Fast Multipole-Type Methods in One and Two Dimensions, with Application to Parallel Fourier Transforms

by

Peter William McCorquodale

B.Sc. (Hons.), Simon Fraser University (1990)

M.S., Stanford University (1992)

Submitted to the Department of Mathematics
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 1998

© Peter William McCorquodale, MCMXCVIII. All rights reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part, and to grant others the right to do so.

Author ....
Department of Mathematics
January 16, 1998

Certified by .....
Alan Edelman
Assistant Professor
Thesis Supervisor

Accepted by ....
Hung Cheng
Chairman, Applied Mathematics Committee

Accepted by .....
Richard Melrose
Chairman, Departmental Committee on Graduate Students

FEB 18 1998 ARCHIVES

Fast Multipole-Type Methods in One and Two Dimensions, with Application to Parallel Fourier Transforms

by

Peter William McCorquodale

Submitted to the Department of Mathematics
on January 16, 1998, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis has three parts. In the first part, a new, accurate version of the fast multipole method in one dimension is presented, and error estimates for it are derived for the $1/x$ potential on the line, and for the $\cot(x/2)$ potential on the circle. The second part introduces two new approximate algorithms for the discrete Fourier transform. Each one can save up to a factor of three in communication over the conventional parallel FFT algorithm, at the cost of more arithmetic. The first method is based on singular value decompositions and matrix multiplication. The second, more efficient method is based on the algorithm from the first part of this thesis, and our implementation of it has been found to run faster than a conventional parallel FFT on some computer architectures. The third part of this thesis compares three fast multipole-type methods for the logarithmic potential in the complex plane: the original Greengard–Rokhlin algorithm, Anderson's algorithm, and a new algorithm based on complex polynomial interpolation. All three methods are based on approximation by polynomials, and we show by experiment that they also have nearly identical performance, as measured by accuracy versus computational complexity, except for numerical instabilities with the polynomial interpolation method when very high accuracy is needed.

Thesis Supervisor: Alan Edelman
Title: Assistant Professor

Acknowledgments

I would first like to thank my advisor, Professor Alan Edelman, for making this thesis possible. I am deeply grateful for his constant encouragement and enthusiasm, personal attention, and sharing of his perspectives on numerical analysis, parallel computing, and effective presentation. I feel privileged to be his first doctoral student, and the first to graduate.

Thanks also to Professors Jacob White and Gilbert Strang for serving as readers on my thesis committee.

For generous financial support during all of my time at MIT, I am grateful for teaching assistantships and a fellowship from the mathematics department, and research assistantships funded by National Science Foundation grants 9501278-DMS and 9404326-CCR to Professor Edelman. I would also like to thank Mercury Computer Systems for showing interest in my work and sponsoring me in implementing an early version of an algorithm in Chapter 3 during the summer of 1995.

Sivan Toledo provided the initial insight of Chapter 3, and later ran tests of my MPI code on an SP2 parallel computer at IBM Watson Research Center. I am also grateful for the use of the clusters of SMPs being developed at MIT in collaboration with Sun Microsystems, Inc. and Digital Equipment Corp.

I thank Dr. Alok Dutt and Professor Vladimir Rokhlin for helpful discussions on the algorithms in Chapter 2.

I am thankful for both technical help and companionship from colleagues in the math department and the Laboratory for Computer Science, particularly the numerics group of Yanyuan Ma, Parry Husbands and Ross Lippert. Finally, thanks to my family for their devotion and support during these years.

Contents

1 Introduction **14**

1.1 Fast Multipole-type methods 14

1.2 Description of FMM 17

1.3 Computational models 21

1.4 Contributions of this thesis 22

2 Evaluating One-Dimensional Potentials **24**

2.1 Introduction 24

2.2 Mathematical preliminaries 25

2.3 Algorithms on the line: LIN and variants 28

2.3.1 Different approaches 28

2.3.2 Description 29

2.3.3 Complexity 35

2.3.4 Parallel complexity 36

2.3.5 Compressing with the SVD: algorithm $LINSVD$ 39

2.3.6 One-sided algorithms 41

2.4 Algorithms on the circle: $CIRC$ and variants 49

2.4.1 Different approaches 49

2.4.2 Description 50

2.4.3 Complexity 56

2.4.4 Parallel complexity 57

2.4.5 Compressing with the SVD: $CIRCSVD$ 58

2.5 Error in LIN for $1/x$ potential 59

2.5.1 Approximation error in expansions 60
2.5.2 Overall error estimates for a single charge 64
2.5.3 General error estimate for LIN 70
2.6 Error in $CIRC$ for $\cot(\theta/2)$ potential 73
2.6.1 Approximation error in expansions 73
2.6.2 Overall error estimates for a single charge 79
2.6.3 General error estimate for $CIRC$ 80
2.7 Numerical experiments 82
2.8 Conclusions 85

3 Low-Communication Parallel Fourier Transforms 88
3.1 Parallel Fourier Transforms 88
3.2 Mathematical insights 90
3.3 $MVDFT$: A Matrix-Vector Algorithm 95
3.3.1 Accuracy 97
3.3.2 Complexity 98
3.4 $FMDFT$: Fast Multipole Approach 99
3.4.1 Matrix factorizations 99
3.4.2 General Approach 101
3.4.3 Accuracy 103
3.4.4 Complexity 104
3.5 Experimental Results 106
3.5.1 Performance Results 106
3.5.2 Extrapolation to Other Machines 109
3.6 Conclusions 111

4 Comparison of Two-Dimensional Methods 118
4.1 Approaches to the problem 118
4.2 Structure of fast multipole-type methods in two dimensions 119
4.3 Complexity of $POT2D_EVAL$ 124
4.4 Greengard–Rokhlin method 126

4.5 Anderson's method 129
4.6 Polynomial interpolation method 132
4.7 Coefficients from Taylor series 137
4.8 Comparison of algorithms 143
4.9 Conclusions 145

List of Figures

1-1 Boxes on two successive levels of a two-dimensional square domain. Each box has four children at the next level. The parent $p(b)$ of the box labelled b at level 3 is indicated at level 2. An "o" marks the adjacent boxes in the sets $J(b)$ at level 3 and $J(p(b))$ at level 2. An "×" marks boxes in the interaction lists, $I(b)$ at level 3 and $I(p(b))$ at level 2. At each level, the FMM computes the potentials due to charges in the interaction list of all boxes at that level. 18

1-2 The matrix V for a one-dimensional problem with $h = 5$ and hence 32 boxes. Rows correspond to potential evaluation points, and columns correspond to charge points. The directly evaluated component $V_{near-field}$ consists of the submatrices marked with "×". The rest of the matrix is $V_{far-field}$, with different shadings corresponding to different levels (2, 3, 4, 5) at which the particular interaction is computed. 19

1-3 Architecture of a distributed-memory machine. 22

2-1 Transformations used in approximating the potential at points marked with "o" due to charges at points marked with "×". At top, charges are in the inner interval and are well separated from the evaluation points, which are mapped to $[-1, 1]$ using X_O. At bottom, charges are outside the inner interval and are well separated from the evaluation points, which are mapped to $[-1, 1]$ using X_I. 31

2-2 Transformations used in approximating the potential at points marked with "o" due to charges at points marked with "×". At top, charges are in the narrower interval and are well separated from the evaluation points, which are mapped to $[-1, 1]$ using X_O. At bottom, charges are in the wider interval and are well separated from the evaluation points, which are mapped to $[-1, 1]$ using X_I. 51

2-3 Value of the error $1/(y+1) - LIN(10, 8, -1, y, 1)$. Between each pair of dashed vertical lines, the horizontal axis is scaled linearly with a different scaling. The maximum absolute difference between the exact error plotted here and Estimate 2.5.8 is 5×10^{-10}, which is too small to be distinguishable in their graphs. Also note that Estimate 2.5.9 gives a bound of 1.5×10^{-7} on the absolute error. This bound is within a factor of 2 of the observed maximum error. 72

2-4 Value of the error $\cot((y - 0.66)/2) - CIRC(10, 8, 0.66, y, 1)$. Between each pair of dashed vertical lines, the horizontal axis is scaled linearly with a different scaling. The maximum absolute difference between the exact error plotted here and Estimate 2.5.8 is 1.5×10^{-10}, which is too small to be distinguishable in the graphs. Also note that Estimate 2.6.11 gives a bound of 2×10^{-7} on the absolute error. This bound is within a factor of 4 of the observed maximum error. 81

2-5 Plot of relative error vs. number of flops of the LIN family of algorithms, for a random distribution of 1024 charges and 1024 points with $1/x$ potential. Each marker labelled with a number p is for an algorithm using expansions of length p with no compression. It is connected by thin black lines to smaller markers of the same type, for algorithms with compression. Notice the convergence to an envelope. 83

2-6 Relative error in LIN and $LINSVD$ from Figure 2-5, together with the error bound from Estimate 2.5.11 84

2-7 Plot of relative error vs. number of flops of the $CIRC$ family of algorithms, for a random distribution of 1024 charges and 1024 points with $\cot(\theta/2)$ potential. Each marker labelled with a number p is for an algorithm using expansions of length p with no compression. It is connected by thin black lines to smaller markers of the same type, for algorithms with compression. Notice the convergence to an envelope. 86

2-8 Relative error in $CIRC$ and $CIRCSVD$ from Figure 2-7, together with the error bound from Estimate 2.6.13. 87

3-1 Communication pattern in parallel FFT of length 32 over 4 processors, using the six-step framework based on the factorization $F_{32} = (F_4 \otimes I_8)T(I_4 \otimes F_8)\Pi$ of equation (3.8) in Section 3.4. The step numbers are indicated at the bottom of the figure. 89

3-2 Singular values of $F_{1024|4}$ 92

3-3 Singular values of a 256×256 section of a random 1024×1024 unitary matrix, computed with MATLAB. 92

3-4 Normalized times on the SP2, in microseconds per point. The times for $PFFT$ are indicated with "□". The times for $FMDFT$ are indicated with "○". These graphs contain the same data as Table 3.3. 113

3-5 Normalized times on the SP2 (microseconds per point) with the High-Performance Switch without operating system overhead (US-HPS). The times for $PFFT$ are indicated with "□". The times for $FMDFT$ are indicated with "○". In each graph, the bottom component is the communication time, the top component is for the local FFT of size n/p, and the middle component is the time for all other arithmetic. These graphs contain the same data as Table 3.4. 115

3-6 Normalized times on the Sun cluster, in microseconds per point. The times for $PFFT$ are indicated with "□". The times for $FMDFT$ are indicated with "○". For each algorithm in each graph, the bottom component is the communication time, the top component is for the local FFT of size n/p, and the middle component is the time for all other arithmetic. These graphs contain the same data as Table 3.5. . 117

4-1 Points in a box at level l have maximum distance $w_l\sqrt{2}$ from the center. The closest point in the interaction list of a box has distance $3w_l$ from the center. 121

4-2 An illustration of the polynomial interpolation method, with $p = 10$. Double lines indicate boundaries of boxes at the next higher level. The coefficients of the far-field expansion for the charges in the center box, at level l, are the potentials evaluated at the $p-1$ circled points on the large circle of radius $2w_lR_O$, where here $R_O = 2\sqrt{2}$. The box containing the charges has 27 other boxes in its interaction list. The local expansions at these boxes are found by interpolating at the $p-1$ asterisked points on the smaller circles of radius $2w_lR_I$, where here $R_I = 1/2$. 135

4-3 The three types of particle distributions. Each leaf-level box contains 40 particles. 144

4-4 Plot of normalized ℓ^2 error against flop count for $POT2D_EVAL$ with four hierarchy levels and 10,240 charges. There are 40 random charges in each of the 256 leaf-level boxes. These graphs contain the same information as Tables 4.2, 4.3, and 4.4. 149

4-5 Plot of normalized ℓ^2 error against flop count for $POT2D_EVAL$ with four hierarchy levels and 10,240 charges. Each graph is for a separate method. There are 40 random charges in each of the 256 leaf-level boxes. 150

4-6 Plot of normalized ℓ^2 error against flop count for $POT2D_EVAL$ with five hierarchy levels and 40,960 charges. There are 40 random charges in each of the 1024 leaf-level boxes. These graphs contain the same information as Tables 4.5, 4.6, and 4.7. 154

4-7 Plot of normalized ℓ^2 error against flop count for $POT2D_EVAL$ with five hierarchy levels and 40,960 charges. Each graph is for a separate method. There are 40 random charges in each of the 1024 leaf-level boxes. 155

List of Tables

3.1 A listing of eigenvalues in a transition interval for the matrices $G_{n|p}$ and W_p, of order n/p, and the approximation (3.5). 95

3.2 Number of significant singular values required for given relative accuracy. 98

3.3 A comparison of the performance of $PFFT$ and $FMDFT$ on an SP2 parallel computer using three communication mechanisms. Running times are in seconds. The three communication mechanisms that were used are user-space communication over the High-Performance Switch (US-HPS), internet protocol over the High-Performance Switch (IP-HPS), and internet protocol over Ethernet (IP-EN). The last two rows give the minimum and maximum ratios of the timings reported in the table to what one would expect from the sum of Equations (3.16)–(3.18) for T_C, or Equations (3.19)–(3.21) for T_N. 112

3.4 A comparison of the performance of $PFFT$ and $FMDFT$ on an SP2 parallel computer. The communication software used the High-Performance Switch without operating system overhead (US-HPS). Mean times are reported in seconds. The total time is divided into three parts: T_{fftloc} spent in Netlib local FFTs, T_{comm} used for communication, and T_{arith} for other arithmetic. The last two rows give the minimum and maximum ratios of the timings reported in the table to what one would expect from Equations (3.16)–(3.21). 114

3.5 A comparison of the performance of the two algorithms on a cluster of servers of UltraSPARC processors. Mean times are reported in seconds. The total time is divided into three parts: T_{fftloc} spent in Netlib local FFTs, T_{comm} used for communication, and T_{arith} for other arithmetic. The last two rows give the minimum and maximum ratios of the timings reported in the table to what one would expect from Equations (3.16)–(3.21). 116

4.1 Analytic functions used for each approximation method. 137

4.2 Normalized ℓ^2 error in the Greengard-Rokhlin algorithm with 10K random charges and four hierarchy levels. 146

4.3 Normalized ℓ^2 error in Anderson's algorithm with 10K random charges and four hierarchy levels. 147

4.4 Normalized ℓ^2 error in polynomial interpolation algorithm with 10K random charges and four hierarchy levels. 148

4.5 Normalized ℓ^2 error in Greengard–Rokhlin algorithm with 40K random charges and five hierarchy levels. 151

4.6 Normalized ℓ^2 error in Anderson's algorithm with 40K random charges and five hierarchy levels. 152

4.7 Normalized ℓ^2 error in polynomial interpolation algorithm with 40K random charges and five hierarchy levels. 153

Chapter 1

Introduction

1.1 Fast Multipole-type methods

In many applications in science, engineering and mathematics, there arise numerical problems that can be expressed as matrix-vector multiplications of the form

$$\mathbf{f} = V(\mathbf{y}, \mathbf{x})\mathbf{q}, \tag{1.1}$$

where the finite matrix $V(\mathbf{y}, \mathbf{x})$ has elements

$$V_{ij} = \rho(y_i - x_j)$$

for some function ρ which has a singularity at zero. We may interpret $\mathbf{f}$ as electrostatic potentials at positions $\mathbf{y}$, due to a collection of point charges at positions $\mathbf{x}$ and with charge strengths $\mathbf{q}$. The matrix element V_{ij} gives the potential at position y_i due to a unit charge at position x_j.

For $\mathbf{x}$ and $\mathbf{y}$ each of length N, the direct method of evaluating $\mathbf{f}$ in Equation (1.1) requires an amount of work proportional to N^2. In 1987, Greengard and Rokhlin [14] introduced the Fast Multipole Method (FMM) to solve problems of this type using

only $O(N)$ work. The FMM is based on partitioning the matrix V into two parts,

$$V = V_{near-field} + V_{far-field}.$$

The component $V_{near-field}$ corresponds to interactions between nearby particles, and is evaluated directly. The component $V_{far-field}$ corresponds to interactions between more distant particles, and is evaluated using approximate formulas on an hierarchical set of subdomains.

The approximation schemes used in the FMM depend on the domain and on the potential ρ in the particular problem. Generally, two different types of formulas are used to approximate the far-field component of the potential:

1. far-field expansions, representing the potential due to charges in a particular region D, valid at points well outside D;

2. local expansions, valid in a region D, representing the potential due to charges well outside D.

In their original paper, Greengard and Rokhlin [14] used truncated Taylor series for local expansions, and truncated multipole series for far-field expansions.

Following is a partial list of previously published papers on FMM algorithms, including the main sources for those referenced in this thesis. A more extensive list can be found in a 1997 article by Greengard and Rokhlin [16].

Domain	Potential	Interpolation functions	Reference
$\mathbb{R}$	$1/x$	truncated multipole series, Taylor series	Gu [20]
$\mathbb{R}$	— " —	Chebyshev interpolation polynomials	Dutt, Gu, Rokhlin [10]
S^1	$\cot(\theta/2)$	Chebyshev interpolation polynomials	Dutt, Rokhlin [11]
$\mathbb{R}$	$1/x$, $\log\lvert x\rvert$, $1/\sqrt{x}$	singular functions of integral operators, discretized integrals of exponentials	Yarvin, Rokhlin [34]
$\mathbb{C}$	$\log\lvert z\rvert$	truncated multipole series, Taylor series	Greengard, Rokhlin [14]
$\mathbb{C}$	— " —	discretized Poisson's formula	Anderson [3]
$\mathbb{C}$	$1/z$	singular functions of integral operators, discretized integrals of exponentials	Hrycak, Rokhlin [23]
$\mathbb{R}^3$	$1/\lVert\mathrm{x}\rVert$	truncated multipole series, Taylor series	Greengard [17]
$\mathbb{R}^3$	— " —	discretized Poisson's formula	Anderson [3]
$\mathbb{R}^3$	— " —	singular functions of integral operators, discretized integrals of exponentials	Greengard, Rokhlin [16]

In this thesis, we present new interpolation-based algorithms for problems on the line and on the circle. We derive overall error estimates for these algorithms with the $1/x$ potential on the line and the $\cot(\theta/2)$ potential on the circle. We also introduce a new algorithm based on polynomial interpolation for the logarithmic potential in the complex plane, and compare its performance with the Greengard–Rokhlin [14] and Anderson [3] methods.

1.2 Description of FMM

In the FMM, the domain containing all the charge positions $\mathbf{x}$ and potential evaluation positions $\mathbf{y}$ is subdivided into "boxes" at different levels of refinement. The entire original domain is considered as a single box at level 0. Each box at level l is subdivided into some fixed number of boxes, called its *children*, at level $l+1$. The box at level l containing the children is called the *parent* of the boxes at level $l+1$. For a d-dimensional domain, each box has 2^d children. We choose some maximum number h of levels to be used. Then in d dimensions, there are 2^{dh} boxes at the finest level of refinement.

For each box b at every level, we define:

- $p(b)$ is the parent of box b.
- $C(b)$ is the set of children of b.
- $J(b)$ is the set of boxes that are *adjacent* to box b, in the sense of having boundaries that intersect with those of b. These are also called the *neighbors* of b.
- $I(b)$ is the *interaction list* of box b; these boxes are not adjacent to b, but their parents are adjacent to the parent of b. $I(b) = \{b' : p(b') \in J(p(b)) \ \& \ b' \notin J(b)\}$.

Examples of these sets are illustrated in Figure 1-1 for levels 2 and 3 of a non-periodic square domain in two dimensions.

The purpose of these hierarchical domains is that at level l, the FMM finds an approximation for the potentials in each box b, due to charges in $I(b)$. Doing this recursively and adding up over a suitably chosen maximum number of levels, the algorithm performs $O(N)$ work for N charges and evaluation points.

Figure 1-2 illustrates which individual interactions are computed at each level when $h = 5$ on a non-periodic linear domain. At each successive level above level 5, roughly twice as many individual interactions are computed as at the previous level.

For each b at the lowest level, h, we use the notation

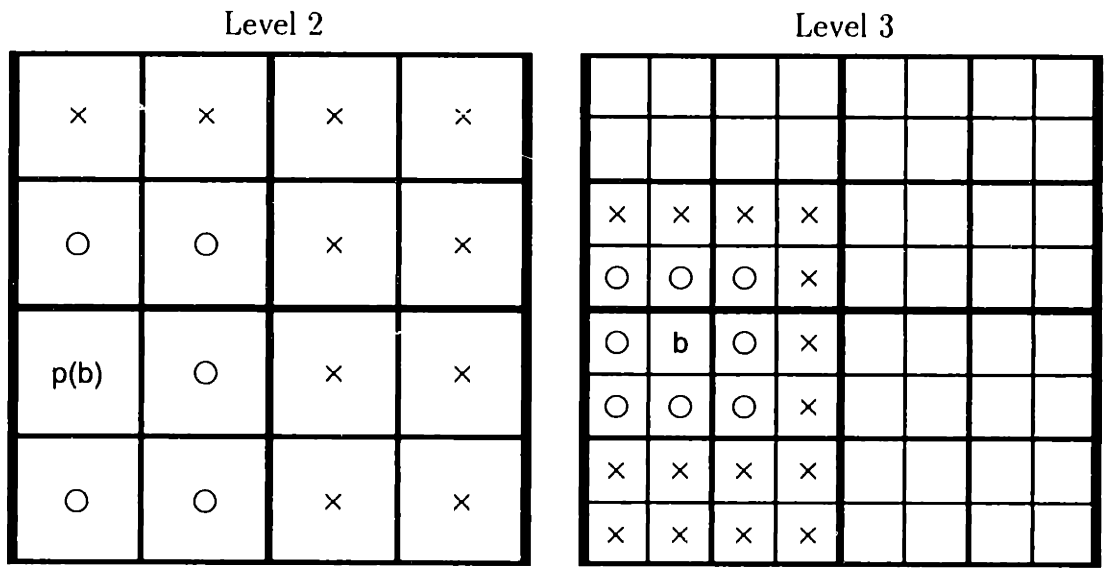


Figure 1-1: Boxes on two successive levels of a two-dimensional square domain. Each box has four children at the next level. The parent $p(b)$ of the box labelled b at level 3 is indicated at level 2. An "o" marks the adjacent boxes in the sets $J(b)$ at level 3 and $J(p(b))$ at level 2. An "×" marks boxes in the interaction lists, $I(b)$ at level 3 and $I(p(b))$ at level 2. At each level, the FMM computes the potentials due to charges in the interaction list of all boxes at that level.

