} \tag{4.52}$$

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

Observation 4.10 *Combining (4.49) - (4.52) with the triangle inequality, we see that*

$$|g_j - \tilde{g}_j| < \varepsilon \cdot \sum_{k=0}^{N} |\beta_k| \tag{4.53}$$

for $j = 0, \ldots, N$, where $\{g_j = G(\beta)_j\}$ are defined by (4.2).

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

$$R_{jk} = \frac{1}{2\sqrt{b\pi}} \cdot e^{-(x_j N/2\pi - (\eta_j + k))^2/4b} \tag{4.54}$$

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

Observation 4.11 *Setting $d = N/2$ in Theorem 4.10 we see that*

$$\left| e^{ikx_j/m} - e^{b(2\pi k/mN)^2} \cdot \sum_{l=-q/2}^{q/2} R_{jk} \cdot e^{i(\eta_j + l)2\pi k/mN} \right| < \varepsilon \tag{4.55}$$

for any $j = 0, \ldots, N$ and any $k \in [-N/2, N/2]$ where ε is defined by (4.39).

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

$$\sum_{k=0}^{N} \gamma_k \cdot \sum_{j=-q/2}^{q/2} P_{jk} \cdot e^{i(\mu_k + j)x/m} = \sum_{j=-mN/2}^{mN/2} v_j \cdot e^{ijx/m}, \tag{4.56}$$

so that

$$v_l = \sum_{j,k,\eta_k + j = l} \gamma_k \cdot P_{jk}. \tag{4.57}$$

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

$$V_l = \sum_{k=-mN/2}^{mN/2} v_k \cdot e^{b(2\pi k/m^2N)^2} \cdot e^{2\pi ikl/m^2N} \tag{4.58}$$

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

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

$$\tilde{h}_j = e^{b(x_j/m)^2} \cdot \sum_{l=-q/2}^{q/2} R_{jl} \cdot V_{\eta_j+l} \tag{4.59}$$

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

Observation 4.12 *Combining (4.42) and (4.55) - (4.59) with the triangle inequality, we see that*

$$|h_j - \tilde{h}_j| < \delta \cdot \sum_{k=0}^{N} |\gamma_k| \tag{4.60}$$

for $j = 0, \ldots, N$, where $\{h_j = H(\gamma)_j\}$ are defined by (4.4), and

$$\delta = 2e^{-b\pi^2(1-2/m^2)} \cdot (4b + 9). \tag{4.61}$$

4.4 Detailed Descriptions of the Algorithms

This section contains descriptions of algorithms for Problems 4.1–4.3 of this chapter. In the tables below we will make use of the facts that $q \sim \log(\frac{1}{\varepsilon})$ and $m^2 \ll N$.

Remark 4.13 Results of our extensive numerical experiments indicate that $m = 2$ is an optimal choice for both efficiency and accuracy. Each of Algorithms 4.1–4.3 then requires

$$O\left(N \cdot \log N + N \cdot \log\left(\frac{1}{\varepsilon}\right)\right) \tag{4.62}$$

arithmetic operations.

Remark 4.14 The storage requirements of an algorithm are also an important characteristic. From the descriptions of the initialization steps the asymptotic storage requirements for each of Algorithms 4.1–4.3 are of the form

$$\lambda \cdot N \cdot q \tag{4.63}$$

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

Algorithm 4.1

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 (4.41) **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 4.2

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 (4.48) **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 4.3

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 (4.41) **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 (4.54) **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	$O(m^2 N \cdot \log N + N \cdot \log(\frac{1}{\epsilon}))$	