- $\mathcal{X}(b) = \{j : x_j \in B_b\}$ is the set of indices of charges in box B_b.
- $\mathcal{Y}(b) = \{i : y_i \in B_b\}$ is the set of indices of evaluation positions in box B_b.

The function $INIT(h, \mathbf{x}, \mathbf{y})$ sets up operators that depend on the charge positions $\mathbf{x}$ and evaluation positions $\mathbf{y}$.

For each box b at level h, we use the following linear operators.

- G_b, which depends on $\mathbf{x}_{\mathcal{X}(b)}$, is an operator for obtaining far-field expansion coefficients from charges in box b at level h.
- E_b, which depends on $\mathbf{y}_{\mathcal{Y}(b)}$, is an operator for obtaining potentials at points in box b at level h from local expansion coefficients.
- D_b, which depends on $\mathbf{y}_{\mathcal{Y}(b)}$ and $\mathbf{x}_{\mathcal{X}(b \cup J(b))}$, is an operator for obtaining potentials at points in box b at level h due to charges in box b and its neighbors.

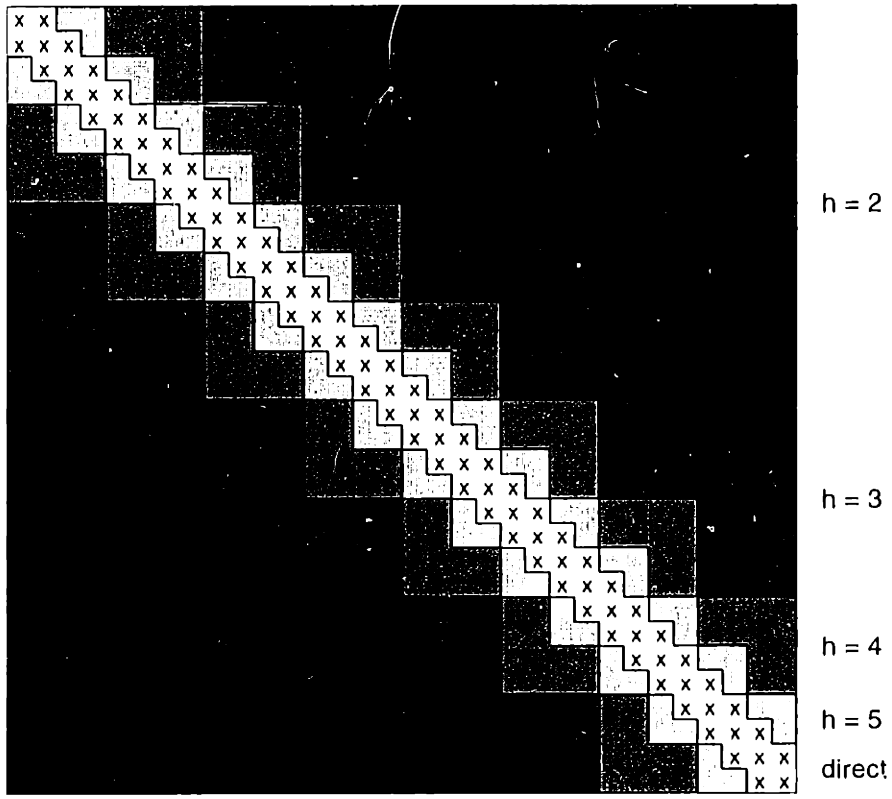


Figure 1-2: The matrix V for a one-dimensional problem with $h = 5$ and hence 32 boxes. Rows correspond to potential evaluation points, and columns correspond to charge points. The directly evaluated component $V_{near-field}$ consists of the submatrices marked with "×". The rest of the matrix is $V_{far-field}$, with different shadings corresponding to different levels (2, 3, 4, 5) at which the particular interaction is computed.

After initializing operators with $INIT$, the FMM algorithm calls the function $EVAL$ with the charge strengths $\mathbf{q}$. $EVAL$ may be called again with different choices of $\mathbf{q}$ without having to rerun $INIT$.

In the description of the FMM algorithm, two representations for potential are used:

- $\Phi_b^{(l)}$ is an expression for the potential due to the charges in box b at level l, valid outside b and its neighbors;
- $\Psi_b^{(l)}$ is an expression for the potential within box b due to the charges outside b and its neighbors.

Algorithm 1.2.1

$\mathbf{f} = FMM(h, \mathbf{x}, \mathbf{y}, \mathbf{q})$ evaluates $f_j = \sum_k \rho(y_j - x_k) \cdot q_k$, using h hierarchy levels.

function $INIT(h, \mathbf{x}, \mathbf{y})$:

foreach box b at level h: Initialize G_b, E_b, and D_b.

end function $INIT$.

function $\mathbf{f} = EVAL(\mathbf{q})$:

1. **comment** [Get coefficients of far-field expansions.]
   foreach box b at level h, find $\Phi_b^{(h)} = G_b(q_{\mathcal{X}(b)})$.

2. **comment** [Get coefficients of local expansions.]
   $(\Psi^{(h)}) = RECUR(h, \Phi^{(h)})$

3. **comment** [Evaluate potentials from local expansions and direct interactions.]
   foreach box b at level h,
   $$f_{\mathcal{Y}(b)} = E_b(\Psi_b^{(h)}) + D_b(q_{\mathcal{X}(b \cup J(b))})$$
   end

end function $EVAL$.

function $(\Psi^{(l)}) = RECUR(l, \Phi^{(l)})$:

if $l = 0$ **then**

$\Psi^{(0)} = 0$.

else

1. **comment** [Compute coefficients of far-field expansions at level $l - 1$.]
   foreach box b at level $l - 1$,
   $$\tilde{\Phi}_b^{(l-1)} = \sum_{c \in C(b)} \Phi_c^{(l)}$$
   end

2. **comment** [Get coefficients of local expansions at level $l-1$.]
   $(\tilde{\Psi}^{(l-1)}) = RECUR(l-1, \tilde{\Phi}^{(l-1)})$

3. **comment** [Compute coefficients of local expansions at level l.]
   foreach box b at level l,
   $$\Psi_b^{(l)} = \Psi_{p(b)}^{(l-1)} + \sum_{i \in I(b)} \Phi_i^{(l)}$$
   end

end

end function $RECUR$.

The fast multipole-type algorithms described in this thesis all follow the pattern of Algorithm 1.2.1. We restrict the analysis of run-time performance of algorithms to the $EVAL$ portion only, ignoring the computations of $INIT$. This corresponds to having fixed $\mathbf{x}$ and $\mathbf{y}$.

1.3 Computational models

In this thesis, we shall measure the arithmetic complexity of algorithms by counting *flops*, or *fl*oating-point *op*erations (see, for example, Golub and Van Loan [13]). A flop is one addition or subtraction or multiplication of real floating-point numbers. For complex data, an addition involves two flops, and a multiplication involves six flops. Our algorithms shall consist primarily of matrix-matrix and matrix-vector multiplications. If A is $m \times k$, B is $k \times n$, and C is $m \times n$, then the number of flops involved in computing $AB + C$ by the direct method is:

- $2mkn$, if A, B, and C are all real;
- $4mkn$, if one of A or B is real, and the other is complex, as is C;
- $8mkn$, if A, B, and C are all complex.

Flop counts provide the most convenient estimate of relative execution times of different algorithms on a sequential machine, although the actual execution time may also depend on memory traffic patterns and other overheads.

In Chapters 2 and 3, we shall describe algorithms that use a parallel distributed-memory machine model (see, for example, Hennessy and Patterson [22]). In this model, as illustrated in Figure 1-3, there are a number of independent processing nodes, each with its own memory and its own input and output. The nodes are connected by a network. This architecture describes networks of workstations, as well as current machines such as the IBM SP2, and clusters of multiprocessors such as the Sun Ultra Enterprise server cluster of MIT, in which each individual node has a small number of processors.

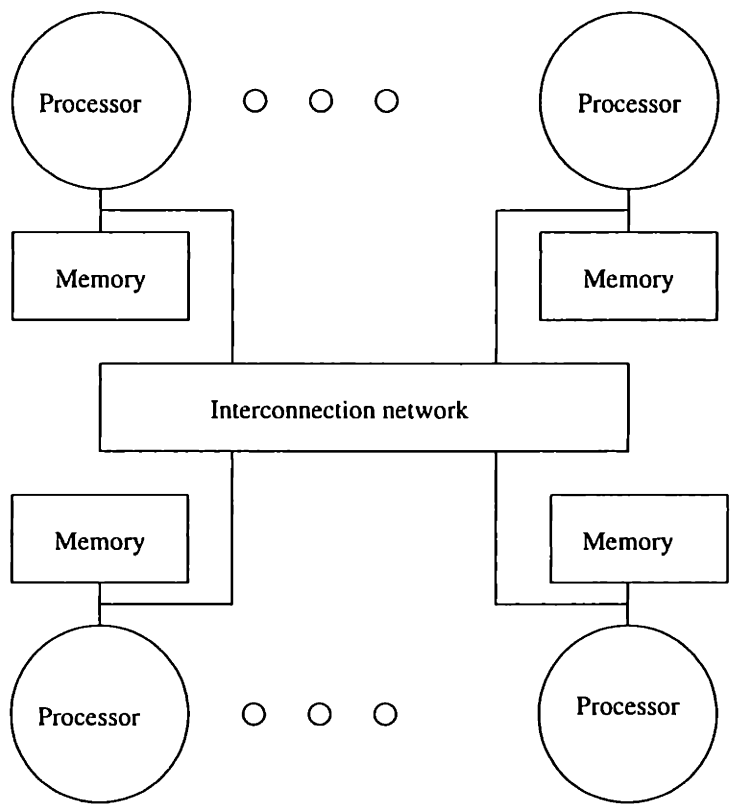


Figure 1-3: Architecture of a distributed-memory machine.

1.4 Contributions of this thesis

Chapter 2 of this thesis describes efficient fast multipole-type methods for evaluating one-dimensional potentials on the line or on the circle. These algorithms improve

upon the methods of Dutt, Gu Rokhlin [10], and Dutt and Rokhlin [11], with a new set of interpolation functions, and we derive error estimates for them. We find that after applying compression techniques to the space of interpolating functions, the complexity-accuracy tradeoffs of the algorithms are the same. However, no analytical error estimates are known for the compressed algorithms. and the new uncompressed algorithms have an accuracy within an order of magnitude of the compressed algorithms.

Chapter 3 describes how an approximate algorithm can reduce communication in parallel computation of the discrete Fourier transform (DFT) by up to a factor of three. Two new parallel DFT algorithms are presented: one based on matrix-vector multiplication, and a more efficient one based on a one-dimensional fast multipole algorithm from Chapter 2. We present results of tests of the FMM-based algorithm on two computer platforms, and show that for some architectures it outperforms conventional parallel algorithms for this problem.

In Chapter 4, we move to the two-dimensional domain with a logarithmic potential, the original problem studied by Greengard and Rokhlin [14]. We compare three fast multipole-type algorithms that use different approximating functions: the original Greengard–Rokhlin algorithm, a method by Anderson [3] based on Poisson's formula, and a new method based on interpolation by complex polynomials. We show the mathematical relations among the three methods, and report the results of numerical experiments. These experiments show that up to a certain accuracy, the behavior of the three algorithms is virtually identical, as measured by work required for a given accuracy. For high accuracy, the Greengard–Rokhlin and Anderson methods are found to behave equally well, and polynomial interpolation is either less accurate or unstable, depending on the location of the interpolation points.

Chapter 2

Evaluating One-Dimensional Potentials

2.1 Introduction

This chapter discusses efficient methods of computing the one-dimensional version of (1.1), which is

$$f_j = \sum_{k=1}^{K} \rho(y_j - x_k) \cdot q_k, \tag{2.1}$$

where $q_1, \ldots, q_K$ are real numbers, and ρ has a singularity at 0 but is smooth elsewhere. We consider two one-dimensional domains:

- on the line: $x_1, \ldots, x_K, y_1, \ldots, y_N \in [-1, 1]$;
- on the circle: $x_1, \ldots, x_K, y_1, \ldots, y_N \in [0, 2\pi]$, and ρ is 2π-periodic.

The first case is representative of any finite interval.

Important examples of potentials are $\rho(x) = 1/x$ on the line, and $\rho(x) = \cot(\theta/2)$ on the circle. For these two potentials, we derive error estimates for our methods.

We use an interpolation-based approach as proposed by Dutt, Gu and Rokhlin [10], using different interpolation functions to provide greater accuracy. These algorithms can be made more efficient using a singular value decompositions to compress the basis of interpolation functions, as outlined by Dutt et al. [10] and discussed in greater

detail by Yarvin and Rokhlin [34]. However, error bounds other than empirical ones have not been obtainable for the compressed versions.

The remainder of this chapter is organized as follows. Section 2.2 has mathematical preliminaries introducing our interpolation functions. Section 2.3 describes an algorithm, LIN, and some variants of it for problem (2.1) on the line. Section 2.4 describes an algorithm, $CIRC$, and some variants of it for problem (2.1) on the circle. Section 2.5 gives error estimates for LIN with potential $\rho(x) = 1/x$, and Section 2.6 gives error estimates for $CIRC$ with potential $\rho(\theta) = \cot(\theta/2)$. Section 2.7 reports results of numerical experiments of LIN with $1/x$ potential and $CIRC$ with $\cot(\theta/2)$ potential. We find that when the same form of compression of function spaces is used, all of the algorithms can be made to perform equally well, but without compression, our algorithms LIN and $CIRC$ come closest to optimal. Section 2.8 discusses our conclusions.

2.2 Mathematical preliminaries

Definition 2.2.1 *The pth-degree Chebyshev polynomial, $T_p(x)$, is defined by the formulae*

$$T_p(x) = \cos(p \arccos x) \tag{2.2}$$

$$= \frac{1}{2}((x + \sqrt{x^2 - 1})^p + (x - \sqrt{x^2 - 1})^p). \tag{2.3}$$

Definition 2.2.2 *The Chebyshev nodes $t_1, \ldots, t_p$ of order p on the interval $[-1, 1]$ are*

$$t_k = -\cos((k - \frac{1}{2})\frac{\pi}{p}) \tag{2.4}$$

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

Lemma 2.2.3 *The zeroes of T_p are $t_1, \ldots, t_p$.*

Proof. This is clear from Equations (2.2) and (2.4). □

Lemma 2.2.4 *If $x \in [-1, 1]$ then $|T_p(x)| \leq 1$.*

Proof. Obvious from Equation (2.2). □

Lemma 2.2.5 *If* $|x| \geq 3$ *then* $|T_p(x)| > \frac{1}{2}((1+\frac{\sqrt{8}}{3})|x|)^p$.

Proof. Suppose $|x| \geq 3$. Then from Equation (2.3),

$$\begin{aligned} |T_p(x)| &= \frac{1}{2}|(x+\sqrt{x^2-1})^p + (x-\sqrt{x^2-1})^p| \\ &> \frac{1}{2}|x+\sqrt{x^2-1}|^p = \frac{1}{2}((1+\sqrt{1-\frac{1}{x^2}})|x|)^p \geq \frac{1}{2}((1+\frac{\sqrt{8}}{3})|x|)^p, \end{aligned}$$

since $1/x^2 \leq 1/9$. □

We define u_k to be the Lagrange interpolating polynomial for the function that is 1 at the kth Chebyshev node and 0 at the other Chebyshev nodes: $u_k(t_l) = \delta_{kl}$.

Definition 2.2.6 *Let* $u_1, \ldots, u_p$ *denote polynomials of degree* $p-1$ *defined by*

$$u_k(x) = \prod_{\substack{j=1 \\ j \neq k}}^{p} \frac{x - t_j}{t_k - t_j},$$

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

The Chebyshev approximation of a function f defined on $[-1,1]$ is the polynomial

$$\widetilde{f}(x) = \sum_{j=1}^{p} f(t_j) u_j(x), \tag{2.5}$$

which is the Lagrange interpolating polynomial for f with nodes $t_1, \ldots, t_p$. This is the unique polynomial of degree $p-1$ that agrees with f at all of the Chebyshev nodes t_j. It is well known [25] that the Chebyshev nodes are a practical set of nodes for interpolating an unknown function with polynomials on $[-1,1]$.

Using appropriate transformations at different scales, Dutt, Gu and Rokhlin [10] apply the Fast Multipole Method to reduce the problem of Equation (2.1) to that of approximating the potential on $[-1,1]$ due to charges in $(-\infty,-3] \cup [3,+\infty)$, with Chebyshev approximations.

We propose the following alternative: interpolate a potential function on $[-1,1]$ which has poles in $(-\infty,-3]\cup[3,+\infty)$, by picking some $\gamma_1,\ldots,\gamma_{p-1}$ with $|\gamma_j|\geq 3$ as positions of "virtual charges" (with the $1/x$ potential), and taking a sum

$$c+\sum_{j=1}^{p-1}\frac{a_j}{x-\gamma_j}=\frac{P(x)}{\prod_{j=1}^{p-1}(x-\gamma_j)},$$

where P is a polynomial of degree $p-1$ with coefficients that depend on c, the a_j and γ_j. The constant term c on the left-hand side is necessary for approximating the potential on a narrow interval due to charges outside the interval. We discovered numerically that for most accurate results, $\gamma_j=3/t'_j$ where t'_j is the jth Chebyshev node of order $p-1$.

Definition 2.2.7 *Let $v_1,\ldots,v_p$ denote rational functions defined by*

$$v_k(x)=u_k(x)\prod_{i=1}^{p-1}\frac{t_k-\gamma_i}{x-\gamma_i},$$

where u_k is as in Definition 2.2.6, and $\gamma_i=3/t'_i$, where t'_i is the ith Chebyshev node of order $p-1$: $t'_i=-\cos((i-\frac{1}{2})\frac{\pi}{p-1})$.

Then in analogy with Expression (2.5), we may approximate a function f on $[-1,1]$ by the expression

$$\widetilde{f}(x)=\sum_{j=1}^{p}f(t_j)v_j(x). \tag{2.6}$$

The following lemma will be useful in proving properties of approximation using the v_k.

Lemma 2.2.8 *For any $x,y\neq 0$,*

$$\prod_{i=1}^{p-1}\frac{x-\gamma_i}{y-\gamma_i}=(\frac{x}{y})^{p-1}\frac{T_{p-1}(3/x)}{T_{p-1}(3/y)}.$$

Proof. Substituting $\gamma_i = 3/t'_i$:

$$\prod_{i=1}^{p-1} \frac{x-\gamma_i}{y-\gamma_i} = \prod_{i=1}^{p-1} \frac{x-3/t'_i}{y-3/t'_i} = \prod_{i=1}^{p-1} \frac{(t'_i-3/x)(x/t'_i)}{(t'_i-3/y)(y/t'_i)} = (\frac{x}{y})^{p-1} \prod_{i=1}^{p-1} \frac{3/x-t'_i}{3/y-t'_i},$$

and then the result follows from Lemma 2.2.3. □

2.3 Algorithms on the line: LIN and variants

2.3.1 Different approaches

This section describes different algorithms for evaluating (2.1) on the line. There are three alternative choices leading to eight different algorithms:

- Interpolation functions can be either the polynomials u_k from Definition 2.2.6, or the rational functions v_k from Definition 2.2.7 representing potentials due to "virtual charges".

- Methods can be "one-sided" or "two-sided". A one-sided approach computes, for each box b, separate far-field expansions valid to the left or to the right of b, and separate local expansions due to charges to the left or to the right of b. A two-sided approach computes one far-field expansion valid outside b, on both sides, and one local expansion due to charges on both sides of b.

- Singular value decompositions can be used or not used to compress function spaces, as described initially in Section 2.3.5.

The following table summarizes the different algorithms on the line that are used in this chapter.

Algorithm	interpolation functions	SVD compression	1-sided or 2-sided	symbol in Fig. 2-5
LIN	v_k	no	2	○
$LINSVD$	v_k	yes	2	○
$LIN1$	v_k	no	1	□
$LIN1SVD$	v_k	yes	1	□
Alg. 3.1 of [10]	u_k	no	2	△
not in [10]	u_k	yes	2	△
Alg. 3.2 of [10] with $\bar{p} = p$	u_k	no	1	▽
Alg. 3.2 of [10]	u_k	yes	1	▽

We find that:

- two-sided algorithms are more accurate than one-sided algorithms;
- interpolating with the virtual-charge functions v_k provides more accuracy than with the polynomials u_k;
- with function spaces compressed by the SVD, the performance of all algorithms becomes the same.

2.3.2 Description

Our first algorithm is $LIN(p, h, \mathbf{x}, \mathbf{y}, \mathbf{q})$, where we have:

- p is the length of expansions to be used in the algorithm;
- h is the maximum number of levels to be used in the algorithm;
- $\mathbf{x}$ is a vector containing charge positions $x_1, \ldots, x_K \in [-1, 1]$;
- $\mathbf{y}$ is a vector containing evaluation points $y_1, \ldots, y_N \in [-1, 1]$;
- $\mathbf{q}$ is a vector of length K with its kth component q_k being the strength of the charge at point x_k.

The idea is to interpolate on $[-1,1]$ by mapping the potential evaluation points to $[-1,1]$.

We use two types of expansions for intervals $[c-w, c+w]$:

- Far-field expansions, where the charges are located in the interval, $|y-c| \leq w$, and the evaluation points are at y with $|y-c| \geq 3w$. If

$$x = X_O(y) = \frac{3w}{y-c}$$

  then X_O maps evaluation points to $|x| \leq 1$ and charge positions to $|x| \geq 3$. The inverse function of X_O is

$$y = Y_O(x) = \frac{3w}{x} + c.$$

- Local expansions, where the evaluation points are located in the interval, $|y-c| \leq w$, and charges are located at points y with $|y-c| \geq 3w$. If

$$x = X_I(y) = \frac{y-c}{w}$$

  then X_I maps evaluation points to $|x| \leq 1$ and charge positions to $|x| \geq 3$. The inverse function of X_I is

$$y = Y_I(x) = wx + c.$$

These mappings are illustrated in Figure 2-1.

The algorithm also uses the following:

- for each level $l = 1, \ldots, h$ we define:
  - $w_l = 1/2^l$ is the half-width of subintervals at level l;
  - $X_I^{(l)}$ and $Y_I^{(l)}$ are functions used to transform between the interval $[-1,1]$ and $[-w_l, w_l]$:

$$X_I^{(l)}(y) = \frac{y}{w_l}, \qquad Y_I^{(l)}(x) = w_l x. \tag{2.7}$$

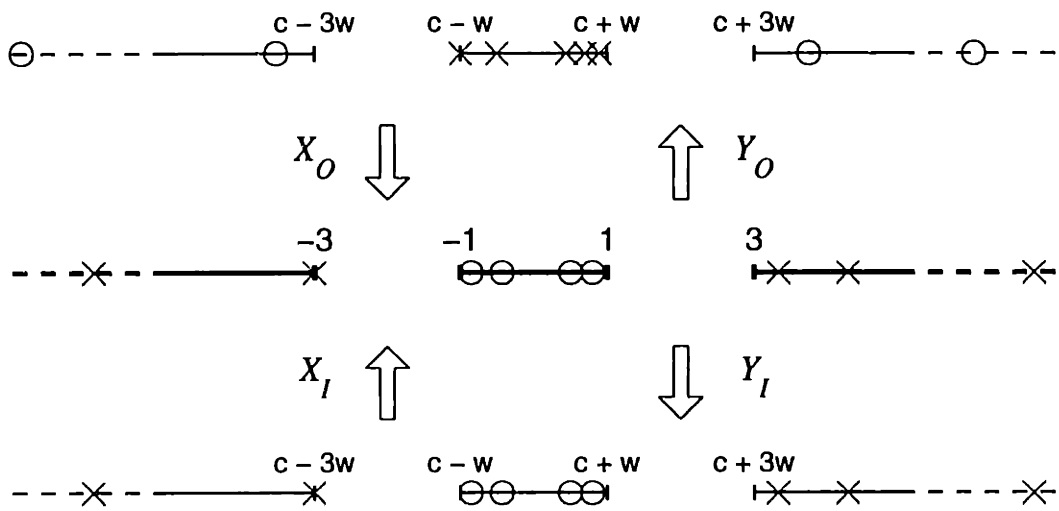


Figure 2-1: Transformations used in approximating the potential at points marked with "o" due to charges at points marked with "×". At top, charges are in the inner interval and are well separated from the evaluation points, which are mapped to $[-1, 1]$ using X_O. At bottom, charges are outside the inner interval and are well separated from the evaluation points, which are mapped to $[-1, 1]$ using X_I.

- ◇ $X_O^{(l)}$ and $Y_O^{(l)}$ are functions used to transform between the interval $[-1, 1]$ and $(-\infty, -3w_l] \cup [3w_l, +\infty)$:

$$X_O^{(l)}(y) = \frac{3w_l}{y}, \qquad Y_O^{(l)}(x) = \frac{3w_l}{x}. \tag{2.8}$$

- $s = 2^h$ is the number of intervals at level h;
- for $b = 1, \ldots, s$, we define the following parameters for the intervals at level h:
  - ◇ $I_b = [c_b - w_h, c_b + w_h]$ is the bth subinterval at level h;
  - ◇ $c_b = (2b - 1)w_h - 1$ is the center of I_b;
  - ◇ $\mathcal{X}_b = \{k : x_k \in I_b\}$ is the set of indices of charge positions in I_b.
  - ◇ $\mathcal{Y}_b = \{j : y_j \in I_b\}$ is the set of indices of evaluation points in I_b.
- for $l = 2, \ldots, h$ and $b = 1, \ldots, 2^l$, we define:

- ◇ $\Phi_b^{(l)}$ is a p-vector containing the coefficients of the far-field expansion for the bth subinterval at level l.
- ◇ $\Psi_b^{(l)}$ is a p-vector containing the coefficients of the local expansion for the bth subinterval at level l.

If $b \notin \{1, \ldots, 2^l\}$ then $\Phi_b^{(l)}$ and $\Psi_b^{(l)}$ are taken to be zero.

To convert between expansions at different levels, the algorithm uses the following operators which are fixed by p:

- M_L and M_R are $p \times p$ matrices for obtaining far-field expansions for subintervals from the far-field expansions for their children:

$$M_L(j,k) = v_k(X_O^{(l+1)}(Y_O^{(l)}(t_j) + w_{l+1})) = v_k(\frac{3t_j}{6+t_j})$$
$$M_R(j,k) = v_k(X_O^{(l+1)}(Y_O^{(l)}(t_j) - w_{l+1})) = v_k(\frac{3t_j}{6-t_j})$$

- S_L and S_R are $p \times p$ matrices for obtaining local expansions for subintervals from local expansions for their parent:

$$S_L(j,k) = v_k(X_I^{(l-1)}(Y_I^{(l)}(t_j) - w_l)) = v_k(\frac{t_j - 1}{2})$$
$$S_R(j,k) = v_k(X_I^{(l-1)}(Y_I^{(l)}(t_j) + w_l)) = v_k(\frac{t_j + 1}{2})$$

- T_1, T_2, T_3, and T_4 are $p \times p$ matrices for obtaining local expansions from far-field expansions:

$$T_1(j,k) = v_k(X_O^{(l)}(Y_I^{(l)}(t_j) + 6w_l)) = v_k(\frac{3}{t_j+6})$$
$$T_2(j,k) = v_k(X_O^{(l)}(Y_I^{(l)}(t_j) + 4w_l)) = v_k(\frac{3}{t_j+4})$$
$$T_3(j,k) = v_k(X_O^{(l)}(Y_I^{(l)}(t_j) - 4w_l)) = v_k(\frac{3}{t_j-4})$$
$$T_4(j,k) = v_k(X_O^{(l)}(Y_I^{(l)}(t_j) - 6w_l)) = v_k(\frac{3}{t_j-6})$$

The function $LIN_INIT(p, h, \mathbf{x}, \mathbf{y})$ sets up operators that depend on $\mathbf{x}$ and $\mathbf{y}$. For each $b = 1, \ldots, s$:

- G_b is a $p \times |\mathcal{X}_b|$ matrix for obtaining far-field expansion coefficients from charges in I_b:

$$G_b(j,k) = \rho(Y_O^{(h)}(t_j) - (x_k - c_b)) = \rho(\frac{t_j}{3w - t_j(x_k - c_b)}). \tag{2.9}$$

  G_b uses only the columns with indices in $\mathcal{X}_b$.

- E_b is a $|\mathcal{Y}_b| \times p$ matrix for obtaining potentials at points in I_b from local expansion coefficients:

$$E_b(j,k) = v_k(X_I^{(h)}(y_j - c_b)) = v_k((y_j - c_b)s). \tag{2.10}$$

  E_b uses only the rows with indices in $\mathcal{Y}_b$.

- D_b is a $|\mathcal{Y}_b| \times |\mathcal{X}_{b-1} \cup \mathcal{X}_b \cup \mathcal{X}_{b+1}|$ matrix for obtaining potentials at points in I_b due to charges in I_b and its neighbors:

$$D_b(j,k) = \rho(y_j - x_k). \tag{2.11}$$

  D_b uses only the rows with indices in $\mathcal{Y}_b$ and the columns with indices in $\mathcal{X}_{b-1} \cup \mathcal{X}_b \cup \mathcal{X}_{b+1}$.

Algorithm LIN_INIT initializes these matrices.

Algorithm 2.3.1

$\mathbf{f} = LIN(p, h, \mathbf{x}, \mathbf{y}, \mathbf{q})$ evaluates $f_j = \sum_k \rho(y_j - x_k) \cdot q_k$ for $x_k, y_j \in [-1, 1]$, using expansions with p terms and with h hierarchy levels.

function $LIN_INIT(p, h, \mathbf{x}, \mathbf{y})$:

for $b = 1, \ldots, 2^h$: Compute G_b, E_b, and D_b using Equations (2.9)–(2.11).

end function LIN_INIT.

function $\mathbf{f} = LIN_EVAL(\mathbf{q})$:

1. **comment** [Get coefficients of far-field expansions.]

   $(\Phi_1^{(h)}, \ldots, \Phi_s^{(h)}) = (G_1 \mathbf{q}_{\mathcal{X}_1}, \ldots, G_s \mathbf{q}_{\mathcal{X}_s})$

2. **comment** [Get coefficients of local expansions.]

   $(\Psi_1^{(h)}, \ldots, \Psi_s^{(h)}) = LIN_RECUR(h, \Phi_1^{(h)}, \ldots, \Phi_s^{(h)})$

3. **comment** [Evaluate potentials from local expansions and direct interactions.]

   for $b = 1 : s$,

   $f_{\mathcal{Y}_b} = E_b \Psi_b^{(h)} + D_b \mathbf{q}_{\mathcal{X}_{b-1} \cup \mathcal{X}_b \cup \mathcal{X}_{b+1}}$

   end

end function LIN_EVAL.

function $(\Psi_1^{(l)}, \ldots, \Psi_{2^l}^{(l)}) = LIN_RECUR(l, \Phi_1^{(l)}, \ldots, \Phi_{2^l}^{(l)})$:

if $l = 2$ **then**

$(\Psi_1^{(2)}, \Psi_2^{(2)}, \Psi_3^{(2)}, \Psi_4^{(2)}) = (T_3\Phi_3^{(2)} + T_4\Phi_4^{(2)}, T_3\Phi_4^{(2)}, T_2\Phi_1^{(2)}, T_1\Phi_1^{(2)} + T_2\Phi_2^{(2)})$

else

1. **comment** [Compute coefficients of far-field expansions at level $l-1$.]

   for $b = 1, \ldots, 2^{l-1}$:

   $\widetilde{\Phi}_b^{(l-1)} = M_L \Phi_{2b-1}^{(l)} + M_R \Phi_{2b}^{(l)}$

   end

2. **comment** [Get coefficients of local expansions at level $l-1$.]

   $(\widetilde{\Psi}_1^{(l-1)}, \ldots, \widetilde{\Psi}_{2^{l-1}}^{(l-1)}) = LIN_RECUR(l-1, \widetilde{\Phi}_1^{(l-1)}, \ldots, \widetilde{\Phi}_{2^{l-1}}^{(l-1)})$

3. **comment** [Compute coefficients of local expansions at level l.]

   for $b = 1, \ldots, 2^{l-1}$:

   $\Psi_{2b-1}^{(l)} = S_L \widetilde{\Psi}_b^{(l-1)} + T_2\Phi_{2b-3}^{(l)} + T_3\Phi_{2b+1}^{(l)} + T_4\Phi_{2b+2}^{(l)}$

   $\Psi_{2b}^{(l)} = S_R \widetilde{\Psi}_b^{(l-1)} + T_1\Phi_{2b-3}^{(l)} + T_2\Phi_{2b-2}^{(l)} + T_3\Phi_{2b+2}^{(l)}$

   end

end

end function LIN_RECUR.

2.3.3 Complexity

Assuming that $\mathbf{q}$ is real, the flop counts for each step of $LIN_RECUR(l, \Phi_1^{(l)}, \ldots, \Phi_{2^l}^{(l)})$ are:

- If $l = 2$ then the flop count is $6 \cdot 2p^2 = 12p^2$.
- Otherwise:
  1. flop count is $2^{l-1} \cdot 2 \cdot 2p^2 = 2^l \cdot 2p^2$;
  2. flop count for $LIN_RECUR(l-1, \Phi_1^{(l-1)}, \ldots, \Phi_{2^{l-1}}^{(l-1)})$;
  3. flop count is $(2^{l-1} \cdot 8 - 6) \cdot 2p^2 = 2^l \cdot 8p^2 - 12p^2$.

The total is $10p^2 \cdot 2^l - 12p^2$ plus the flop count for $LIN_RECUR(l-1, \Phi_1^{(l-1)}, \ldots, \Phi_{2^{l-1}}^{(l-1)})$, which totals $(20 \cdot 2^l - 12l - 44)p^2$.

Flop counts for LIN_EVAL:

1. $2pK$;
2. $(20 \cdot 2^h - 12h - 44)p^2$;
3. $2pN + 2\sum_{b=1}^{s} |\mathcal{Y}_b|(|\mathcal{X}_{b-1}| + |\mathcal{X}_b| + |\mathcal{X}_{b+1}|)$.

The total for LIN_EVAL is then

$$2p(K+N) + (20s - 12h - 44)p^2 + 2\sum_{b=1}^{s} |\mathcal{Y}_b|(|\mathcal{X}_{b-1}| + |\mathcal{X}_b| + |\mathcal{X}_{b+1}|). \tag{2.12}$$

If all s intervals at level h contain equal numbers of charges, and also equal numbers of evaluation points, then $|\mathcal{X}_b| = K/s$ and $|\mathcal{Y}_b| = N/s$, and expression (2.12) gives

$$2p(K+N) + (20s - 12\lg s - 44)p^2 + \frac{6KN}{s},$$

which is a minimum when

$$s = s_{\text{opt}} = \frac{3}{10\ln 2} + \frac{1}{10}\sqrt{(\frac{3}{\ln 2})^2 + \frac{30KN}{p^2}} \approx \frac{1}{p}\sqrt{\frac{3KN}{10}}. \tag{2.13}$$

Since s must be a power of 2, we may pick $s_{\text{opt}}/\sqrt{2} \leq s \leq s_{\text{opt}}\sqrt{2}$. Then the number of flops is at most

$$2p(3\sqrt{15KN} + K + N) - 12p^2 \lg(\frac{\sqrt{5KN}}{p}). \tag{2.14}$$

2.3.4 Parallel complexity

Assume a distributed-memory model as described in Section 1.3, with $P \geq 2$ processors, where P divides s, the number of intervals. Processors are labelled $0, 1, \ldots, P-1$, and each one contains data for s/P intervals. Specifically, processor μ contains $\mathbf{x}_{\mathcal{X}_b}$, $\mathbf{q}_{\mathcal{X}_b}$, and $\mathbf{y}_{\mathcal{Y}_b}$ for $b = (s/P)\mu + 1, \ldots, (s/P)\mu + s/P$, and is to contain $\mathbf{f}_{\mathcal{Y}_b}$ at the conclusion of the algorithm.

We count the parallel arithmetic complexity as being the maximum number of floating-point operations performed by any processor. The communication complexity is the maximum number of scalars sent out by any one processor to other processors.

In the parallel version of LIN, the far-field and local expansion coefficients are stored across processors. For $l \geq \lg P$, processor μ contains $\Phi_b^{(l)}$, $\Psi_b^{(l)}$, $\widetilde{\Phi}_b^{(l)}$, and $\widetilde{\Psi}_b^{(l)}$ for $b = (2^l/P)\mu + 1, \ldots, (2^l/P)\mu + 2^l/P$. For $2 \leq l \leq \lg P - 1$, the coefficients $\Phi_b^{(l)}$, $\Psi_b^{(l)}$, $\widetilde{\Phi}_b^{(l)}$, and $\widetilde{\Psi}_b^{(l)}$ are stored in processor $(b-1)P/2^l$.

For $LIN_RECUR(l, \Phi_1^{(l)}, \ldots, \Phi_{2^l}^{(l)})$, the communication patterns depend on whether $l \leq \lg P$.

If $l \geq \lg P + 1$ (and so $P \leq 2^{l-1}$):

- If $l = 2$ then the constraints $l \geq \lg P + 1$ and $P \geq 2$ imply $P = 2$. Each processor performs $3 \cdot 2p^2 = 6p^2$ flops and sends $2p$ coefficients in $\Phi^{(2)}$ to the other processor.

- Otherwise:

  1. Each processor performs $(2^{l-1}/P) \cdot 2 \cdot 2p^2 = 2^l \cdot 2p^2/P$ flops, and there is no communication.

  2. Complexity of $LIN_RECUR(l-1, \Phi_1^{(l-1)}, \ldots, \Phi_{2^{l-1}}^{(l-1)})$;

3. Each processor performs at most $(2^{l-1}/P) \cdot 8 \cdot 2p^2 = 2^l \cdot 8p^2/P$ flops, and sends $2p$ coefficients in $\Phi^{(l)}$ to each of the two adjacent processors.

If $l \leq \lg P$ (and so $P \geq 2^l$):

- If $l = 2$ then $P \geq 4$ in this case. Each of four processors performs at most $2 \cdot 2p^2 = 4p^2$ flops after having sent p coefficients in $\Phi^{(2)}$ to each of one or two other processors.

- Otherwise:

  1. Each of 2^l processors performs $2p^2$ flops, and each of 2^{l-1} sends p coefficients $M_L \Phi^{(l)}_{2b-1}$ to the processor containing $\Phi^{(l)}_2 b$ and $\widetilde{\Phi}^{(l-1)}_b$.
  2. Complexity of $LIN_RECUR(l-1, \Phi^{(l-1)}_1, \ldots, \Phi^{(l-1)}_{2^{l-1}})$;
  3. Each of 2^l processors performs at most $4 \cdot 2p^2 = 8p^2$ flops after sending p coefficients in $\Phi^{(l)}$ to each of three other processors. Also, each of 2^{l-1} processors sends p coefficients $\Psi^{(l-1)}_b$ to the processor containing $\Psi^{(l)}_{2b-1}$.

The maximum total flop count of $LIN_RECUR(h, \Phi^{(h)}_1, \ldots, \Phi^{(h)}_s)$ for a processor is then at most

$$4p^2 + \sum_{l=3}^{\lg P} 10p^2 + \sum_{l=\lg P+1}^{h} (\frac{2^l}{P}) 10p^2 \leq p^2(\frac{20s}{P} + 10 \lg P - 36)$$

and the maximum number of scalars sent by any processor is at most

$$2p + \sum_{l=3}^{\lg P} 5p + \sum_{l=\lg P+1}^{h} 4p \leq p(4h + \lg P - 7).$$

If $l \geq \lg P + 1$, all of the scalars communicated in LIN_RECUR are from far-field expansions, Φ. The communication can be arranged so that they are all sent at the same time, before the local expansions Ψ are computed. Then processor μ need only send one separate message each to processors $\mu - 1$ and $\mu + 1$ for $l \geq \lg P + 1$.

For $l \leq \lg P$, at Step 1 of LIN_RECUR, at most one message containing far-field coefficients in Φ needs to be sent by each processor over all $l \leq \lg P$. At step

3, at most one message containing local coefficients in Ψ needs to be sent for each $l \leq \lg P$, for a total of $\lg P - 2$. In addition, there are three messages containing far-field coefficients in Φ for each level l, giving a total of $3 \lg P - 2$ messages.

In *LIN_RECUR*, then, the communication can be arranged so that at most $4 \lg P - 3$ messages are sent by each processor.

For *LIN_EVAL*:

1. Each processor μ performs $2p \sum_{b=(s/P)\mu+1}^{(s/P)(\mu+1)} |\mathcal{X}_b|$ flops and sends no data.

2. Each processor performs at most $p^2(\frac{20s}{P} + 10 \lg P - 36)$ flops and sends at most $p(4h + \lg P - 7)$ scalars in at most $4 \lg P - 3$ messages.

3. Each processor μ performs $2p \sum_{b=(s/P)\mu+1}^{(s/P)(\mu+1)} |\mathcal{Y}_b| + 2 \sum_{b=(s/P)\mu+1}^{(s/P)(\mu+1)} |\mathcal{Y}_b|(|\mathcal{X}_{b-1}| + |\mathcal{X}_b| + |\mathcal{X}_{b+1}|)$ flops, and sends $2|\mathcal{X}_{(s/P)\mu+1}|$ scalars from $\mathbf{q}$ and $\mathbf{x}$ to $\mu - 1$, and $2|\mathcal{X}_{(s/P)(\mu+1)}|$ scalars from $\mathbf{q}$ and $\mathbf{x}$ to $\mu + 1$.