4.5 Numerical Estimates of Error Bounds

In this section we present numerical estimates of error bounds for Theorem 4.10 of Section 4.1.2. For our experiments we chose $c = 0$ and $d = \pi$, and chose two different sets of values of m, b and q. The expression

$$r(x) = 1 - e^{bx^2} \cdot \sum_{k=-q/2}^{q/2} \rho_k \cdot e^{ikx}, \tag{4.64}$$

where ρ_k is defined by (4.18), 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

$$E_\infty = \max_{1 \le k \le n} |r(x_k)|, \tag{4.65}$$

- the relative L_2 error E_2, defined by the formula

$$E_2 = \sqrt{\frac{\sum_{k=1}^{n} |r(x_k)|^2}{n}}, \tag{4.66}$$

- the error bound E_B of Theorem 4.10, defined by the formula

$$E_B = e^{-b\pi^2(1-1/m^2)} \cdot (4b + 9). \tag{4.67}$$

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

Table 4.1: Error Bounds for Theorem 4.10.

The results of this experiment are presented in Table 4.1.

We observe from Table 4.1 that the theoretical error bound E_B of Theorem 4.10 is very weak compared with the experimentally obtained bound. Indeed, the requirement that E_B be appropriately small would impose much larger values of b and q than are actually needed for Algorithms 4.1–4.3.

Chapter 5

Implementation and Numerical Results

We have written FORTRAN implementations of the algorithms of this thesis using double precision arithmetic, and have applied these programs to a variety of situations. In this chapter we will restrict our attention to the implementations of Algorithms 3.1–3.4 and Algorithms 4.1–4.3, and will demonstrate the performance of these algorithms with 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. The parameters for each algorithm were chosen to retain maximum precision. For Algorithms 4.1–4.3 we chose $m = 2$, $b = 1.5629$ and $q = 28$, and for the FMM algorithms of Chapter 2 used by Algorithms 3.1–3.4 we chose $p = 22$, $\hat{p} = 10$, $s = 16$ and $nlevs = \log_2(n/s)$.

3. Algorithms 3.1–3.4 and 4.1–4.3 all require the evaluation of sums of the form

$$f_j = \sum_{k=-N/2}^{N/2-1} \alpha_k \cdot e^{2\pi ikj/N} \tag{5.1}$$

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

$$f_j = \sum_{k=0}^{N-1} \alpha_k \cdot e^{2\pi ikj/N} \tag{5.2}$$

for $j = 0, \ldots, N-1$. We used a standard FFT to evaluate sums of the form (5.1) 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 (5.1) to the form (5.2).

4. The algorithms of this thesis which require an FFT of size proportional to N will perform efficiently whenever the FFT does. This restriction on problem 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.

Our implementations of the algorithms of this thesis have been tested on the the Sun SPARCstation 1 for a variety of input data. Seven experiments are described in this chapter, and their results are summarized in Tables 5.1–5.7. These tables contain error estimates and CPU timings for the algorithms, with all computations performed in double precision arithmetic.

The table entries are described below.

- The first column in each table contains the problem size N.

- The second column in each table contains the relative ∞-norm error defined by the formula

$$E_\infty = \max_{1 \le j \le N} |\tilde{f}_j - f_j| \Big/ \max_{1 \le j \le N} |f_j| \,, \tag{5.3}$$

  where the vector $\tilde{f}$ is the algorithm output and the vector f is the result of a direct calculation.

- The third column in each table contains the relative 2-norm error defined by the formula

$$E_2 = \sqrt{\sum_{j=1}^{N} |\tilde{f}_j - f_j|^2 \Big/ \sum_{j=1}^{N} |f_j|^2} \,, \tag{5.4}$$

  where the vector $\tilde{f}$ is the algorithm output and the vector f is the result of a direct calculation.

- The fourth and fifth columns in each table contain CPU timings for the initialization and evaluation stages of the algorithm.

- The sixth column in each table contains CPU timings for the corresponding direct calculation.

- The last column in each table contains CPU timings for an FFT of the same size.