The parallel arithmetic complexity of *LIN_EVAL* is then

$$p^2(\frac{20s}{P} + 10 \lg P - 36) \tag{2.15}$$
$$+ \max_{\mu=0,\ldots,P-1} \{2p \sum_{b=(s/P)\mu+1}^{(s/P)(\mu+1)} |\mathcal{X}_b| + 2 \sum_{b=\mu(s/P)+1}^{(s/P)(\mu+1)} |\mathcal{Y}_b|(p + |\mathcal{X}_{b-1}| + |\mathcal{X}_b| + |\mathcal{X}_{b+1}|)\},$$

and the communication complexity is

$$p(4h + \lg P - 7) + 2 \cdot \max_{\mu=0,\ldots,P-1} \{|\mathcal{X}_{(s/P)\mu+1}| + |\mathcal{X}_{(s/P)(\mu+1)}|\}. \tag{2.16}$$

The number of messages sent by each processor is at most $4 \lg P - 3$.

If all s intervals at level h contain equal numbers of charges, and also equal numbers of evaluation points, then $|\mathcal{X}_b| = K/s$ and $|\mathcal{Y}_b| = N/s$, and expression (2.15) gives

$$p^2(\frac{20s}{P} + 10 \lg P - 36) + \frac{2p(K+N)}{P} + \frac{6KN}{sP},$$

which is a minimum when

$$s = s_{\text{opt}} = \frac{1}{p}\sqrt{\frac{3KN}{10}}.$$

Since s must be a power of 2, we may pick $s_{\text{opt}}/\sqrt{2} \leq s \leq s_{\text{opt}}\sqrt{2}$. Then the parallel arithmetic complexity is at most

$$\frac{2p}{P}(3\sqrt{15KN} + K + N) + 10p^2 \lg P - 36p^2.$$

The communication complexity from (2.16) is

$$p(4h + \lg P - 7) + \frac{4K}{s},$$

which when $s_{\text{opt}}/\sqrt{2} \leq s \leq s_{\text{opt}}\sqrt{2}$ gives

$$8p\sqrt{\frac{5K}{3N}} + 4p\lg(\frac{1}{p}\sqrt{\frac{3KN}{10}}) + p(\lg P - 5).$$

2.3.5 Compressing with the SVD: algorithm *LINSVD*

The algorithm *LIN* of Section 2.3.2 may be made more efficient using an SVD compression technique. This idea is used frequently in diverse fields including image compression [4] and even for fast multipole computations [10, 34, 23].

The *singular value decomposition* [13, p. 70] of a real square matrix A of order p is

$$A = U\Sigma V^T,$$

where $U = (\mathbf{u_1}, \ldots, \mathbf{u_p})$ and $V = (\mathbf{v_1}, \ldots, \mathbf{v_p})$ are orthogonal matrices with columns containing the *singular vectors*, and

$$\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_p),$$

with *singular values* $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$.

We can write A as

$$A = \sum_{i=1}^{p} \sigma_i \mathbf{u_i} \mathbf{v_i}^T. \tag{2.17}$$

If A is numerically rank-deficient, then some of the σ_i are close to zero, allowing for

compression by approximating

$$A \approx \sum_{i=1}^{\bar{p}} \sigma_i \mathbf{u_i} \mathbf{v_i}^T, \tag{2.18}$$

for some $\bar{p} \leq p$. The expression on the right-hand side is, in fact, the matrix of rank $\bar{p}$ that is closest to A in ℓ^2 norm (or least squares).

In algorithm LIN, we have been manipulating elements of a p-dimensional function space to represent the potential functions induced from charges. From a least-squares point of view, the best function space of dimension p corresponds to the principal p left singular functions of the operator that maps charges to potential functions, as discussed in Yarvin and Rokhlin [34].

We follow the approach of Dutt et al. [10] in compressing a function space of dimension p to a space of dimension $\bar{p} < p$, by taking an SVD of an operator that maps strengths of charges at p fixed points to potentials at p fixed points. The components of the $\bar{p}$ most significant singular vectors are then taken as coefficients of linear combinations of v_k (or u_k) to be used in $LINSVD$.

To be precise:

- For each $l = 1, \ldots, h$, $\mathcal{M}^{(l)}$ is the $p \times p$ matrix with the entry at (j, k) measuring the potential at $Y_O^{(l)}(t_j)$ due to a charge at $Y_I^{(l)}(t_k)$, taking Y_I and Y_O from Equations (2.7)–(2.8).

$$\mathcal{M}^{(l)}(j,k) = \rho(Y_O^{(l)}(t_j) - Y_I^{(l)}(t_k)) = \rho((\frac{3}{t_j} - t_k)\frac{1}{2^l}). \tag{2.19}$$

- $\bar{p} \leq p$ is the number of terms used in the compressed expansions.

- $U^{(l)}$ is a $p \times \bar{p}$ matrix containing the $\bar{p}$ most significant left singular vectors of $\mathcal{M}^{(l)}$, and $V^{(l)}$ is a matrix containing the $\bar{p}$ most significant right singular vectors of $\mathcal{M}^{(l)}$. Thus

$$U^{(l)}\Sigma^{(l)}V^{(l)T} \approx \mathcal{M}^{(l)}$$

from Approximation (2.18). Here Σ is a diagonal matrix containing the $\bar{p}$ most

significant singular values of $\mathcal{M}^{(l)}$.

We then obtain algorithm $LINSVD$ from algorithm LIN, by making the following replacements:

Algorithm LIN	Algorithm $LINSVD$
G_b	$U^{(h)T} \cdot G_b$
$M_L\Phi^{(l)}$	$(U^{(l-1)T} \cdot M_L \cdot U^{(l)})\Phi^{(l)}$
$M_R\Phi^{(l)}$	$(U^{(l-1)T} \cdot M_R \cdot U^{(l)})\Phi^{(l)}$
$S_L\widetilde{\Psi}^{(l-1)}$	$(V^{(l)T} \cdot S_L \cdot V^{(l-1)})\widetilde{\Psi}^{(l-1)}$
$S_R\widetilde{\Psi}^{(l-1)}$	$(V^{(l)T} \cdot S_R \cdot V^{(l-1)})\widetilde{\Psi}^{(l-1)}$
$T_1\Phi^{(l)}$	$(V^{(l)T} \cdot T_1 \cdot U^{(l)})\Phi^{(l)}$
$T_2\Phi^{(l)}$	$(V^{(l)T} \cdot T_2 \cdot U^{(l)})\Phi^{(l)}$
$T_3\Phi^{(l)}$	$(V^{(l)T} \cdot T_3 \cdot U^{(l)})\Phi^{(l)}$
$T_4\Phi^{(l)}$	$(V^{(l)T} \cdot T_4 \cdot U^{(l)})\Phi^{(l)}$
E_b	$E_b \cdot V^{(h)}$

Here $\Phi_b^{(l)}$ and $\Psi_b^{(l)}$ are $\bar{p}$-vectors representing the far-field and local expansions.

The number of flops in $LINSVD_EVAL$ is obtained by replacing p in expression (2.12) with $\bar{p}$:

$$2\bar{p}(K+N) + (20s - 12h - 44)\bar{p}^2 + 2\sum_{b=1}^{s} |\mathcal{Y}_b|(|\mathcal{X}_{b-1}| + |\mathcal{X}_b| + |\mathcal{X}_{b+1}|). \tag{2.20}$$

Analogously to Section 2.3.3, if all s intervals at level h contain equal numbers of charges, and also equal numbers of evaluation points, then optimally

$$s = s_{\text{opt}} = \frac{3}{10\ln 2} + \frac{1}{10}\sqrt{(\frac{3}{\ln 2})^2 + \frac{30KN}{\bar{p}^2}} \approx \frac{1}{\bar{p}}\sqrt{\frac{3KN}{10}}. \tag{2.21}$$

2.3.6 One-sided algorithms

This section introduces a new algorithm, $LIN1$, which is more complicated than LIN because two separate far-field and local expansions are maintained for each interval. For the same accuracy, $LIN1$ is also less efficient than LIN, but with compression

of function spaces, as algorithm $LIN1SVD$, it can be made to achieve accuracy comparable to that of $LINSVD$ with the same running time.

We describe $LIN1$ and $LIN1SVD$ because $LIN1SVD$ is the virtual-charge version of Algorithm 3.2 cited by Dutt et al. [10] as the most efficient algorithm for problem (2.1) on the line. In Section 2.7 we shall see that $LIN1SVD$ does not have any advantage over $LINSVD$.

The algorithm $LIN1(p, h, \mathbf{x}, \mathbf{y}, \mathbf{q})$ has the same input parameters as LIN.

We define the following:

- $t_1^L, \ldots, t_p^L$ and $t_1^R, \ldots, t_p^R$ denote the Chebyshev nodes on the intervals $[-1, 0]$ and $[0, 1]$, respectively:

$$\begin{aligned} t_j^L &= \frac{t_j - 1}{2}, \\ t_j^R &= \frac{t_j + 1}{2}. \end{aligned}$$

- $v_1^L(t), \ldots, v_p^L(t)$ and $v_1^R(t), \ldots, v_p^R(t)$ denote the rational interpolating functions used on $[-1, 0]$ and $[0, 1]$, respectively:

$$\begin{aligned} v_k^L(t) &= (\prod_{\substack{j=1 \\ j \neq k}}^{p} \frac{t - t_j^L}{t_k^L - t_j^L})(\prod_{i=1}^{p-1} \frac{t_k - \gamma_i}{t - \gamma_i}), \\ \text{and} v_k^R(t) &= (\prod_{\substack{j=1 \\ j \neq k}}^{p} \frac{t - t_j^R}{t_k^R - t_j^R})(\prod_{i=1}^{p-1} \frac{t_k - \gamma_i}{t - \gamma_i}). \end{aligned}$$

The algorithm computes:

- for $l = 2, \ldots, h$ and $b = 1, \ldots, 2^l$;
  - $\Phi_b^{L(l)}$ and $\Phi_b^{R(l)}$ are p-vectors containing the coefficients of the left and right far-field expansions for the bth subinterval at level l.
  - $\Psi_b^{L(l)}$ and $\Psi_b^{R(l)}$ are p-vectors containing the coefficients of the left and right local expansions for the bth subinterval at level l.
- If $b \notin \{1, \ldots, 2^l\}$ then $\Phi_b^{(l)}$ and $\Psi_b^{(l)}$ are taken to be zero.

To convert between expansions at different levels, the algorithm uses the following operators which are fixed by p:

- M_L^L, M_L^R, M_R^L, and M_R^R are $p \times p$ matrices for obtaining far-field expansions for subintervals from the far-field expansions for their children:

$$M_L^L(j,k) = v_k^L(X_O^{(l+1)}(Y_O^{(l)}(t_j^L) + w_{l+1})) = v_k^L(\frac{3t_j^L}{6+t_j^L}) \tag{2.22}$$

$$M_L^R(j,k) = v_k^R(X_O^{(l+1)}(Y_O^{(l)}(t_j^R) + w_{l+1})) = v_k^R(\frac{3t_j^R}{6+t_j^R}) \tag{2.23}$$

$$M_R^L(j,k) = v_k^L(X_O^{(l+1)}(Y_O^{(l)}(t_j^L) - w_{l+1})) = v_k^L(\frac{3t_j^L}{6-t_j^L}) \tag{2.24}$$

$$M_R^R(j,k) = v_k^R(X_O^{(l+1)}(Y_O^{(l)}(t_j^R) - w_{l+1})) = v_k^R(\frac{3t_j^R}{6-t_j^R}). \tag{2.25}$$

- S_L^L, S_L^R, S_R^L, and S_R^R are $p \times p$ matrices for obtaining local expansions for subintervals from local expansions for their parent:

$$S_L^L(j,k) = S_L^R(j,k) = v_k(X_I^{(l-1)}(Y_I^{(l)}(t_j) - w_l)) = v_k(t_j^L) \tag{2.26}$$

$$S_R^L(j,k) = S_R^R(j,k) = v_k(X_I^{(l-1)}(Y_I^{(l)}(t_j) + w_l)) = v_k(t_j^R). \tag{2.27}$$

- T_1^L, T_2^L, T_1^R, and T_2^R are $p \times p$ matrices for obtaining local expansions from far-field expansions.

$$T_1^L(j,k) = v_k^R(X_O^{(l)}(Y_I^{(l)}(t_j) + 4w_l)) = v_k^R(\frac{3}{t_j+4}) \tag{2.28}$$

$$T_2^L(j,k) = v_k^R(X_O^{(l)}(Y_I^{(l)}(t_j) + 6w_l)) = v_k^R(\frac{3}{t_j+6}) \tag{2.29}$$

$$T_1^R(j,k) = v_k^L(X_O^{(l)}(Y_I^{(l)}(t_j) - 4w_l)) = v_k^L(\frac{3}{t_j-4}) \tag{2.30}$$

$$T_2^R(j,k) = v_k^L(X_O^{(l)}(Y_I^{(l)}(t_j) - 6w_l)) = v_k^L(\frac{3}{t_j-6}) \tag{2.31}$$

The function $LIN1_INIT(p, h, \mathbf{x}, \mathbf{y})$ sets up operators that depend on $\mathbf{x}$ and $\mathbf{y}$. For each $b = 1, \ldots, s$:

- G_b^L and G_b^R are $p \times |\mathcal{X}_b|$ matrix for obtaining far-field expansion coefficients from charges in I_b:

$$G_b^L(j,k) = \rho(Y_O^{(h)}(t_j^L) - (x_k - c_b)) = \rho(\frac{t_j^L}{3w - t_j(x_k - c_b)}), \quad (2.32)$$

$$G_b^R(j,k) = \rho(Y_O^{(h)}(t_j^L) - (x_k - c_b)) = \rho(\frac{t_j^R}{3w - t_j(x_k - c_b)}). \quad (2.33)$$

  G_b^L and G_b^R use only the columns with indices in $\mathcal{X}_b$.

- E_b and D_b are as in LIN.

Algorithm 2.3.2

$\mathbf{f} = LIN1(p, h, \mathbf{x}, \mathbf{y}, \mathbf{q})$ evaluates $f_j = \sum_k \rho(y_j - x_k) \cdot q_k$ for $x_k, y_j \in [-1, 1]$, using expansions with p terms and with h hierarchy levels.

function $LIN1_INIT(p, h, \mathbf{x}, \mathbf{y})$:

for $b = 1, \ldots, 2^h$: Compute G_b^L, G_b^R, E_b, and D_b using Equations (2.32)–(2.33) and (2.10)–(2.11).

end function $LIN1_INIT$.

function $\mathbf{f} = LIN1_EVAL(\mathbf{q})$:

1. **comment** [Get coefficients of far-field expansions.]
   $(\Phi_1^{L(h)}, \ldots, \Phi_s^{L(h)}) = (G_1^L \mathbf{q}_{\mathcal{X}_1}, \ldots, G_s^L \mathbf{q}_{\mathcal{X}_s})$
   $(\Phi_1^{R(h)}, \ldots, \Phi_s^{R(h)}) = (G_1^R \mathbf{q}_{\mathcal{X}_1}, \ldots, G_s^R \mathbf{q}_{\mathcal{X}_s})$

2. **comment** [Get coefficients of local expansions.]
   $(\Psi_1^{R(h)}, \ldots, \Psi_s^{R(h)}) = LIN1_RECUR_R(h, \Phi_1^{L(h)}, \ldots, \Phi_s^{L(h)})$
   $(\Psi_1^{L(h)}, \ldots, \Psi_s^{L(h)}) = LIN1_RECUR_L(h, \Phi_1^{R(h)}, \ldots, \Phi_s^{R(h)})$

3. **comment** [Evaluate potentials from local expansions and direct interactions.]
   for $b = 1, \ldots, s$,

$$f_{\mathcal{Y}_b} = E_b \Psi_b^{L(h)} + E_b \Psi_b^{R(h)} + D_b \mathbf{q}_{\mathcal{X}_{b-1} \cup \mathcal{X}_b \cup \mathcal{X}_{b+1}}$$

end

end function $LIN1_EVAL$.

function $(\Psi_1^{R(l)}, \ldots, \Psi_{2^l}^{R(l)}) = LIN1_RECUR_R(l, \Phi_1^{L(l)}, \ldots, \Phi_{2^l}^{L(l)})$:

if $l = 2$ **then**

$$(\Psi_1^{R(2)}, \Psi_2^{R(2)}, \Psi_3^{R(2)}, \Psi_4^{R(2)}) = (T_1^R \Phi_3^{L(2)} + T_2^R \Phi_4^{L(2)}, T_1^R \Phi_4^{L(2)}, 0, 0)$$

else

1. **comment** [Compute coefficients of far-field expansions at level $l-1$.]

   for $b = 1, \ldots, 2^{l-1}$:

   $$\widetilde{\Phi}_b^L = M_L^L \Phi_{2b-1}^{L(l)} + M_R^L \Phi_{2b}^{L(l)}$$

   end

2. **comment** [Get coefficients of local expansions at level $l-1$.]

   $$(\widetilde{\Psi}_1^R, \ldots, \widetilde{\Psi}_{2^{l-1}}^R) = LIN1_RECUR_R(l-1, \widetilde{\Phi}_1^L, \ldots, \widetilde{\Phi}_{2^{l-1}}^L)$$

3. **comment** [Compute coefficients of local expansions at level l.]

   for $b = 1, \ldots, 2^{l-1} - 1$:

   $$\Psi_{2b-1}^{R(l)} = S_L^R \widetilde{\Psi}_b^R + T_1^R \Phi_{2b+1}^{L(l)} + T_2^R \Phi_{2b+2}^{L(l)}$$
   $$\Psi_{2b}^{R(l)} = S_R^R \widetilde{\Psi}_b^R + T_1^R \Phi_{2b+2}^{L(l)}$$

   end

end

end function $LIN1_RECUR_R$.

function $(\Psi_1^{L(l)}, \ldots, \Psi_{2^l}^{L(l)}) = LIN1_RECUR_L(l, \Phi_1^{R(l)}, \ldots, \Phi_{2^l}^{R(l)})$:

if $l = 2$ **then**

$$(\Psi_1^{L(2)}, \Psi_2^{L(2)}, \Psi_3^{L(2)}, \Psi_4^{L(2)}) = (0, 0, T_1^L \Phi_1^{R(2)}, T_1^L \Phi_2^{R(2)} + T_2^L \Phi_1^{L(2)})$$

else

1. **comment** [Compute coefficients of far-field expansions at level $l-1$.]

   for $b = 1, \ldots, 2^{l-1}$:

   $\widetilde{\Phi}_b^R = M_L^R \Phi_{2b-1}^{R(l)} + M_R^R \Phi_{2b}^{R(l)}$

   end

2. **comment** [Get coefficients of local expansions at level $l-1$.]

   $(\widetilde{\Psi}_1^L, \ldots, \widetilde{\Psi}_{2^{l-1}}^L) = LIN1_RECUR_L(l-1, \widetilde{\Phi}_1^R, \ldots, \widetilde{\Phi}_{2^{l-1}}^R)$

3. **comment** [Compute coefficients of local expansions at level l.]

   for $b = 1, \ldots, 2^{l-1} - 1$:

   $\Psi_{2b-1}^{L(l)} = S_L^L \widetilde{\Psi}_b^L + T_1^L \Phi_{2b-3}^{R(l)}$

   $\Psi_{2b}^{L(l)} = S_R^L \widetilde{\Psi}_b^L + T_1^L \Phi_{2b-2}^{(l)} + T_2^L \Phi_{2b-3}^{(l)}$

   end

end

end function $LIN1_RECUR_L$.

The complexity for $LIN1_EVAL$ is as follows.

For $LIN1_RECUR_L(l, \Phi_1^{R(l)}, \ldots, \Phi_{2^l}^{R(l)})$:

- If $l = 2$ then flop count is $3 \cdot 2p^2 = 6p^2$.
- Otherwise:
  1. flop count is $2^{l-1} \cdot 2 \cdot 2p^2 = 2^l \cdot 2p^2$;
  2. flop count for $LIN1_RECUR_L(l-1, \Phi_1^{R(l-1)}, \ldots, \Phi_{2^{l-1}}^{R(l-1)})$;
  3. flop count is $(2^{l-1} - 1) \cdot 3 \cdot 2p^2 + (2^{l-1} - 2) \cdot 2 \cdot 2p^2 = 2^l \cdot 5p^2 - 14p^2$.

So the total is $14p^2 \cdot 2^l - 14p^2$ plus the flop count for $LIN1_RECUR_L(l-1, \Phi_1^{(l-1)}, \ldots, \Phi_{2^{l-1}}^{(l-1)})$.

The total comes to $(14 \cdot 2^l - 14l - 22)p^2$. For $LIN1_RECUR_R$, the complexity is the same.

Flop counts for $LIN1_EVAL$:

1. flop count is $4pK$;

2. flop count is $(28 \cdot 2^h - 28h - 44)p^2$;

3. flop count is $2pN + 2\sum_{b=1}^{s} |\mathcal{Y}_b|(|\mathcal{X}_{b-1}| + |\mathcal{X}_b| + |\mathcal{X}_{b+1}|)$.