Remark 5.1 Our implementations of the direct methods for Examples 1, 2, 4 and 5 were optimized by using the fact that $e^{ikx_j} = (e^{ix_j})^k$ to reduce the number of complex exponential computations. In Example 3 however, N^2 exponentials are required for the direct method, 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 5.2 Standard LINPACK Gaussian Elimination subroutines were used as the direct methods for comparing timings in Examples 6 and 7. 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 $F : \mathbf{C}^{N+1} \rightarrow \mathbf{C}^{N+1}$ of Problem 4.1, defined by the formula

$$F(\alpha)_j = \sum_{k=0}^{N} \alpha_k \cdot e^{i\omega_k \cdot 2\pi j/N} \tag{5.5}$$

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 complex numbers randomly chosen from the unit square

$$0 \leq \mathrm{Re}(z) \leq 1,\ 0 \leq \mathrm{Im}(z) \leq 1. \tag{5.6}$$

The results of applying Algorithm 4.1 to this problem are presented in Table 5.1.

Example 2.
Here we consider the transformation $G : \mathbf{C}^{N+1} \rightarrow \mathbf{C}^{N+1}$ of Problem 4.2, defined by the formula

$$G(\beta)_j = \sum_{k=-N/2}^{N/2} \beta_k \cdot e^{ikx_j} \tag{5.7}$$

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 complex numbers randomly chosen from the unit square

$$0 \leq \mathrm{Re}(z) \leq 1,\ 0 \leq \mathrm{Im}(z) \leq 1. \tag{5.8}$$

The results of applying Algorithm 4.2 to this problem are presented in Table 5.2.

Example 3.
Here we consider the transformation $H : \mathbf{C}^{N+1} \to \mathbf{C}^{N+1}$ of Problem 4.3, defined by the formula

$$H(\gamma)_j = \sum_{k=0}^{N} \gamma_k \cdot e^{i\omega_k x_j} \tag{5.9}$$

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 complex numbers randomly chosen from the unit square

$$0 \leq \mathrm{Re}(z) \leq 1,\ 0 \leq \mathrm{Im}(z) \leq 1. \tag{5.10}$$

The results of applying Algorithm 4.3 to this problem are presented in Table 5.3.

Example 4.
Here we consider the transformation $F : \mathbf{C}^N \to \mathbf{C}^N$ of Problem 3.1, defined by the formula

$$F(\alpha)_j = \sum_{k=-N/2}^{N/2-1} \alpha_k \cdot e^{ikx_j} \tag{5.11}$$

for $j = 1, \ldots, N$. In this example, $\{x_1, \ldots, x_N\}$ were randomly distributed on the interval $[-\pi, \pi]$ and $\{\alpha_{-N/2}, \ldots, \alpha_{N/2-1}\}$ were complex numbers randomly chosen from the unit square

$$0 \leq \mathrm{Re}(z) \leq 1,\ 0 \leq \mathrm{Im}(z) \leq 1. \tag{5.12}$$

The results of applying Algorithm 3.1 to this problem are presented in Table 5.4.

Example 5.
Here we consider the transformation $F^T : \mathbf{C}^N \to \mathbf{C}^N$ of Problem 3.2, defined by the formula

$$F^T(\alpha)_j = \sum_{k=1}^{N} \alpha_k \cdot e^{ijx_k} \tag{5.13}$$

for $j = -N/2, \ldots, N/2 - 1$. In this example, $\{x_1, \ldots, x_N\}$ were randomly distributed on the interval $[-\pi, \pi]$ and $\{\alpha_1, \ldots, \alpha_N\}$ were complex numbers randomly chosen from the unit square

$$0 \leq \mathrm{Re}(z) \leq 1,\ 0 \leq \mathrm{Im}(z) \leq 1. \tag{5.14}$$

The results of applying Algorithm 3.2 to this problem are presented in Table 5.5.