The total for $LIN1_EVAL$ is then

$$4pK + 2pN + (28s - 28h - 44)p^2 + 2\sum_{b=1}^{s} |\mathcal{Y}_b|(|\mathcal{X}_{b-1}| + |\mathcal{X}_b| + |\mathcal{X}_{b+1}|). \tag{2.34}$$

In a similar way as Equation (2.13) in Section 2.3.3, one can show that if all s intervals at level h contain equal numbers of charges, and also equal numbers of evaluation points, then optimally

$$s = s_{\text{opt}} \approx \frac{1}{p}\sqrt{\frac{3KN}{14}}. \tag{2.35}$$

We can also compress function spaces using an SVD, to obtain algorithm $LIN1SVD$.

- ${\mathcal{M}_{\mathcal{L}}}^{(l)}$ is a $p \times p$ matrix such that

$$\mathcal{M}_{\mathcal{L}}^{(l)}(j,k) = \rho(Y_O^{(l)}(t_j^L) - Y_I^{(l)}(t_k)) = \rho((\frac{3}{t_j^L} - t_k)\frac{1}{2^l}).$$

- ${\mathcal{M}_{\mathcal{R}}}^{(l)}$ is a $p \times p$ matrix such that

$$\mathcal{M}_{\mathcal{R}}^{(l)}(j,k) = \rho(Y_O^{(l)}(t_j^R) - Y_I^{(l)}(t_k)) = \rho((\frac{3}{t_j^R} - t_k)\frac{1}{2^l}).$$

- $U_L^{(l)}$ is a $p \times \bar{p}$ matrix containing the $\bar{p}$ most significant left singular vectors of ${\mathcal{M}_{\mathcal{L}}}^{(l)}$, and $V_L^{(l)}$ is a matrix containing the $\bar{p}$ most significant right singular vectors of ${\mathcal{M}_{\mathcal{L}}}^{(l)}$. Thus

$$U_L^{(l)}\Sigma_L^{(l)}V_L^{(l)T} \approx {\mathcal{M}_{\mathcal{L}}}^{(l)}.$$

  Here $\Sigma_L^{(l)}$ is a diagonal matrix containing the $\bar{p}$ most significant singular values of ${\mathcal{M}_{\mathcal{L}}}^{(l)}$. Equality holds if $\bar{p} = p$.

- $U_R^{(l)}$ is a $p \times \bar{p}$ matrix containing the $\bar{p}$ most significant left singular vectors of $\mathcal{M}_{\mathcal{R}}{}^{(l)}$, and $V_R^{(l)}$ is a matrix containing the $\bar{p}$ most significant right singular vectors of $\mathcal{M}_{\mathcal{R}}{}^{(l)}$. Thus

$$U_R^{(l)} \Sigma_R^{(l)} V_R^{(l)T} \approx \mathcal{M}_{\mathcal{R}}{}^{(l)}.$$

Here $\Sigma_R^{(l)}$ is a diagonal matrix containing the $\bar{p}$ most significant singular values of $\mathcal{M}_{\mathcal{R}}{}^{(l)}$. Equality holds if $\bar{p} = p$.

To obtain $LIN1SVD$ from $LIN1$, we make the following replacements.

Algorithm $LIN1$	Algorithm $LIN1SVD$
G_b^L	$U_L^{(n)T} \cdot G_b^L$
G_b^R	$U_R^{(n)T} \cdot G_b^R$
M_L^L	$U_L^{(l)T} \cdot M_L^L \cdot U_L^{(l+1)}$
M_L^R	$U_R^{(l)T} \cdot M_L^R \cdot U_R^{(l+1)}$
M_R^L	$U_L^{(l)T} \cdot M_R^L \cdot U_L^{(l+1)}$
M_R^R	$U_R^{(l)T} \cdot M_R^R \cdot U_R^{(l+1)}$
S_L^L	$V_L^{(l)T} \cdot S_L^L \cdot V_L^{(l-1)}$
S_L^R	$V_L^{(l)T} \cdot S_L^R \cdot V_L^{(l-1)}$
S_R^L	$V_R^{(l)T} \cdot S_R^L \cdot V_R^{(l-1)}$
S_R^R	$V_R^{(l)T} \cdot S_R^R \cdot V_R^{(l-1)}$
T_1^L	$V_L^{(l)T} \cdot T_1^L \cdot U_R^{(l)}$
T_2^L	$V_L^{(l)T} \cdot T_2^L \cdot U_R^{(l)}$
T_1^R	$V_R^{(l)T} \cdot T_1^R \cdot U_L^{(l)}$
T_2^R	$V_R^{(l)T} \cdot T_2^R \cdot U_L^{(l)}$
$E_b \Psi_b^{L(h)}$	$E_b \cdot V_L^{(n)} \Psi_b^{L(h)}$
$E_b \Psi_b^{R(h)}$	$E_b \cdot V_R^{(n)} \Psi_b^{R(h)}$

The number of flops used in $LIN1SVD_EVAL$ is obtained by replacing p in expression (2.34) with $\bar{p}$, and also replacing $2pN$ (for multiplying by E_b in Step 3) with $4\bar{p}N$ (for multiplying by $E_b \cdot V_L^{(n)}$ and $E_b \cdot V_R^{(n)}$) to obtain:

$$4\bar{p}K + 4\bar{p}N + (28s - 28h - 44)\bar{p}^2 + 2\sum_{b=1}^{s} |\mathcal{Y}_b|(|\mathcal{X}_{b-1}| + |\mathcal{X}_b| + |\mathcal{X}_{b+1}|). \qquad (2.36)$$

If all s intervals at level h contain equal numbers of charges, and also equal numbers of evaluation points, then optimally

$$s = s_{\text{opt}} \approx \frac{1}{\bar{p}}\sqrt{\frac{3KN}{14}}. \tag{2.37}$$

2.4 Algorithms on the circle: $CIRC$ and variants

2.4.1 Different approaches

$CIRC$ is the periodic version of LIN. There are two alternative choices leading to four different algorithms:

- Interpolation functions can be either the polynomials u_k from Definition 2.2.6, or the rational functions v_k from Definition 2.2.7 representing potentials due to "virtual charges".
- Singular value decompositions can be used or not used to compress expansions, as described in Section 2.4.5.

The following table summarizes the different algorithms on the circle that are used in this chapter.

Algorithm	interpolation functions	SVD compression	symbol in Fig. 2-7
$CIRC$	v_k	no	○
$CIRCSVD$	v_k	yes	○
FMM of [11]	u_k	no	△
FMM of [11] with compression	u_k	yes	△

One-sided algorithms, as described in Section 2.3.6 on the line, are possible but not natural on the circle. We do not discuss them here.

2.4.2 Description

The idea again is to interpolate on $[-1, 1]$ by mapping the potential evaluation points to $[-1, 1]$.

We use two types of expansions for intervals $[c - w, c + w]$:

- Far-field expansions, where the charges are located in the interval, $y \in (c - w, c + w)$, and evaluation points are at y with $y \in (c + 3w, c + 2\pi - 3w)$. If
$$x = X_O(y) = \frac{3\tan(w/2)}{\tan((y-c)/2)}$$
then X_O maps evaluation points to $|x| \leq 1$ and charge positions to $|x| \geq 3$. The inverse function of X_O is
$$y = Y_O(x) = 2\arctan(\frac{3\tan(w/2)}{x}) + c.$$

- Local expansions, where the evaluation points are located in the interval, $y \in (c - w, c + w)$, and charges are located at points y with $y \in (c + 3w, c + 2\pi - 3w)$. If
$$x = X_I(y) = \frac{\tan((y-c)/2)}{\tan(w/2)}$$
then X_I maps evaluation points to $|x| \leq 1$ and charge positions to $|x| \geq 3$. The inverse function of X_I is
$$y = Y_I(x) = 2\arctan(x\tan(w/2)) + c.$$

Figure 2-2 illustrates these mappings, which are analogous to the mappings of the same names described in Section 2.3.2 and illustrated in Figure 2-1 for LIN.

The algorithm is $CIRC(p, h, \mathbf{x}, \mathbf{y}, \mathbf{q})$, where we have:

- p is the length of expansions to be used in the algorithm;
- h is the maximum number of levels to be used in the algorithm;
- $\mathbf{x}$ is a vector containing charge positions $x_1, \ldots, x_K \in [0, 2\pi]$;

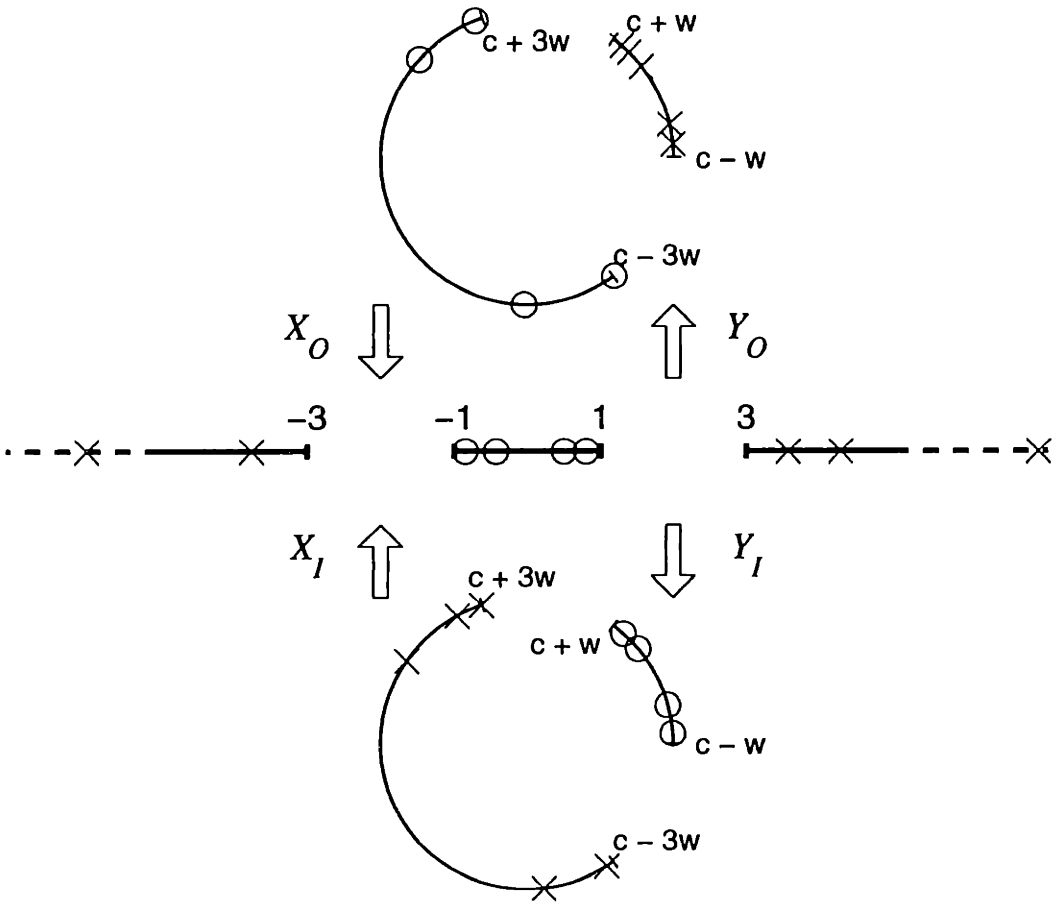


Figure 2-2: Transformations used in approximating the potential at points marked with "o" due to charges at points marked with "×". At top, charges are in the narrower interval and are well separated from the evaluation points, which are mapped to $[-1, 1]$ using X_O. At bottom, charges are in the wider interval and are well separated from the evaluation points, which are mapped to $[-1, 1]$ using X_I.

- **y** is a vector containing evaluation points $y_1, \ldots, y_N \in [0, 2\pi]$;
- **q** is a vector of length K with its kth component q_k being the strength of the charge at point x_k.

The algorithm also uses the following:

- for $l = 1, \ldots, h$ we have:
  - ◇ $w_l = \pi/2^l$ is the half-width of subintervals at level l;

- ◇ $X_I^{(l)}$ and $Y_I^{(l)}$ are functions used to transform between the interval $[-1,1]$ and $[-w_l, w_l]$:

$$X_I^{(l)}(y) = \frac{\tan(y/2)}{\tan(w_l/2)}, \qquad Y_I^{(l)}(x) = 2\arctan(x\tan(w_l/2)). \tag{2.38}$$

- ◇ $X_O^{(l)}$ and $Y_O^{(l)}$ are functions used to transform between the interval $[-1,1]$ and $(3w_l, 2\pi - 3w_l)$:

$$X_O^{(l)}(y) = \frac{3\tan(w_l/2)}{\tan(y/2)}, \qquad Y_O^{(l)}(x) = 2\arctan(3\tan(w_l/2)/x). \tag{2.39}$$

- $s = 2^h$ is the number of intervals at level h;
- for $b = 1, \ldots, s$, we define the following for the intervals at level h:
  - ◇ $I_b = [c_b - w_h, c_b + w_h]$ is the bth subinterval at level h;
  - ◇ $c_b = (2b-1)w_h$ is the center of I_b;
  - ◇ $\mathcal{X}_b = \{k : x_k \in I_b\}$ is the set of indices of charge positions in I_b.
  - ◇ $\mathcal{Y}_b = \{j : y_j \in I_b\}$ is the set of indices of evaluation points in I_b.
- for $l = 2, \ldots, h$ and $b = 1, \ldots, 2^l$, the algorithm computes:
  - ◇ $\Phi_b^{(l)}$ is a p-vector containing the coefficients of the far-field expansion for the bth subinterval at level l.
  - ◇ $\Psi_b^{(l)}$ is a p-vector containing the coefficients of the local expansion the bth subinterval at level l.

  If $b \notin \{1, \ldots, 2^l\}$ then the index is taken to wrap around: $\Phi_{b+2^l}^{(l)}$ is identified with $\Phi_b^{(l)}$, and $\Psi_{b+2^l}^{(l)}$ is identified with $\Psi_b^{(l)}$.

To convert between expansions at different levels, the algorithm uses the following operators which are fixed by p.

- For $l = 3, \ldots, h+1$, set

- $\alpha_l = \tan(w_l)$;
- $\beta_l = \tan(3w_l)$.

- For $l = 2, \ldots, h-1$, $M_L^{(l)}$ and $M_R^{(l)}$ are $p \times p$ matrices for obtaining far-field expansions at level l from far-field expansions at level $l+1$:

$$M_L^{(l)}(j,k) = v_k(X_O^{(l+1)}(Y_O^{(l)}(t_j) + w_{l+1})) = v_k(3\alpha_{l+2}\frac{t_j - 3\alpha_{l+1}\alpha_{l+2}}{3\alpha_{l+1} + t_j\alpha_{l+2}})$$
$$M_R^{(l)}(j,k) = v_k(X_O^{(l+1)}(Y_O^{(l)}(t_j) - w_{l+1})) = v_k(3\alpha_{l+2}\frac{t_j + 3\alpha_{l+1}\alpha_{l+2}}{3\alpha_{l+1} - t_j\alpha_{l+2}})$$

- For $l = 3, \ldots, h$, $S_L^{(l)}$ and $S_R^{(l)}$ are $p \times p$ matrices for obtaining local expansions at level l from local expansions at level $l-1$:

$$S_L^{(l)}(j,k) = v_k(X_I^{(l-1)}(Y_I^{(l)}(t_j) - w_l)) = v_k(\frac{t_j - 1}{\alpha_l(1/\alpha_{l+1} + t_j\alpha_{l+1})})$$
$$S_R^{(l)}(j,k) = v_k(X_I^{(l-1)}(Y_I^{(l)}(t_j) + w_l)) = v_k(\frac{t_j + 1}{\alpha_l(1/\alpha_{l+1} - t_j\alpha_{l+1})})$$

- For $l = 3, \ldots, h$, $T_1^{(l)}$, $T_2^{(l)}$, $T_3^{(l)}$, and $T_4^{(l)}$ are $p \times p$ matrices for obtaining local expansions from far-field expansions at level l:

$$\begin{aligned}
T_1^{(l)}(j,k) &= v_k(X_O^{(l)}(Y_I^{(l)}(t_j) + 6w_l)) = v_k(3\alpha_{l+1}\frac{1 - t_j\beta_l\alpha_{l+1}}{t_j\alpha_{l+1} + \beta_l}) \\
T_2^{(l)}(j,k) &= v_k(X_O^{(l)}(Y_I^{(l)}(t_j) + 4w_l)) = v_k(3\alpha_{l+1}\frac{1 - t_j\alpha_{l-1}\alpha_{l+1}}{t_j\alpha_{l+1} + \alpha_{l-1}}) \\
T_3^{(l)}(j,k) &= v_k(X_O^{(l)}(Y_I^{(l)}(t_j) - 4w_l)) = v_k(3\alpha_{l+1}\frac{1 + t_j\alpha_{l-1}\alpha_{l+1}}{t_j\alpha_{l+1} - \alpha_{l-1}}) \\
T_4^{(l)}(j,k) &= v_k(X_O^{(l)}(Y_I^{(l)}(t_j) - 6w_l)) = v_k(3\alpha_{l+1}\frac{1 + t_j\beta_l\alpha_{l+1}}{t_j\alpha_{l+1} - \beta_l})
\end{aligned}$$

- F is a $p \times p$ matrix for obtaining local expansions from far-field expansions at level 2:

$$F = v_k(X_O^{(2)}(Y_I^{(2)}(t_j) + \pi)) = v_k(-3\alpha_3^2 t_j) \tag{2.40}$$

The function $CIRC_INIT(p, h, \mathbf{x}, \mathbf{y})$ initializes operators that depend on $\mathbf{x}$ and $\mathbf{y}$. For each $b = 1, \ldots, s$:

- G_b is a $p \times |\mathcal{X}_b|$ matrix for obtaining far-field expansion coefficients from charges in I_b:

$$G_b(j,k) = \rho(Y_O^{(h)}(t_j) - (x_k - c_b)) = \rho(2\arctan(3\alpha_{n+1}/t_j) - (x_k - c_b)) \quad (2.41)$$

  G_b uses only the columns with indices in $\mathcal{X}_b$.

- E_b is a $|\mathcal{Y}_b| \times p$ matrix for obtaining potentials at points in I_b from local expansion coefficients:

$$E_b(j,k) = v_k(X_I^{(h)}(y_j - c_b)) = v_k((y_j - c_b)s) = v_k(\frac{\tan((y_j - c_b)/2)}{\alpha_{n+1}}), \quad (2.42)$$

  E_b uses only the rows with indices in $\mathcal{Y}_b$.

- D_b is a $|\mathcal{Y}_b| \times |\mathcal{X}_{b-1} \cup \mathcal{X}_b \cup \mathcal{X}_{b+1}|$ matrix for obtaining potentials at points in I_b due to charges in I_b and its neighbors:

$$D_b(j,k) = \rho(y_j - x_k) \quad (2.43)$$

  D_b uses only the rows with indices in $\mathcal{Y}_b$ and the columns with indices in $\mathcal{X}_{b-1} \cup \mathcal{X}_b \cup \mathcal{X}_{b+1}$. Here, $\mathcal{X}_0$ is identified with $\mathcal{X}_s$ and $\mathcal{X}_{s+1}$ is identified with $\mathcal{X}_1$.

Algorithm 2.4.1

$\mathbf{f} = CIRC(p, h, \mathbf{x}, \mathbf{y}, \mathbf{q})$ evaluates $f_j = \sum_k \rho(y_j - x_k) \cdot q_k$ for $x_k, y_j \in [0, 2\pi]$, using expansions with p terms and with h hierarchy levels.

function $CIRC_INIT(p, h, \mathbf{x}, \mathbf{y})$:

  for $b = 1, \ldots, 2^h$: Compute G_b, E_b, and D_b using Equations (2.41)–(2.43).

end function $CIRC_INIT$.

function $\mathbf{f} = CIRC_EVAL(\mathbf{q})$:

1. **comment** [Get coefficients of far-field expansions.]

   $(\Phi_1^{(h)}, \ldots, \Phi_s^{(h)}) = (G_1 \mathbf{q}_{x_1}, \ldots, G_s \mathbf{q}_{x_s})$

2. **comment** [Get coefficients of local expansions.]

   $(\Psi_1^{(h)}, \ldots, \Psi_s^{(h)}) = CIRC_RECUR(h, \Phi_1^{(h)}, \ldots, \Phi_s^{(h)})$

3. **comment** [Evaluate potentials from local expansions and direct interactions.]

   for $b = 1, \ldots, s$:

   $f_{y_b} = E_b \Psi_b^{(h)} + D_b \mathbf{q}_{x_{b-1} \cup x_b \cup x_{b+1}}$

   end

end function $CIRC_EVAL$.

function $(\Psi_1^{(l)}, \ldots, \Psi_{2^l}^{(l)}) = CIRC_RECUR(l, \Phi_1^{(l)}, \ldots, \Phi_{2^l}^{(l)})$:

if $l = 2$ **then**

$(\Psi_1^{(2)}, \Psi_2^{(2)}, \Psi_3^{(2)}, \Psi_4^{(2)}) = (F\Phi_3^{(2)}, F\Phi_4^{(2)}, F\Phi_1^{(2)}, F\Phi_2^{(2)})$

else

1. **comment** [Compute coefficients of far-field expansions at level $l-1$.]

   for $b = 1, \ldots, 2^{l-1}$:

   $\widetilde{\Phi}_b = M_L^{(l-1)} \Phi_{2b-1}^{(l)} + M_R^{(l-1)} \Phi_{2b}^{(l)}$

   end

2. **comment** [Get coefficients of local expansions at level $l-1$.]