Example 6.
Here we consider the transformation $F^{-1} : \mathbf{C}^N \to \mathbf{C}^N$ of Problem 3.3 where F is

defined by the formula

$$F(\alpha)_j = \sum_{k=-N/2}^{N/2-1} \alpha_k \cdot e^{ikx_j} \tag{5.15}$$

for $j = 1, \ldots, N$. In this example, $\{x_1, \ldots, x_N\}$ were defined by the formulae

$$x_j = -\pi + 2\pi \cdot \frac{j + 0.5 + \delta_j}{N} \tag{5.16}$$

for $j = 1, \ldots, N$, where δ_j were randomly distributed on the interval $[-0.1, 0.1]$. In addition, $\{\alpha_{-N/2}, \ldots, \alpha_{N/2-1}\}$ were complex numbers randomly chosen from the unit square

$$0 \leq \mathrm{Re}(z) \leq 1,\ 0 \leq \mathrm{Im}(z) \leq 1, \tag{5.17}$$

and the numbers $\{f_1, \ldots, f_N\}$ were computed directly in double precision arithmetic according to the formula $f_j = F(\alpha)_j$. The vector f was then used as input for Algorithm 3.3. Results of this experiment are presented in Table 5.6.

Example 7.
Here we consider the transformation $(F^T)^{-1} : \mathbf{C}^N \to \mathbf{C}^N$ of Problem 3.4 where F^T is defined by the formula

$$F^T(\alpha)_j = \sum_{k=1}^{N} \alpha_k \cdot e^{ijx_k} \tag{5.18}$$

for $j = -N/2, \ldots, N/2-1$. In this example, $\{x_1, \ldots, x_N\}$ were defined by the formulae

$$x_j = -\pi + 2\pi \cdot \frac{j + 0.5 + \delta_j}{N} \tag{5.19}$$

for $j = 1, \ldots, N$, where δ_j were randomly distributed on the interval $[-0.1, 0.1]$. In addition, $\{\alpha_1, \ldots, \alpha_N\}$ were complex numbers randomly chosen from the unit square

$$0 \leq \mathrm{Re}(z) \leq 1,\ 0 \leq \mathrm{Im}(z) \leq 1, \tag{5.20}$$

and the numbers $\{f_{-N/2}, \ldots, f_{N/2-1}\}$ were computed directly in double precision arithmetic according to the formula $f_j = F^T(\alpha)_j$. The vector f was then used as input for Algorithm 3.4. Results of this experiment are presented in Table 5.7.

The following observations can be made from Tables 5.1–5.7, and are in agreement with results of our more extensive experiments for this particular architecture, implementation and range of N.

1. All of the algorithms permit high accuracy to be attained, and the observed errors are in accordance with the theoretically obtained error bounds.

2. As expected, the CPU timings for all the algorithms grow slightly faster than linearly with the problem size, N.

3. The timings for Algorithms 4.1 and 4.2 are similar, which is to be expected since Problem 4.2 is the adjoint of Problem 4.1. Algorithm 4.3 is about twice as costly, which is in agreement with the fact that it is a synthesis of Algorithms 4.1 and 4.2.

4. The timings for Algorithms 3.1–3.4 are similar, which is to be expected since these four algorithms are so closely related.

5. The initialization times for Algorithms 3.1 and 3.2 are considerably less than those for Algorithms 3.3 and 3.4. This is because the former pair does not require the additional computation of the numbers $\{c_k\}$ and $\{d_k\}$.

6. Algorithms 4.1 and 4.2 are about 5 times as costly as an FFT of the same size, and for Algorithms 3.1–3.4, this ratio is about 15.

7. Break-even points for Algorithms 4.1 and 4.2 with the direct method are at $N = 128$ if the initialization time is included, and at $N = 16$ if it is ignored. Algorithm 4.3 becomes faster than the direct calculation at $N = 32$, even when the initialization time is included.

8. Algorithms 3.1 and 3.2 can compete with the direct method at about $N = 32$ ignoring initialization time, and at $N = 1024$ including the initialization. Algorithms 3.3 and 3.4 are always dramatically faster than the direct calculation (15000 times faster at $N = 2048$) if we ignore initialization time, and break even with it at $N = 64$ if we include the initialization.