   $(\widetilde{\Psi}_1, \ldots, \widetilde{\Psi}_{2^{l-1}}) = CIRC_RECUR(l-1, \widetilde{\Phi}_1, \ldots, \widetilde{\Phi}_{2^{l-1}})$

3. **comment** [Compute coefficients of local expansions at level l.]

   for $b = 1, \ldots, 2^{l-1}$:

   $\Psi_{2b-1}^{(l)} = S_L^{(l)} \widetilde{\Psi}_b + T_2^{(l)} \Phi_{2b-3}^{(l)} + T_3^{(l)} \Phi_{2b+1}^{(l)} + T_4^{(l)} \Phi_{2b+2}^{(l)}$

   $\Psi_{2b}^{(l)} = S_R^{(l)} \widetilde{\Psi}_b + T_1^{(l)} \Phi_{2b-3}^{(l)} + T_2^{(l)} \Phi_{2b-2}^{(l)} + T_3^{(l)} \Phi_{2b+2}^{(l)}$

   end

end

end function $CIRC_RECUR$.

2.4.3 Complexity

Assume $\mathbf{q}$ is real. For $CIRC_RECUR(l, \Phi_1^{(l)}, \ldots, \Phi_{2^l}^{(l)})$:

- If $l = 2$ then flop count is $4 \cdot 2p^2 = 8p^2$.
- Otherwise:
  1. flop count is $2^{l-1} \cdot 2 \cdot 2p^2 = 2^l \cdot 2p^2$;
  2. flop count for $CIRC_RECUR(l-1, \Phi_1^{(l-1)}, \ldots, \Phi_{2^{l-1}}^{(l-1)})$;
  3. flop count is $2^{l-1} \cdot 8 \cdot 2p^2 = 2^l \cdot 8p^2$;

  So the total is $2^l \cdot 10p^2$ plus the flop count for $CIRC_RECUR(l - 1, \Phi_1^{(l-1)}, \ldots, \Phi_{2^{l-1}}^{(l-1)})$.

The total comes to $(20 \cdot 2^l - 72)p^2$.

Flop counts for $CIRC_EVAL$:

1. $2pK$;
2. $(20 \cdot 2^h - 72)p^2$;
3. $2pN + 2\sum_{b=1}^{s} |\mathcal{Y}_b|(|\mathcal{X}_{b-1}| + |\mathcal{X}_b| + |\mathcal{X}_{b+1}|)$.

The total number of flops for $CIRC_EVAL$ is then

$$2p(K+N) + (20s - 72)p^2 + 2\sum_{b=1}^{s} |\mathcal{Y}_b|(|\mathcal{X}_{b-1}| + |\mathcal{X}_b| + |\mathcal{X}_{b+1}|). \tag{2.44}$$

If all s intervals at level h contain equal numbers of charges, and also equal numbers of evaluation points, then $|\mathcal{X}_b| = K/s$ and $|\mathcal{Y}_b| = N/s$, and expression (2.44) gives

$$2p(K+N) + (20s - 72)p^2 + \frac{6KN}{s}, \tag{2.45}$$

which is a minimum when

$$s = s_{\text{opt}} = \frac{1}{p}\sqrt{\frac{3KN}{10}}. \tag{2.46}$$

Since s must be a power of 2, we may pick $s_{\text{opt}}/\sqrt{2} \le s \le s_{\text{opt}}\sqrt{2}$. Then the number of flops is at most

$$2p(3\sqrt{15KN} + K + N) - 72p^2.$$

2.4.4 Parallel complexity

With the same model as used for LIN in Section 2.3.4, the only difference in the parallel arithmetic complexity of $CIRC$ is that when $l = 2$, the maximum number of flops to be performed is reduced by $2p^2$.

The parallel arithmetic complexity of $CIRC_EVAL$ is then

$$\begin{aligned} & p^2\left(\frac{20s}{P} + 10\lg P - 38\right) \\ + & \max_{\mu=0,\ldots,P-1}\{2p \sum_{b=\mu(s/P)+1}^{(\mu+1)s/P} |\mathcal{X}_b| + 2 \sum_{b=\mu(s/P)+1}^{(\mu+1)(s/P)} |\mathcal{Y}_b|(p + |\mathcal{X}_{b-1}| + |\mathcal{X}_b| + |\mathcal{X}_{b+1}|)\}, \end{aligned} \tag{2.47}$$

and the communication complexity is, again,

$$p(4h + \lg P - 7) + 2 \cdot \max_{\mu=0,\ldots,P-1}\{|\mathcal{X}_{\mu(s/P)+1}| + |\mathcal{X}_{(\mu+1)(s/P)}|\}. \tag{2.48}$$

The number of messages sent by each processor is at most $4\lg P - 3$.

If all s intervals at level h contain equal numbers of charges, and also equal numbers of evaluation points, then $|\mathcal{X}_b| = K/s$ and $|\mathcal{Y}_b| = N/s$, and expression (2.47) gives

$$p^2\left(\frac{20s}{P} + 10\lg P - 38\right) + \frac{2p(K+N)}{P} + \frac{6KN}{sP},$$

which is a minimum when $s = s_{\text{opt}}$ from Equation (2.46). Picking s a power of 2 with $s_{\text{opt}}/\sqrt{2} \le s \le s_{\text{opt}}\sqrt{2}$, then the parallel arithmetic complexity is at most

$$\frac{2p}{P}(3\sqrt{15KN} + K + N) + 10p^2\lg P - 38p^2. \tag{2.49}$$

The communication complexity from (2.48) is

$$p(4h + \lg P - 7) + \frac{4K}{s}, \tag{2.50}$$

which when $s_{\text{opt}}/\sqrt{2} \leq s \leq s_{\text{opt}}\sqrt{2}$ gives

$$p(4\lg(\frac{1}{p}\sqrt{\frac{3KN}{10}}) + 8\sqrt{\frac{5K}{3N}} + \lg P - 5). \tag{2.51}$$

2.4.5 Compressing with the SVD: *CIRCSVD*

As we did with *LIN* in Section 2.3.5, we can again use SVD compression to speed up the computation.

- For each $l = 1, \ldots, h$, $\mathcal{M}^{(l)}$ is a $p \times p$ matrix such that

$$\mathcal{M}^{(l)}(j,k) = \rho(Y_O^{(l)}(t_j) - Y_I^{(l)}(t_k)) = \rho(2\arctan(t_k\alpha_{l+1}) - 2\arctan(3\alpha_{l+1}/t_j)) \tag{2.52}$$

- $U^{(l)}$ is a $p \times \bar{p}$ matrix containing the $\bar{p}$ most significant left singular vectors of $\mathcal{M}^{(l)}$, and $V^{(l)}$ is a matrix containing the $\bar{p}$ most significant right singular vectors of $\mathcal{M}^{(l)}$. Thus

$$U^{(l)}\Sigma^{(l)}V^{(l)T} \approx \mathcal{M}^{(l)}.$$

  Here Σ is a diagonal matrix containing the $\bar{p}$ most significant singular values of $\mathcal{M}^{(l)}$. Equality holds if $\bar{p} = p$.

Then $\Phi_{l,b}$ and $\Psi_{l,b}$ are $\bar{p}$-vectors representing the far-field and local expansions. The following replacements are made to Algorithm 2.3.1

$CIRC$	$CIRCSVD$
G_b	$U^{(h)T} \cdot G_b$
$M_L^{(l-1)}$	$U^{(l-1)T} \cdot M_L^{(l-1)} \cdot U^{(l)}$
$M_R^{(l-1)}$	$U^{(l-1)T} \cdot M_R^{(l-1)} \cdot U^{(l)}$
$S_L^{(l)}$	$V^{(l)T} \cdot S_L^{(l)} \cdot V^{(l-1)}$
$S_R^{(l)}$	$V^{(l)T} \cdot S_R^{(l)} \cdot V^{(l-1)}$
$T_1^{(l)}$	$V^{(l)T} \cdot T_1^{(l)} \cdot U^{(l)}$
$T_2^{(l)}$	$V^{(l)T} \cdot T_2^{(l)} \cdot U^{(l)}$
$T_3^{(l)}$	$V^{(l)T} \cdot T_3^{(l)} \cdot U^{(l)}$
$T_4^{(l)}$	$V^{(l)T} \cdot T_4^{(l)} \cdot U^{(l)}$
F	$V^{(2)T} \cdot F \cdot U^{(2)}$
E_b	$E_b \cdot V^{(h)}$

The number of flops used in $CIRCSVD_EVAL$ is obtained by replacing p in expression (2.44) with $\bar{p}$:

$$2\bar{p}(K+N) + (20s - 72)\bar{p}^2 + 2\sum_{b=1}^{s} |\mathcal{Y}_b|(|\mathcal{X}_{b-1}| + |\mathcal{X}_b| + |\mathcal{X}_{b+1}|). \tag{2.53}$$

Analogously to Section 2.4.3, if all s intervals at level h contain equal numbers of charges, and also equal numbers of evaluation points, then optimally

$$s = s_{\text{opt}} = \frac{1}{\bar{p}}\sqrt{\frac{3KN}{10}}. \tag{2.54}$$

2.5 Error in LIN for $1/x$ potential

This section gives estimates for the error in algorithm LIN when the potential is $\rho(x) = 1/x$.

In Section 2.5.1 we derive formulas for the *exact* error in approximating $\rho(y-x)$ by virtual-charge functions v_k in either far-field or local expansions. By interpolating these error functions in Section 2.5.2, we derive an estimate for the error in the overall LIN algorithm with a single charge and a single evaluation point. Finally,

Section 2.5.3 provides general error estimates for LIN.

2.5.1 Approximation error in expansions

The following theorem gives the error in a far-field expansion for the $1/x$ potential.

Theorem 2.5.1 *Suppose $p \geq 2$, $w > 0$, $|x| \leq w$, $|y| \geq 3w$. Then*

$$\frac{1}{y-x} - \sum_{j=1}^{p}\left(\frac{t_j}{3w - t_j x}\right)v_j\left(\frac{3w}{y}\right) = \left(\frac{1}{y-x}\right)\left(\frac{y}{x}\right)^p \frac{T_p(3w/y)}{T_p(3w/x)}\frac{T_{p-1}(x/w)}{T_{p-1}(y/w)}. \tag{2.55}$$

Proof. Let $z = 3w/y$. Then

$$\frac{1}{y-x} = \frac{z}{3w - xz}. \tag{2.56}$$

If we define

$$Q(z) = z\prod_{i=1}^{p-1}(z-\gamma_i) - (3w - xz)\sum_{j=i}^{p}\left(\frac{t_j}{3w - t_j x}\right)u_j(z)\prod_{i=1}^{p-1}(t_j - \gamma_i), \tag{2.57}$$

then Q is a polynomial of degree p in z.

For $k = 1, \ldots, p$,

$$Q(t_k) = t_k\prod_{i=1}^{p-1}(t_k - \gamma_i) - (3w - xt_k)\left(\frac{t_k}{3w - t_k x}\right)\prod_{i=1}^{p-1}(t_k - \gamma_i) = 0. \tag{2.58}$$

Thus the p zeroes of Q coincide with those of T_p. Also,

$$Q(3w/x) = \frac{3w}{x}\prod_{i=1}^{p-1}\left(\frac{3w}{x} - \gamma_i\right). \tag{2.59}$$

Since there is only one polynomial of degree p in z that satisfies the $p+1$ conditions (2.58)-(2.59), we have

$$Q(z) = \frac{3w}{x}\frac{T_p(z)}{T_p(3w/x)}\prod_{i=1}^{p-1}\left(\frac{3w}{x} - \gamma_i\right). \tag{2.60}$$

This gives

$$\frac{3w}{y}\prod_{i=1}^{p-1}(\frac{3w}{y}-\gamma_i)-\frac{3w}{y}(y-x)\sum_{j=1}^{p}(\frac{t_j}{3w-t_jx})u_j(\frac{3w}{y})\prod_{i=1}^{p-1}(t_j-\gamma_i)$$
$$=\frac{3w}{x}\frac{T_p(3w/y)}{T_p(3w/x)}\prod_{i=1}^{p-1}(\frac{3w}{x}-\gamma_i). \qquad (2.61)$$

Dividing equation (2.61) through by $(y-x)\frac{3w}{y}\prod_{i=1}^{p-1}(\frac{3w}{y}-\gamma_i)$, then

$$\frac{1}{y-x}-\sum_{j=1}^{p}(\frac{t_j}{3w-t_jx})v_j(\frac{3w}{y})=(\frac{1}{y-x})(\frac{y}{x})\frac{T_p(3w/y)}{T_p(3w/x)}\prod_{i=1}^{p-1}(\frac{3w/x-\gamma_i}{3w/y-\gamma_i}).$$

The result follows by Lemma 2.2.8. □

We use X_O and Y_O as defined in Equations 2.8.

Definition 2.5.2 *Let $p \geq 2$, $l \geq 2$, $|x| \leq w_l$, $|y| \geq 3w_l$, where $w_l = 1/2^l$. Then we define*

$$\begin{aligned}\mathrm{Errfar}_{1/x}(p,l,x,y) &= \rho(y-x)-\sum_{j=1}^{p}\rho(Y_O^{(l)}(t_j)-x)v_j(X_O^{(l)}(y))\\ &= (\frac{1}{y-x})(\frac{y}{x})^p\frac{T_p(3w_l/y)}{T_p(3w_l/x)}\frac{T_{p-1}(x/w_l)}{T_{p-1}(y/w_l)}\end{aligned}$$

which by Theorem 2.5.1 is the error in the far-field expansion of $1/(y-x)$ when using p terms.

Theorem 2.5.3 *Suppose $p \geq 2$, $l \geq 2$, $|x| \leq w_l$, $|y| \geq 3w_l$, where $w_l = 1/2^l$. Then*

$$|\mathrm{Errfar}_{1/x}(p,l,x,y)| < \frac{6+4\sqrt{2}}{(17/3+4\sqrt{2})^p}2^l$$

Proof. Since $|3w_l/y| \leq 1$ and $|x/w_l| \leq 1$, by Lemma 2.2.4 we have $|T_p(3w_l/y)| \leq 1$ and $|T_{p-1}(x/w_l)| \leq 1$.

Since $|3w_l/x| \geq 3$ and $|y/w_l| \geq 3$, we can also apply Lemma 2.2.5 to the expression in Definition 2.5.2, to obtain

$$\begin{aligned} |\mathrm{Errfar}_{1/x}(p,l,x,y)| &< |\frac{1}{y-x}||\frac{y}{x}|^p(\frac{2}{(1+\sqrt{8}/3)^p|3w_l/x|^p})(\frac{2}{(1+\sqrt{8}/3)^{p-1}|y/w_l|^{p-1}}) \\ &= |\frac{y}{y-x}|\frac{4}{3^p(1+\sqrt{8}/3)^{2p-1}w_l}. \end{aligned}$$

The result follows after using the inequality $|\frac{y}{y-x}| \leq \frac{3}{2}$. □

Theorem 2.5.4 gives the error in a local expansion for the $1/x$ potential.

Theorem 2.5.4 *Suppose $p \geq 2$, $w > 0$, $|x| \geq 3w$, $|y| \leq w$. Then*

$$\frac{1}{y-x} - \sum_{j=1}^{p}(\frac{1}{wt_j - x})v_j(\frac{y}{w}) = (\frac{1}{y-x})(\frac{x}{y})^{p-1}\frac{T_p(y/w)}{T_p(x/w)}\frac{T_{p-1}(3w/x)}{T_{p-1}(3w/y)}. \tag{2.62}$$

Proof. Let

$$Q(y) = \prod_{i=1}^{p-1}(\frac{y}{w} - \gamma_i) - (y-x)\sum_{j=1}^{p}(\frac{1}{wt_j - x})u_j(\frac{y}{w})\prod_{i=1}^{p-1}(t_j - \gamma_i), \tag{2.63}$$

Then Q is a polynomial of degree p in y.

For $k = 1, \ldots, p$,

$$Q(wt_k) = \prod_{i=1}^{p-1}(t_k - \gamma_i) - (wt_k - x)(\frac{1}{wt_k - x})\prod_{i=1}^{p-1}(t_k - \gamma_i) = 0. \tag{2.64}$$

Thus the p zeroes of Q coincide with those of T_p. Also,

$$Q(x) = \prod_{i=1}^{p-1}(\frac{x}{w} - \gamma_i). \tag{2.65}$$

Since there is only one polynomial of degree p in y that satisfies the $p+1$ conditions (2.64)-(2.65), we have

$$Q(y) = \frac{T_p(y/w)}{T_p(x/w)}\prod_{i=1}^{p-1}(\frac{x}{w} - \gamma_i). \tag{2.66}$$

Dividing equation (2.66) through by $(y-x)\prod_{i=1}^{p-1}(\frac{y}{w}-\gamma_i)$ yields

$$\frac{1}{y-x}-\sum_{j=1}^{p}(\frac{1}{wt_j-x})v_j(\frac{y}{w})=(\frac{1}{y-x})\frac{T_p(y/w)}{T_p(x/w)}\prod_{i=1}^{p-1}(\frac{x/w-\gamma_i}{y/w-\gamma_i}).$$

The result follows by Lemma 2.2.8. □

We use X_I and Y_I as defined in Equations 2.7.

Definition 2.5.5 *Let $p \geq 2$, $l \geq 2$, $|x| \geq 3w_l$, $|y| \leq w_l$, where $w_l = 1/2^l$. Then we define*

$$\begin{aligned}\mathrm{Errloc}_{1/x}(p,l,x,y) &= \rho(y-x)-\sum_{j=1}^{p}\rho(Y_I^{(l)}(t_j)-x)v_j(X_I^{(l)}(y)) \\ &= (\frac{1}{y-x})(\frac{x}{y})^{p-1}\frac{T_p(y/w_l)}{T_p(x/w_l)}\frac{T_{p-1}(3w_l/x)}{T_{p-1}(3w_l/y)}.\end{aligned}$$

which by Theorem 2.5.4 is the error in the local expansion of $1/(y-x)$ when using p terms.

Theorem 2.5.6 *Suppose $p \geq 2$, $l \geq 2$, $|x| \geq 3w_l$, $|y| \leq w_l$, where $w_l = 1/2^l$. Then*

$$|\mathrm{Errloc}_{1/x}(p,l,x,y)| < \frac{2+4\sqrt{2}/3}{(17/3+4\sqrt{2})^p}2^l$$

Proof. Since $|y/w_l| \leq 1$ and $|3w_l/x| \leq 1$, by Lemma 2.2.4 we have $|T_p(y/w_l)| \leq 1$ and $|T_{p-1}(3w_l/x)| \leq 1$.

Since $|x/w_l| \geq 3$ and $|3w_l/y| \geq 3$, we can also apply Lemma 2.2.5 to the expression in Definition 2.5.5, to obtain

$$\begin{aligned}&|\mathrm{Errloc}_{1/x}(p,l,x,y)| \\ &< |\frac{1}{y-x}||\frac{x}{y}|^{p-1}(\frac{2}{(1+\sqrt{8}/3)^p|x/w_l|^p})(\frac{2}{(1+\sqrt{8}/3)^{p-1}|3w_l/y|^{p-1}}) \\ &= \frac{4w_l}{|x(y-x)|3^{p-1}(1+\sqrt{8}/3)^{2p-1}}.\end{aligned}$$

The result follows after using the inequality $|x(y-x)| \geq 6w_l^2$. □

2.5.2 Overall error estimates for a single charge

We consider problem (2.1) on the interval $[-1,1]$ with potential $\rho(x) = 1/x$. Then $w_l = 1/2^l$, and $X_I^{(l)}$, $Y_I^{(l)}$, $X_O^{(l)}$ and $Y_O^{(l)}$ are as in Equations (2.7)-(2.8).

Assume we use the algorithm with h levels. In the remainder of this section we fix the following:

- x is the position of a unit charge on the interval $[-1,1]$.
- y is another point on $[-1,1]$, where LIN evaluates the potential due to the charge at x.
- At each level $l = 1,\ldots h$, the interval $[-1,1]$ is divided into 2^l subintervals of length w_l.
  - ◇ ξ_l is the center of the subinterval at level l containing x. Note that $|x-\xi_l| \leq w_l$ and $\xi_l - \xi_{l-1} = \pm w_l$.
  - ◇ η_l is the center of the subinterval at level l containing y.

  Note that $|y - \eta_l| \leq w_l$ and $\eta_l - \eta_{l-1} = \pm w_l$.

  Also, $\xi_l - \eta_l$ is a multiple of $2w_l$.

Lemma 2.5.7 *If $|\xi_h - \eta_h| \leq 2w_h$ then $LIN(p,h,x,y,1)$ returns $\rho(y-x)$.*

Otherwise, let d be the greatest number such that $|\xi_{d-1} - \eta_{d-1}| = 2w_{d-1}$.