9. The initialization stage is much more costly than the evaluation stage for all of the algorithms. Implementing the algorithms in two stages thus gives considerable time savings whenever the same linear transformation is to be applied to multiple vectors.

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	0.08	0.002
256	0.437 E-14	0.946 E-14	0.148	0.034	0.31	0.005
512	0.519 E-14	0.160 E-13	0.297	0.075	1.20	0.012
1024	0.518 E-14	0.314 E-13	0.600	0.155	4.76	0.026
2048	0.755 E-14	0.631 E-13	1.204	0.322	18.93	0.059

Table 5.1: Example 1, Numerical Results for Algorithm 4.1.

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	0.02	0.001
128	0.501 E-14	0.746 E-14	0.075	0.012	0.09	0.002
256	0.418 E-14	0.623 E-14	0.148	0.028	0.33	0.005
512	0.356 E-14	0.831 E-14	0.297	0.060	1.24	0.012
1024	0.793 E-14	0.192 E-13	0.596	0.126	4.93	0.026
2048	0.138 E-13	0.405 E-13	1.188	0.264	19.62	0.059

Table 5.2: Example 2, Numerical Results for Algorithm 4.2.

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

Table 5.3: Example 3, Numerical Results for Algorithm 4.3.

N	Errors		Timings (sec.)			
	E_∞	E_2	Alg. Init.	Alg. Eval.	Direct	FFT
128	0.379 E-14	0.704 E-14	1.04	0.030	0.09	0.002
256	0.398 E-14	0.116 E-13	2.03	0.081	0.33	0.005
512	0.499 E-14	0.195 E-13	3.26	0.171	1.24	0.012
1024	0.318 E-13	0.625 E-13	4.97	0.408	4.93	0.026
2048	0.763 E-13	0.204 E-12	8.07	0.822	19.62	0.059

Table 5.4: Example 4, Numerical Results for Algorithm 3.1.

N	Errors		Timings (sec.)			
	E_∞	E_2	Alg. Init.	Alg. Eval.	Direct	FFT
128	0.206 E-14	0.800 E-14	1.03	0.033	0.08	0.002
256	0.323 E-14	0.136 E-13	2.05	0.081	0.31	0.005
512	0.153 E-13	0.343 E-13	3.21	0.174	1.20	0.012
1024	0.180 E-13	0.654 E-13	5.11	0.409	4.76	0.026
2048	0.470 E-13	0.221 E-12	8.16	0.823	18.93	0.059

Table 5.5: Example 5, Numerical Results for Algorithm 3.2.

N	Errors		Timings (sec.)			
	E_∞	E_2	Alg. Init.	Alg. Eval.	Direct	FFT
128	0.117 E-13	0.800 E-14	1.28	0.034	2.96	0.002
256	0.196 E-13	0.137 E-13	2.51	0.082	23.6	0.005
512	0.344 E-13	0.230 E-13	4.33	0.175	189	0.012
1024	0.107 E-12	0.757 E-13	7.45	0.409	1512 (est.)	0.026
2048	0.357 E-12	0.247 E-12	12.97	0.819	12096 (est.)	0.059

Table 5.6: Example 6, Numerical Results for Algorithm 3.3.

N	Errors		Timings (sec.)			
	E_∞	E_2	Alg. Init.	Alg. Eval.	Direct	FFT
128	0.134 E-13	0.806 E-14	1.26	0.033	2.96	0.002
256	0.511 E-13	0.179 E-13	2.47	0.080	23.6	0.005
512	0.870 E-13	0.373 E-13	4.24	0.173	189	0.012
1024	0.178 E-12	0.811 E-13	7.29	0.407	1512 (est.)	0.026
2048	0.942 E-12	0.369 E-12	12.80	0.820	12096 (est.)	0.059

Table 5.7: Example 7, Numerical Results for Algorithm 3.4.

Chapter 6

Conclusions and Generalizations

The principal results of this thesis are two groups of efficient algorithms for computing FFTs for nonequispaced data to any specified precision. Similarities and differences between the two approaches are summarized below.