Let $\mathbf{g}_1,\ldots,\mathbf{g}_h$ and $\mathbf{c}_1,\ldots,\mathbf{c}_h$ be vectors of length p giving the coefficients of the far-field and local expansions, respectively, for a charge at point x:

$$\begin{aligned} \mathbf{g}_l(k) &= \rho(Y_O^{(l)}(t_k) - (x - \xi_l)) \\ \mathbf{c}_l(k) &= \rho(Y_I^{(l)}(t_k) - (x - \eta_l)) \end{aligned}$$

Let $\mathbf{f}_1,\ldots,\mathbf{f}_h$ and $\mathbf{e}_1,\ldots,\mathbf{e}_h$ be vectors of length p giving the evaluation weights of the far-field and local expansions, respectively, for potential at point y:

$$\mathbf{f}_l(k) = v_k(X_O^{(l)}(y - \xi_l))$$

$$\mathbf{e}_l(k) \;=\; v_k(X_I^{(l)}(y-\eta_l))$$

For each $l = d, \ldots, h-1$, let M_l be the the $p \times p$ matrix with entries

$$M_l(j,k) = v_k(X_O^{(l+1)}(Y_O^{(l)}(t_j) + \xi_l - \xi_{l+1})). \tag{2.67}$$

Let T be the $p \times p$ matrix with entries

$$T(j,k) = v_k(X_O^{(d)}(Y_I^{(d)}(t_j) + \eta_d - \xi_d)). \tag{2.68}$$

For each $l = d+1, \ldots, h$, let S_l be the the $p \times p$ matrix with entries

$$S_l(j,k) = v_k(X_I^{(l-1)}(Y_I^{(l)}(t_j) + \eta_l - \eta_{l-1})). \tag{2.69}$$

Then $LIN(p,h,x,y,1)$ returns

$$\mathbf{e}_h^T S_h \cdots S_{d+1} T M_d \cdots M_{h-1} \mathbf{g}_h. \tag{2.70}$$

Proof.

If $|\xi_h - \eta_h| \leq 2w_h$ then x and y are located in either the same box or in adjacent boxes at level h. Therefore the algorithm evaluates the potential directly.

Otherwise, we use LIN_RECUR, go up the tree and do a flip at the lowest level (highest number) d at which x and y are in adjacent boxes at level $d-1$. Note that x and y are in adjacent boxes at level $d-1$ if and only if $|\xi_{d-1} - \eta_{d-1}| = 2w_{d-1}$.

We begin with $\mathbf{g}_h$, the coefficients for the far-field expansion due to the charge at x.

Moving up the tree, the algorithm uses the following operator to shift a far-field expansion at level $l+1$ to one at level l:

$$\begin{aligned} M_l(j,k) \;&=\; \begin{cases} M_L^{(l)}(j,k) = v_k(X_O^{(l+1)}(Y_O^{(l)}(t_j) + w_{l+1})), & \text{if } \xi_l = \xi_{l+1} + w_{l+1}; \\ M_R^{(l)}(j,k) = v_k(X_O^{(l+1)}(Y_O^{(l)}(t_j) - w_{l+1})), & \text{if } \xi_l = \xi_{l+1} - w_{l+1}; \end{cases} \\ &=\; v_k(X_O^{(l+1)}(Y_O^{(l)}(t_j) + \xi_l - \xi_{l+1})). \end{aligned}$$

At level d, the algorithm flips from far-field to local expansions, using the following operator:

$$
\begin{aligned}
T(j,k) &= \begin{cases} T_1^{(d)}(j,k) = v_k(X_O^{(d)}(Y_I^{(d)}(t_j) + 6w_d)), & \text{if } \eta_d = \xi_d + 6w_d; \\ T_2^{(d)}(j,k) = v_k(X_O^{(d)}(Y_I^{(d)}(t_j) + 4w_d)), & \text{if } \eta_d = \xi_d + 4w_d; \\ T_3^{(d)}(j,k) = v_k(X_O^{(d)}(Y_I^{(d)}(t_j) - 4w_d)), & \text{if } \eta_d = \xi_d - 4w_d; \\ T_4^{(d)}(j,k) = v_k(X_O^{(d)}(Y_I^{(d)}(t_j) - 6w_d)), & \text{if } \eta_d = \xi_d - 6w_d; \end{cases} \\
&= v_k(X_O^{(d)}(Y_I^{(d)}(t_j) + \eta_d - \xi_d)).
\end{aligned}
$$

Moving down the tree, the algorithm uses the following operator to shift a far-field expansion at level $l-1$ to one at level l:

$$
\begin{aligned}
S_l(j,k) &= \begin{cases} S_L^{(l)}(j,k) = v_k(X_I^{(l-1)}(Y_I^{(l)}(t_j) - w_l)), & \text{if } \eta_l = \eta_{l-1} - w_l; \\ S_R^{(l)}(j,k) = v_k(X_I^{(l-1)}(Y_I^{(l)}(t_j) + w_l)), & \text{if } \eta_l = \eta_{l-1} + w_l; \end{cases} \\
&= v_k(X_I^{(l-1)}(Y_I^{(l)}(t_j) + \eta_l - \eta_{l-1})).
\end{aligned}
$$

Finally, the algorithm multiplies by $\mathbf{e}_l^T$ to evaluate the local expansion at x. □

The remainder of this section uses the approximation

$$
\sum_{j=1}^{p} v_j(z) f(t_j) \approx f(z) \tag{2.71}
$$

for functions f defined on $[-1,1]$. Here $v_1, \ldots, v_p$ are the rational functions from Definition 2.2.7.

Estimate 2.5.8 *If $|\xi_h - \eta_h| > 2w_h$ then the error in LIN is given by*

$$
\begin{aligned}
\frac{1}{y-x} &- LIN(p,h,x,y,1) \\
&\approx \sum_{l=d}^{h} (\mathrm{Errloc}_{1/x}(p,l,x-\eta_l,y-\eta_l) + \mathrm{Errfar}_{1/x}(p,l,x-\xi_l,y-\xi_l))
\end{aligned}
$$

where $\mathrm{Errloc}_{1/x}$ *and* $\mathrm{Errfar}_{1/x}$ *are as in Definitions 2.5.5 and 2.5.2.*

Derivation. Using the vectors and operators defined in Lemma 2.5.7, we can write

the expression for the exact error as a telescoping sum:

$$
\begin{aligned}
\rho(y-x) &- LIN(p,h,x,y,1) \\
&= \rho(y-x) - \mathbf{e}_h^T S_h \cdots S_{d+1} T M_d \cdots M_{h-1}\mathbf{g}_h \\
&= \rho(y-x) - \mathbf{e}_h^T \mathbf{c}_h \qquad (2.72) \\
&\quad + \sum_{l=d+1}^{h} \mathbf{e}_h^T S_h \cdots S_{l+1}(\mathbf{c}_l - S_l \mathbf{c}_{l-1}) \\
&\quad + \mathbf{e}_h^T S_h \cdots S_{d+1}(\mathbf{c}_d - T\mathbf{g}_d) \\
&\quad + \sum_{l=d+1}^{h} \mathbf{e}_h^T S_h \cdots S_{d+1} T M_d \cdots M_{l-2}(\mathbf{g}_{l-1} - M_{l-1}\mathbf{g}_l)
\end{aligned}
$$

We derive separate error estimates for each term, and then take the sum.

- $\rho(y-x) - \mathbf{e}_h^T \mathbf{c}_h$:

$$
\begin{aligned}
\rho(y-x) - \mathbf{e}_h^T \mathbf{c}_h &= \rho(y-x) - \sum_{k=1}^{p} v_k(X_I^{(h)}(y-\eta_h))\rho(Y_I^{(h)}(t_k) - (x-\eta_h)) \\
&= \mathrm{Errloc}_{1/x}(p,h,x-\eta_h,y-\eta_h). \qquad (2.73)
\end{aligned}
$$

This estimate is exact.

- $\mathbf{e}_h^T S_h \cdots S_{l+1}(\mathbf{c}_l - S_l \mathbf{c}_{l-1})$, for $l = d+1, d+2, \ldots, h$:

The kth component of $\mathbf{c}_l - S_l \mathbf{c}_{l-1}$ is

$$
\begin{aligned}
&\rho(Y_I^{(l)}(t_k) - (x-\eta_l)) - \\
&\sum_{j=1}^{p} v_j(X_I^{(l-1)}(Y_I^{(l)}(t_k) + \eta_l - \eta_{l-1}))\rho(Y_I^{(l-1)}(t_j) - (x-\eta_{l-1})) \\
&\quad = \mathrm{Errloc}_{1/x}(p, l-1, x-\eta_{l-1}, Y_I^{(l)}(t_k) + \eta_l - \eta_{l-1}).
\end{aligned}
$$

So the kth component of $S_{l+1}(\mathbf{c}_l - S_l \mathbf{c}_{l-1})$ is

$$
\begin{aligned}
&\sum_{j=1}^{p} v_j(X_I^{(l)}(Y_I^{(l+1)}(t_k) + \eta_{l+1} - \eta_l)) \cdot \\
&\mathrm{Errloc}_{1/x}(p, l-1, x-\eta_{l-1}, Y_I^{(l)}(t_k) + \eta_l - \eta_{l-1})
\end{aligned}
$$

$$\approx \quad \mathrm{Errloc}_{1/x}(p, l-1, x-\eta_{l-1}, Y_I^{(l+1)}(t_k)+\eta_{l+1}-\eta_{l-1}),$$

using Approximation 2.71. Continuing these approximations, then the kth component of $S_h \cdots S_{l+1}(\mathbf{c}_l - S_l\mathbf{c}_{l-1})$ is estimated as

$$\mathrm{Errloc}_{1/x}(p, l-1, x-\eta_{l-1}, Y_I^{(h)}(t_k)+\eta_h-\eta_{l-1}).$$

Hence

$$\begin{aligned}
& \mathbf{e}_h^T S_h \cdots S_{l+1}(\mathbf{c}_l - S_l\mathbf{c}_{l-1}) \\
= \quad & \sum_{k=1}^{p} v_k(X_I^{(h)}(y-\eta_h))\mathrm{Errloc}_{1/x}(p, l-1, x-\eta_{l-1}, Y_I^{(h)}(t_k)+\eta_h-\eta_{l-1}) \\
\approx \quad & \mathrm{Errloc}_{1/x}(p, l-1, x-\eta_{l-1}, y-\eta_{l-1}). \qquad (2.74)
\end{aligned}$$

- $\mathbf{e}_h^T S_h \cdots S_{d+1}(\mathbf{c}_d - T\mathbf{g}_d)$:

  The kth component of $\mathbf{c}_d - T\mathbf{g}_d$ is

$$\begin{aligned}
& \rho(Y_I^{(d)}(t_k)-(x-\eta_d))- \\
& \sum_{j=1}^{p} v_j(X_O^{(d)}(Y_I^{(d)}(t_k)+\eta_d-\xi_d))\rho(Y_O^{(d)}(t_j)-(x-\xi_d)) \\
& \quad = \quad \mathrm{Errfar}_{1/x}(p, d, x-\xi_d, Y_I^{(d)}(t_k)+\eta_d-\xi_d).
\end{aligned}$$

So the kth component of $S_{d+1}(\mathbf{c}_d - T\mathbf{g}_d)$ is

$$\begin{aligned}
& \sum_{j=1}^{p} v_j(X_I^{(d)}(Y_I^{(d+1)}(t_k)+\eta_{d+1}-\eta_d))\mathrm{Errfar}_{1/x}(p, d, x-\xi_d, Y_I^{(d)}(t_k)+\eta_d-\xi_d) \\
& \qquad \approx \mathrm{Errfar}_{1/x}(p, d, x-\xi_d, Y_I^{(d+1)}(t_k)+\eta_{d+1}-\xi_d),
\end{aligned}$$

using Approximation 2.71. Continuing these approximations, then the kth component of $S_h \cdots S_{d+1}(\mathbf{c}_d - T\mathbf{g}_d)$ is estimated as

$$\mathrm{Errfar}_{1/x}(p, d, x-\xi_d, Y_I^{(h)}(t_k)+\eta_h-\xi_d).$$

Hence

$$
\begin{aligned}
& \mathbf{e}_h^T S_h \cdots S_{l+1}(\mathbf{c}_d - T\mathbf{c}_d) \\
= & \sum_{k=1}^{p} v_k(X_I^{(h)}(y-\eta_h))\mathrm{Errfar}_{1/x}(p,d,x-\xi_d, Y_I^{(h)}(t_k)+\eta_h-\xi_d) \\
\approx & \ \mathrm{Errfar}_{1/x}(p,d,x-\xi_d, y-\xi_d). \qquad (2.75)
\end{aligned}
$$

- $\mathbf{e}_h^T S_h \cdots S_{d+1} T M_d \cdots M_{l-2}(\mathbf{g}_{l-1} - M_{l-1}\mathbf{g}_l)$, for $l = d+1, d+2, \ldots, h$:

  The kth component of $\mathbf{g}_{l-1} - M_{l-1}\mathbf{g}_l$ is

$$
\begin{aligned}
& \rho(Y_O^{(l-1)}(t_k) - (x-\xi_{l-1})) - \\
& \sum_{j=1}^{p} v_j(X_O^{(l)}(Y_O^{(l-1)}(t_k)+\xi_{l-1}-\xi_l))\rho(Y_O^{(l)}(t_j)-(x-\xi_l)) \\
= & \ \mathrm{Errfar}_{1/x}(p,l,x-\xi_l, Y_O^{(l-1)}(t_k)+\xi_{l-1}-\xi_l).
\end{aligned}
$$

  So the kth component of $M_{l-2}(\mathbf{g}_{l-1} - M_{l-1}\mathbf{g}_l)$ is

$$
\begin{aligned}
& \sum_{j=1}^{p} v_j(X_O^{(l-1)}(Y_O^{(l-2)}(t_k)+\xi_{l-2}-\xi_{l-1})) \cdot \\
& \mathrm{Errfar}_{1/x}(p,l,x-\xi_l, Y_O^{(l-1)}(t_k)+\xi_{l-1}-\xi_l) \\
\approx & \ \mathrm{Errfar}_{1/x}(p,l,x-\xi_l, Y_O^{(l-2)}(t_k)+\xi_{l-2}-\xi_l),
\end{aligned}
$$

  using Approximation 2.71. Continuing these approximations, then the kth component of $M_d \cdots M_{l-2}(\mathbf{g}_{l-1} - M_{l-1}\mathbf{g}_l)$ is estimated as

$$
\mathrm{Errfar}_{1/x}(p,l,x-\xi_l, Y_O^{(d)}(t_k)+\xi_d-\xi_l).
$$

  The kth component of $TM_d \cdots M_{l-2}(\mathbf{g}_{l-1} - M_{l-1}\mathbf{g}_l)$ is estimated to be

$$
\begin{aligned}
& \sum_{j=1}^{p} v_j(X_O^{(d)}(Y_I^{(d)}(t_k)+\eta_d-\xi_d))\mathrm{Errfar}_{1/x}(p,l,x-\xi_l, Y_O^{(d)}(t_j)+\xi_d-\xi_l) \\
\approx & \ \mathrm{Errfar}_{1/x}(p,l,x-\xi_l, Y_I^{(d)}(t_k)+\eta_d-\xi_l)
\end{aligned}
$$

The kth component of $S_{d+1}TM_d \cdots M_{l-2}(\mathbf{g}_{l-1} - M_{l-1}\mathbf{g}_l)$ is then estimated to be

$$\begin{aligned}
&\sum_{j=1}^{p} v_j(X_I^{(d)}(Y_I^{(d+1)}(t_k) + \eta_{d+1} - \eta_d))\mathrm{Errfar}_{1/x}(p, l, x - \xi_l, Y_I^{(d)}(t_j) + \eta_d - \xi_l) \\
&\approx \mathrm{Errfar}_{1/x}(p, l, x - \xi_l, Y_I^{(d+1)}(t_k) + \eta_{d+1} - \xi_l)
\end{aligned}$$

Continuing, then the kth component of $S_h \cdots S_{d+1}TM_d \cdots M_{l-2}(\mathbf{g}_{l-1} - M_{l-1}\mathbf{g}_l)$ is estimated to be

$$\mathrm{Errfar}_{1/x}(p, l, x - \xi_l, Y_I^{(h)}(t_k) + \eta_h - \xi_l).$$

Hence

$$\begin{aligned}
&\mathbf{e}_h^T S_h \cdots S_{d+1}TM_d \cdots M_{l-2}(\mathbf{g}_{l-1} - M_{l-1}\mathbf{g}_l) \\
= &\sum_{k=1}^{p} v_k(X_I^{(h)}(y - \eta_h))\mathrm{Errfar}_{1/x}(p, l, x - \xi_l, Y_I^{(h)}(t_k) + \eta_h - \xi_l) \\
\approx &\mathrm{Errfar}_{1/x}(p, l, x - \xi_l, y - \xi_l). \qquad (2.76)
\end{aligned}$$

The estimate follows from adding up Equations (2.73)–(2.76). □

2.5.3 General error estimate for LIN

Estimate 2.5.9 *The absolute value of the error in calculating $1/(y - x)$ is*

$$\left|\frac{1}{y - x} - LIN(p, h, x, y, 1)\right| \lesssim \frac{8(2 + \sqrt{2}/3)}{(17/3 + 4\sqrt{2})^p} \cdot 2^h$$

Derivation. Using Estimate 2.5.8:

$$\begin{aligned}
\frac{1}{y - x} &- \mathbf{e}_h^T S_h \cdots S_{h+1}TM_d \cdots M_{h-1}\mathbf{g}_h \\
&\approx \sum_{l=d}^{h}(\mathrm{Errloc}(p, l, x - \eta_l, y - \eta_l) + \mathrm{Errfar}(p, l, x - \xi_l, y - \xi_l)).
\end{aligned}$$

Using the bounds from Theorems 2.5.3 and 2.5.6, this sum is at most

$$\begin{aligned} & \frac{2+4\sqrt{2}/3}{(17/3+4\sqrt{2})^p}(2^d+\cdots+2^h)+\frac{6+4\sqrt{2}}{(17/3+4\sqrt{2})^p}(2^d+\cdots+2^h) \\ = & \frac{4(2+4\sqrt{2}/3)}{(17/3+4\sqrt{2})^p}(2^d+\cdots+2^h) < \frac{8(2+4\sqrt{2}/3)}{(17/3+4\sqrt{2})^p}2^h. \end{aligned}$$

□

Figure 2-3 shows a sample plot of the error in calculating $1/(y-x)$ when $x = -1$ and $y \in [-1,1]$. Here we use expansions of size $p = 10$ and $h = 8$ levels. Estimate 2.5.8 gives an error that is within 5×10^{-10} of the observed error. The bound in Estimate 2.5.9 is also within a factor of 2 of the observed maximum error.

Estimate 2.5.10 *If*

$$f_j = \sum_{k=1}^{K} \frac{q_k}{y_j - x_k}$$

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

$$\tilde{\mathbf{f}} = LIN(p, h, \mathbf{x}, \mathbf{y}, \mathbf{q})$$

has components $\tilde{f}_1, \ldots, \tilde{f}_N$, *then*

$$|f_j - \tilde{f}_j| \lesssim (\sum_{k=1}^{K} |q_k|) \frac{8(2+\sqrt{2}/3)}{(17/3+4\sqrt{2})^p} \cdot 2^h.$$

Derivation. Use Estimate 2.5.9 and the fact that $LIN(p, h, \mathbf{x}, \mathbf{y}, \mathbf{q})$ is linear in $\mathbf{q}$.

□

Estimate 2.5.11 *If* $\mathbf{f}$ *has components*

$$f_j = \sum_{k=1}^{K} \frac{q_k}{y_j - x_k}$$

for $j = 1, \ldots, N$, *then*

$$\|\mathbf{f} - LIN(p, h, \mathbf{x}, \mathbf{y}, \mathbf{q})\|_2 \lesssim \|\mathbf{q}\|_2 \sqrt{KN} \cdot \frac{8(2+\sqrt{2}/3)}{(17/3+4\sqrt{2})^p} \cdot 2^h.$$

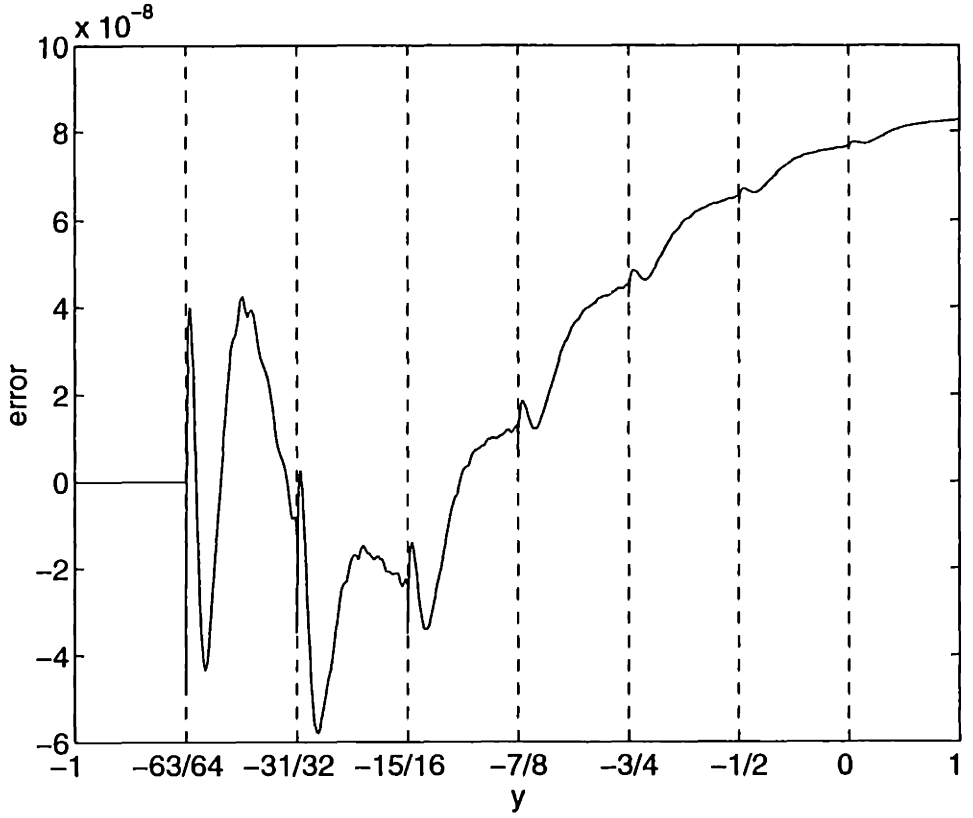


Figure 2-3: Value of the error $1/(y+1) - LIN(10, 8, -1, y, 1)$. Between each pair of dashed vertical lines, the horizontal axis is scaled linearly with a different scaling. The maximum absolute difference between the exact error plotted here and Estimate 2.5.8 is 5×10^{-10}, which is too small to be distinguishable in their graphs. Also note that Estimate 2.5.9 gives a bound of 1.5×10^{-7} on the absolute error. This bound is within a factor of 2 of the observed maximum error.