1. Both sets of algorithms use a standard FFT.

2. Both sets of algorithms use interpolation formulae to transform function values from equispaced to nonequispaced points and vice-versa. Algorithms 3.1–3.4 use an interpolation scheme based on the FMM, while Algorithms 4.1–4.3 use an interpolation scheme based on the Fourier analysis of Gaussian bells.

3. For the application of the linear tranformations being considered, the algorithms of Chapter 4 are more efficient than the algorithms of Chapter 3.

4. For the inversion of these linear transformations, the schemes of Chapter 4 can be applied iteratively, however the direct approach of Chapter 3 is generally more efficient.

5. Algorithms 3.1–3.4 comprise a set of closely related forward and inverse algorithms which can be generalized to complex data points.

In conclusion, two groups of algorithms have 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 a broad range of applications in many branches of mathematics, science and engineering. Several related algorithms have also been presented which utilize the analytical tools of this thesis. These include:

1. an efficient version of the Fast Multipole Method in one dimension,

2. a fast algorithm for polynomial interpolation on the real line,

3. fast algorithms for spectral integration and differentiation of functions tabulated at nodes other than Chebyshev, and,

4. FFT for complex data points.

The results of this thesis are currently being applied to problems in a diversity of areas. Examples include problems in the numerical solution of parabolic partial differential equations, the analysis of seismic data, the modelling of semiconductors, weather prediction and the numerical simulation of fluid behavior.

Several straightforward generalizations of the results of this thesis are discussed below.

1. Problems 3.1, 3.2, 4.1, 4.2 and 4.3 all involve the evaluation of an N-term series at N points. Minor modifications to Algorithms 3.1, 3.2, 4.1, 4.2 and 4.3 will allow the efficient evaluation of these N-term series at M points, where $M \neq N$. These changes have been implemented.

2. The algorithms of this thesis also assume that $\omega_k \in [-N/2, N/2]$ and $x_j \in [-\pi, \pi]$. Other distributions of ω and x can be handled by appropriately partitioning these vectors, 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 4.1–4.3, or with one of Algorithms 3.1–3.2.

Observation 6.1 *Let $a, b > 0, c, d > 0$ be a set of 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*

$$\begin{aligned} \sum_{k=0}^{N} \alpha_k \cdot e^{i\omega_k x_j} &= e^{iax_j} \cdot \sum_{k=0}^{N} \alpha_k \cdot e^{i(\omega_k - a)c} \cdot e^{i(\omega_k - a)(x_j - c)} \\ &= e^{iax_j} \cdot \sum_{k=0}^{N} \alpha'_k \cdot e^{i\omega'_k x'_j}, \end{aligned} \tag{6.1}$$

where

$$\begin{aligned} \alpha'_k &= \alpha_k \cdot e^{i(\omega_k - a)c}, \\ \omega'_k &= (\omega_k - a)d/\pi, \\ x'_j &= (x_j - c)\pi/d \in [-\pi, \pi]. \end{aligned} \tag{6.2}$$

Remark 6.2 An algorithm of this type 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.

3. The algorithms of this thesis 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], [4]), and the use of multipole or Chebyshev expansions for the compression of certain classes of linear operators (see, for example, [2], [7], [22]).

4. One of the more far-reaching extensions of the results of this thesis is a set of algorithms for discrete Fourier transforms in several dimensions. In two dimensions, for example, we may wish to evaluate the sums

$$f_{m,n} = \sum_{k=1}^{N} \alpha_k \cdot e^{imx_k + iny_k} \tag{6.3}$$

for each integer pair (m, n), where the points (x_k, y_k) are generally distributed in the plane. A straightforward application of the techniques of this thesis yields an order

$$N \cdot \log N + N \cdot \log^2 \left(\frac{1}{\varepsilon}\right) \tag{6.4}$$

algorithm for this problem. Special-purpose algorithms can also be designed for increased efficiency in cases when the points (x_k, y_k) lie on a curve. Detailed investigations into higher dimensional problems of this type are currently in progress and will be reported at a later date.

Bibliography

[1] B. ALPERT, G. BEYLKIN, R. COIFMAN AND V. ROKHLIN, *Wavelets for the Fast Solution of Second-Kind Integral Equations*, SIAM J. Sci. Stat. Comp., 14 (1993).

[2] B. ALPERT AND V. ROKHLIN, *A Fast Algorithm for the Evaluation of Legendre Expansions*, Technical Report 671, Yale Computer Science Department, 1988.

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

[4] G. BEYLKIN, R. COIFMAN AND V. ROKHLIN, *Fast Wavelet Transforms and Numerical Algorithms I,* Comm. on Pure and Applied Mathematics, 44 (1991), pp. 141-183.

[5] E. ORAN BRIGHAM, *The Fast Fourier Transform and its Applications*, Prentice Hall Inc., Englewood Cliffs, N.J., 1988.

[6] R. A. BROOKS AND G. DI CHIRO, *Principles of Computer Assisted Tomography (CAT) in Radiographic and Radioisotopic Imaging*, Phys. Med. Biol., 21 (1976), pp. 689-732.

[7] J. CARRIER, L. GREENGARD AND V. ROKHLIN, *A Fast Adaptive Multipole Algorithm for Particle Simulations*, SIAM J. Sci. Stat. Comp., 9 (1988), pp. 669–686.

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

[9] G. DAHLQUIST AND A. BJORCK, *Numerical Methods*, Prentice Hall Inc., Englewood Cliffs, N.J., 1974.

[10] A. DUTT AND V. ROKHLIN, *On the Rapid Evaluation of Trigonometric Series*, Technical Report 893, Yale Computer Science Dept, 1991.

[11] A. DUTT AND V. ROKHLIN, *Fast Fourier Transforms for Nonequispaced Data*, SIAM J. Sci. Stat. Comp., to appear, 1993.

[12] D. GOTTLIEB, M. Y. HUSSAINI AND S. ORSZAG, in *Spectral Methods for Partial Differential Equations*, edited by R. G. Voigt, D. Gottlieb and M. Y. Hussaini, SIAM, Philadelphia PA, 1984, p.1.

[13] D. GOTTLIEB AND S. ORSZAG, *Numerical Analysis of Spectral Methods*, SIAM, Philadelphia PA, 1977.

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

[15] L. GREENGARD, *Spectral Integration and Two-Point Boundary Value Problems*, Technical Report 646, Yale Computer Science Department, 1988.

[16] L. GREENGARD AND V. ROKHLIN, *A Fast Algorithm for Particle Simulations*, J. Comp. Phys., 73 (1987), pp. 325–348.

[17] M. GU AND S. C. EISENSTAT, *A Divide-And-Conquer Algorithm for the Symmetric Tridiagonal Eigenproblem*, Technical Report 932, Yale Computer Science Department, 1992.

[18] M. GU AND S. C. EISENSTAT, *A Divide-And-Conquer Algorithm for the Bidiagonal SVD*, Technical Report 933, Yale Computer Science Department, 1992.

[19] M. GU AND V. ROKHLIN, personal communication.

[20] E. L. HALL, *Computer Image Processing and Recognition*, Academic Press, New York, 1979.

[21] G. T. HERMAN, *Image Reconstruction from Projections: Implementation and Applications*, Springer Verlag, New York, 1979.

[22] V. ROKHLIN, *A Fast Algorithm for the Discrete Laplace Transformation*, Journal of Complexity, 4 (1988), pp. 12-32.

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

[24] C. VAN LOAN, *Computational Frameworks for the Fast Fourier Transform*, SIAM, Philadelphia, 1992.

[25] H. JOSEPH WEAVER, *Theory of Discrete and Continuous Fourier Analysis*, Wiley, 1989.