Derivation. Defining the matrices

$$V_{jk} = \frac{1}{y_j - x_k}$$

and

$$\widetilde{V}_{jk} = LIN(p, h, x_k, y_j, 1),$$

then

$$LIN(p, h, \mathbf{x}, \mathbf{y}, \mathbf{q}) = \widetilde{V}\mathbf{q}$$

because LIN is linear in $\mathbf{q}$. The 2-norm of the difference between V and $\widetilde{V}$ is bounded

by the Frobenius norm:

$$\|V - \widetilde{V}\|_2 \leq \|V - \widetilde{V}\|_F \leq \sqrt{KN} \cdot \max_{j,k} |V_{jk} - \widetilde{V}_j k|.$$

The result follows from Estimate 2.5.9. □

The estimates in this section are for the worst-case error, which doubles with each successive level in LIN. As Figure 2-3 illustrates, with our interpolation functions, the error for a single charge and evaluation point is roughly uniform across the possible relative positions.

2.6 Error in $CIRC$ for $\cot(\theta/2)$ potential

This section gives estimates for the error in algorithm $CIRC$ when it is used with the potential $\rho(\theta) = \cot(\theta/2)$.

In Section 2.6.1 we derive formulas for the *exact* error in approximating $\rho(y - x)$ by virtual-charge functions v_k in either far-field or local expansions. These result from the mappings (2.38)–(2.39) between the circle and the real line. In Section 2.6.2, we derive an estimate for the error in the overall $CIRC$ algorithm with a single charge and a single evaluation point after observing that the analysis of Section 2.5.2 carries through to the circular case with only minor changes. Finally, Section 2.6.3 provides general error estimates for $CIRC$.

2.6.1 Approximation error in expansions

We convert between cotangent functions and rational functions by means of the following two lemmas.

Lemma 2.6.1 *Let* $w > 0$, $c = 3\tan(w/2)$, $x = c\cot(\phi/2)$, $y = c\cot(\theta/2)$. *Then*

$$\cot(\frac{\theta - \phi}{2}) = \frac{xy/c + c}{x - y} \tag{2.77}$$

Proof. Using the subtraction formula for the cotangent function,

$$\cot(\frac{\theta-\phi}{2}) = \frac{1+\cot(\theta/2)\cot(\phi/2)}{\cot(\phi/2)-\cot(\theta/2)} = \frac{1+(y/c)(x/c)}{x/c-y/c} = \frac{xy/c+c}{x-y}.$$

□

Lemma 2.6.2 *Let* $w > 0$, $d = -\cot(w/2)$, $x = -d\tan(\phi/2)$, $y = -d\tan(\theta/2)$. *Then*

$$\cot(\frac{\theta-\phi}{2}) = \frac{xy/d+d}{x-y} \tag{2.78}$$

Proof. Using the subtraction formula for the cotangent function,

$$\cot(\frac{\theta-\phi}{2}) = \frac{1+\cot(\theta/2)\cot(\phi/2)}{\cot(\phi/2)-\cot(\theta/2)} = \frac{1+(-d/y)(-d/x)}{(-d/x)-(-d/y)} = \frac{xy/d+d}{x-y}.$$

□

The following lemma will be used to find error in cotangent expansions.

Lemma 2.6.3 *Let* $p \geq 2$, $|x| \geq 3$, $|y| \leq 1$, $c \neq 0$. *Then*

$$\frac{xy/c+c}{x-y} - \sum_{j=1}^{p} \frac{xt_j/c+c}{x-t_j} v_j(y) = (\frac{x^2/c+c}{x-y})(\frac{x}{y})^{p-1} \frac{T_p(y)}{T_p(x)} \frac{T_{p-1}(3/x)}{T_{p-1}(3/y)} \tag{2.79}$$

Proof. If we define

$$Q(y) = (xy/c+c)\prod_{i=1}^{p-1}(y-\gamma_i) - (x-y)\sum_{j=1}^{p}(\frac{xt_j/c+c}{x-t_j})u_j(y)\prod_{i=1}^{p-1}(t_j-\gamma_i),$$

then Q is a polynomial of degree p in y.

For $k = 1, \ldots, p$,

$$Q(t_k) = (xt_k/c+c)\prod_{i=1}^{p-1}(t_k-\gamma_i) - (x-t_k)(\frac{xt_k/c+c}{x-t_k})\prod_{i=1}^{p-1}(t_k-\gamma_i) = 0. \tag{2.80}$$

Thus the p zeroes of Q coincide with those of T_p. Also,

$$Q(x) = (x^2/c+c)\prod_{i=1}^{p-1}(x-\gamma_i). \tag{2.81}$$

Since there is only one polynomial of degree p in y that satisfies the $p+1$ conditions (2.80)-(2.81), we have

$$Q(y) = \frac{T_p(y)}{T_p(x)} \prod_{i=1}^{p-1} (x - \gamma_i). \tag{2.82}$$

Dividing equation (2.82) through by $(y-x)\prod_{i=1}^{p-1}(y-\gamma_i)$ yields

$$\sum_{j=1}^{p} \frac{xt_j/c + c}{x - t_j} v_j(y) = \left(\frac{x^2/c + c}{x - y}\right) \frac{T_p(y)}{T_p(x)} \prod_{i=1}^{p-1} \left(\frac{x - \gamma_i}{y - \gamma_i}\right).$$

The result follows by Lemma 2.2.8. □

The following theorem gives the error in far-field expansion for $\cot(\theta/2)$.

Theorem 2.6.4 *Let $w > 0$, $c = 3\tan(w/2)$, $d = -\cot(w/2)$, $|\phi| \le w$, $|\theta| \ge 3w$. Then*

$$\begin{aligned} \cot\left(\frac{\theta - \phi}{2}\right) &- \sum_{j=1}^{p} \left(\frac{t_j + c\tan(\phi/2)}{c - t_j \tan(\phi/2)}\right) v_j(c\cot(\theta/2)) \\ &= \frac{\csc^2(\phi/2)}{\cot(\phi/2) - \cot(\theta/2)} \left(\frac{\cot(\phi/2)}{\cot(\theta/2)}\right)^{p-1} \frac{T_p(c\cot(\theta/2))}{T_p(c\cot(\phi/2))} \frac{T_{p-1}(-d\tan(\phi/2))}{T_{p-1}(-d\tan(\theta/2))}. \end{aligned}$$

Proof. Let $x = c\cot(\phi/2)$ and $y = c\cot(\theta/2)$. Then $|x| \ge 3$ and $|y| \le 1$, and so by Lemmas 2.6.1 and 2.6.3,

$$\cot\left(\frac{\theta - \phi}{2}\right) - \sum_{j=1}^{p} \frac{xt_j/c + c}{x - t_j} v_j(c\cot(\theta/2)) = \left(\frac{x^2/c + c}{x - y}\right)\left(\frac{x}{y}\right)^{p-1} \frac{T_p(y)}{T_p(x)} \frac{T_{p-1}(3/x)}{T_{p-1}(3/y)}. \tag{2.83}$$

Here

$$\frac{xt_j/c + c}{x - t_j} = \frac{t_j \cot(\phi/2) + c}{c\cot(\phi/2) - t_j} = \frac{t_j + c\tan(\phi/2)}{c - t_j \tan(\phi/2)}, \tag{2.84}$$

and

$$\frac{x^2/c + c}{x - y} = \frac{c\cot^2(\phi/2) + c}{c\cot(\phi/2) - c\cot(\theta/2)} = \frac{\csc^2(\phi/2)}{\cot(\phi/2) - \cot(\theta/2)}.$$

□

We shall use X_O and Y_O as defined in Equations 2.39.

Definition 2.6.5 *Let $p \geq 2$, $l \geq 2$, $|\phi| \leq w_l$, $|\theta| \geq 3w_l$, where $w_l = \pi/2^l$. Then we define*

$$\begin{aligned}
\mathrm{Errfar}_{\cot(x/2)}(p,l,\phi,\theta) &= \rho(\theta-\phi) - \sum_{j=1}^{p} \rho(Y_O^{(l)}(t_j)-\phi)v_j(X_O^{(l)}(\theta)) \\
&= \frac{\csc^2(\phi/2)}{\cot(\phi/2)-\cot(\theta/2)}\left(\frac{\cot(\phi/2)}{\cot(\theta/2)}\right)^{p-1} \cdot \\
&\quad \frac{T_p(3\tan(w_l/2)/\tan(\theta/2))}{T_p(3\tan(w_l/2)/\tan(\phi/2))}\frac{T_{p-1}(\tan(\phi/2)/\tan(w_l/2))}{T_{p-1}(\tan(\theta/2)/\tan(w_l/2))}.
\end{aligned}$$

which by Theorem 2.6.4 is the error in the far-field expansion of $\cot((\theta-\phi)/2)$ *when using p terms.*

Theorem 2.6.6 *Suppose $p \geq 2$, $l \geq 2$, $|\phi| \leq w_l$, $|\theta| \geq 3w_l$, where $w_l = \pi/2^l$. Then*

$$|\mathrm{Errfar}_{\cot(x/2)}(p,l,\phi,\theta)| < \frac{12+8\sqrt{2}}{(17/3+4\sqrt{2})^p}\csc(w_l)$$

Proof. Letting $c = 3\tan(w_l/2)$, $x = c/\tan(\phi/2)$, and $y = c/\tan(\theta/2)$, then

$$\mathrm{Errfar}_{\cot(x/2)}(p,l,\phi,\theta) = \frac{(x/c)^2+1}{x/c-y/c}\left(\frac{x}{y}\right)^{p-1}\frac{T_p(y)}{T_p(x)}\frac{T_{p-1}(3/x)}{T_{p-1}(3/y)} \tag{2.85}$$

Since $|y| \leq 1$ and $|3/x| \leq 1$, by Lemma 2.2.4 we have $|T_p(y)| \leq 1$ and $|T_{p-1}(3/x)| \leq 1$.

Since $|x| \geq 3$ and $|3/y| \geq 3$, we can also apply Lemma 2.2.5 to the expression in Definition 2.6.5, to obtain

$$\begin{aligned}
|\mathrm{Errfar}_{\cot(x/2)}(p,l,\phi,\theta)| &= \frac{x^2/c+c}{x-y}\left(\frac{x}{y}\right)^{p-1}\frac{T_p(y)}{T_p(x)}\frac{T_{p-1}(3/x)}{T_{p-1}(3/y)} \\
&< \left|\frac{x^2/c+c}{x-y}\right|\left|\frac{x}{y}\right|^{p-1}\frac{2}{((1+\sqrt{8}/3)|x|)^p}\frac{2}{((1+\sqrt{8}/3)|3/y|)^{p-1}} \\
&= \left|\frac{1/c+c/x^2}{1-y/x}\right|\frac{4}{(1+\sqrt{8}/3)^{2p-1}3^{p-1}} \\
&< \left|\frac{1}{c}+\frac{c}{x^2}\right|\frac{6}{(1+\sqrt{8}/3)^{2p-1}3^{p-1}},
\end{aligned} \tag{2.86}$$

where the last inequality follows from $|1 - y/x| \geq \frac{2}{3}$. The result follows from

$$1/c + c/x^2 \leq 1/c + c/9 = \frac{1 + c^2/9}{c} = \frac{1}{3\sin(w_l)\cos(w_l)} = \frac{2}{3}\csc(w_l).$$

□

Theorem 2.6.7 gives the error in local expansion for $\cot(\theta/2)$.

Theorem 2.6.7 *Let* $w > 0$, $c = 3\tan(w/2)$, $d = -\cot(w/2)$, $|\phi| \geq 3w$, $|\theta| \leq w$. *Then*

$$\begin{aligned}\cot(\frac{\theta-\phi}{2}) &- \sum_{j=1}^{p}(\frac{t_j\tan(\phi/2)-d}{t_j+d\tan(\phi/2)})v_j(-d\tan(\theta/2)) \\ &= \frac{\sec^2(\phi/2)}{\tan(\theta/2)-\tan(\phi/2)}(\frac{\tan(\phi/2)}{\tan(\theta/2)})^{p-1}\cdot \\ &\quad \frac{T_p(-d\tan(\theta/2))}{T_p(-d\tan(\phi/2))}\frac{T_{p-1}(c\cot(\phi/2))}{T_{p-1}(c\cot(\theta/2))}.\end{aligned}$$

Proof. Let $x = -d\tan(\phi/2)$ and $y = -d\tan(\theta/2)$. Then $|x| \geq 3$ and $|y| \leq 1$, and so by Lemmas 2.6.2 and 2.6.3,

$$\cot(\frac{\theta-\phi}{2}) - \sum_{j=1}^{p}\frac{xt_j/d+d}{x-t_j}v_j(-d\tan(\theta/2)) = (\frac{x^2/d+d}{x-y})(\frac{x}{y})^{p-1}\frac{T_p(y)}{T_p(x)}\frac{T_{p-1}(3/x)}{T_{p-1}(3/y)}.$$

Here

$$\frac{xt_j/d+d}{x-t_j} = \frac{-t_j\tan(\phi/2)+d}{-d\tan(\phi/2)-t_j} = \frac{t_j\tan(\phi/2)-d}{t_j+d\tan(\phi/2)},$$

and

$$\frac{x^2/d+d}{x-y} = \frac{d\tan^2(\phi/2)+d}{-d\tan(\phi/2)+d\tan(\theta/2)} = \frac{\sec^2(\phi/2)}{\tan(\theta/2)-\tan(\phi/2)}.$$

□

We use X_I and Y_I as defined in Equations 2.38.

Definition 2.6.8 *Let $p \geq 2$, $l \geq 2$, $|\phi| \geq 3w_l$, $|\theta| \leq w_l$, where $w_l = \pi/2^l$. Then we define*

$$\begin{aligned}
\mathrm{Errloc}_{\cot(x/2)}(p,l,\phi,\theta) &= \rho(\theta-\phi) - \sum_{j=1}^{p} \rho(Y_l^{(l)}(t_j)-\phi)v_j(X_l^{(l)}(\theta)) \\
&= \frac{\sec^2(\phi/2)}{\tan(\theta/2)-\tan(\phi/2)}\left(\frac{\tan(\phi/2)}{\tan(\theta/2)}\right)^{p-1} \cdot \\
&\quad \frac{T_p(\tan(\theta/2)/\tan(w_l/2)}{T_p(\tan(\phi/2)/\tan(w_l/2))}\frac{T_{p-1}(3\tan(w_l/2)/\tan(\phi/2))}{T_{p-1}(3\tan(w_l/2)/\tan(\theta/2))}.
\end{aligned}$$

which by Theorem 2.6.7 is the error in the local expansion of $\cot((\theta-\phi)/2)$ *when using p terms.*

Theorem 2.6.9 *Suppose $p \geq 2$, $l \geq 2$, $|\phi| \geq 3w_l$, $|\theta| \leq w_l$, where $w_l = \pi/2^l$. Then*

$$|\mathrm{Errloc}_{\cot(x/2)}(p,l,\phi,\theta)| < \frac{18+12\sqrt{2}}{(17/3+4\sqrt{2})^p}(\tan(w_l/2)+\cot(w_l/2)/9)$$

Proof. Letting $d = -\cot(w_l/2)$, $x = -d\tan(\phi/2)$, and $y = -d\tan(\theta/2)$, then

$$\mathrm{Errloc}_{\cot(x/2)}(p,l,\phi,\theta) = \frac{(x/d)^2+1}{-y/d+x/d}\left(\frac{x}{y}\right)^{p-1}\frac{T_p(y)}{T_p(x)}\frac{T_{p-1}(3/x)}{T_{p-1}(3/y)}$$

where the right-hand side is the same as that of Equation (2.85), with the same conditions on x and y, but with d replacing c. Then from Inequality 2.86,

$$|\mathrm{Errloc}_{\cot(x/2)}(p,l,\phi,\theta)| < \left|\frac{1}{d}+\frac{d}{x^2}\right|\frac{6}{(1+\sqrt{8}/3)^{2p-1}3^{p-1}}.$$

Since $|x| \geq 3$,

$$\left|\frac{1}{d}+\frac{d}{x^2}\right| \leq \left|\frac{1}{d}\right|\left(1+\frac{d^2}{9}\right) = \tan(w_l/2)+\cot(w_l/2)/9.$$

□

2.6.2 Overall error estimates for a single charge

We make the observation that the analysis in Section 2.5.2 also works for $CIRC$ with $\rho(x) = \cot(\theta/2)$, with the substitutions:

- $w_l = \pi/2^l$
- X_I, Y_I, X_O and Y_O are as in Equations (2.38)–(2.39).

Assume that we take h hierarchy levels in $CIRC$. In the remainder of this section we fix the following:

- x is the position of a unit charge on the interval $[0, 2\pi]$.
- y is another point on $[0, 2\pi]$, where $CIRC$ evaluates the potential due to the charge at x.
- At each level $l = 1, \ldots h$, the interval $[0, 2\pi]$ is divided into 2^l subintervals of length w_l, and:
  - ◇ ξ_l is the center of the subinterval at level l containing x. Note that $|x - \xi_l| \leq w_l$ and $\xi_l - \xi_{l-1} = \pm w_l$.
  - ◇ η_l is the center of the subinterval at level l containing y.

  Note that $|y - \eta_l| \leq w_l$ and $\eta_l - \eta_{l-1} = \pm w_l$.

  Also, $\xi_l - \eta_l$ is a multiple of $2w_l$.

Estimate 2.6.10 *If $|\xi_h - \eta_h| > 2w_h$ then the error in $CIRC$ is given by*

$$\cot(\frac{y-x}{2}) - CIRC(p, h, x, y, 1)$$
$$\approx \sum_{l=d}^{h} (\mathrm{Errloc}_{\cot(x/2)}(p, l, x - \eta_l, y - \eta_l) + \mathrm{Errfar}_{\cot(x/2)}(p, l, x - \xi_l, y - \xi_l))$$

where $\mathrm{Errloc}_{\cot(x/2)}$ *and* $\mathrm{Errfar}_{\cot(x/2)}$ *are as in Definitions 2.6.8 and 2.6.5.*

Derivation. Same as Estimate 2.5.8, but with X_I, Y_I, X_O and Y_O coming from Equations (2.38)–(2.39). □

2.6.3 General error estimate for $CIRC$

Estimate 2.6.11 *The absolute value of the error in calculating* $\cot((y-x)/2)$ *is*

$$|\cot(\frac{y-x}{2}) - CIRC(p,h,x,y,1)| \lesssim \frac{8(5+4\sqrt{2})/3}{(17/3+4\sqrt{2})^p} \cdot 2^h$$

Derivation. Using Estimate 2.6.10:

$$\begin{aligned}
&\cot(\frac{y-x}{2}) - CIRC(p,h,x,y,1) \\
&\approx \sum_{l=d}^{h} (\mathrm{Errloc}_{\cot(x/2)}(p,l,x-\eta_l,y-\eta_l) + \mathrm{Errfar}_{\cot(x/2)}(p,l,x-\xi_l,y-\xi_l)).
\end{aligned}$$

Using the bounds from Theorems 2.6.6 and 2.6.9, this sum is at most

$$\frac{12+8\sqrt{2}}{(17/3+4\sqrt{2})^p}\sum_{l=d}^{h}\csc(w_l) + \frac{18+12\sqrt{2}}{(17/3+4\sqrt{2})^p}\sum_{l=d}^{h}(\tan(w_l/2)+\cot(w_l/2)/9)$$

For $l \geq 2$, $\csc(w_l) \leq \frac{1}{2\sqrt{2}}2^l$ and $\tan(w_l/2)+\cot(w_l/2)/9 \leq (\frac{5\sqrt{2}-4}{18})2^l$, so the sum is bounded above by

$$\frac{20/3+16\sqrt{2}/3}{(17/3+4\sqrt{2})^p}(2^d+\cdots 2^h) < \frac{8(5+4\sqrt{2})/3}{(17/3+4\sqrt{2})^p}2^h.$$

□

Figure 2-4 shows a sample plot of the error in calculating $\cot((y-x)/2)$ when $x = 0.66$ and $y \in [0, 2\pi]$. Here we use expansions of size $p = 10$ and $h = 8$ levels. Estimate 2.5.8 gives an error that is within 1.5×10^{-10} of the observed error. The bound in Estimate 2.5.9 is also within a factor of 4 of the observed maximum error.

Estimate 2.6.12 *If*

$$f_j = \sum_{k=1}^{K} q_k \cot(\frac{y_j - x_k}{2})$$

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

$$\tilde{\mathbf{f}} = CIRC(p,h,\mathbf{x},\mathbf{y},\mathbf{q})$$

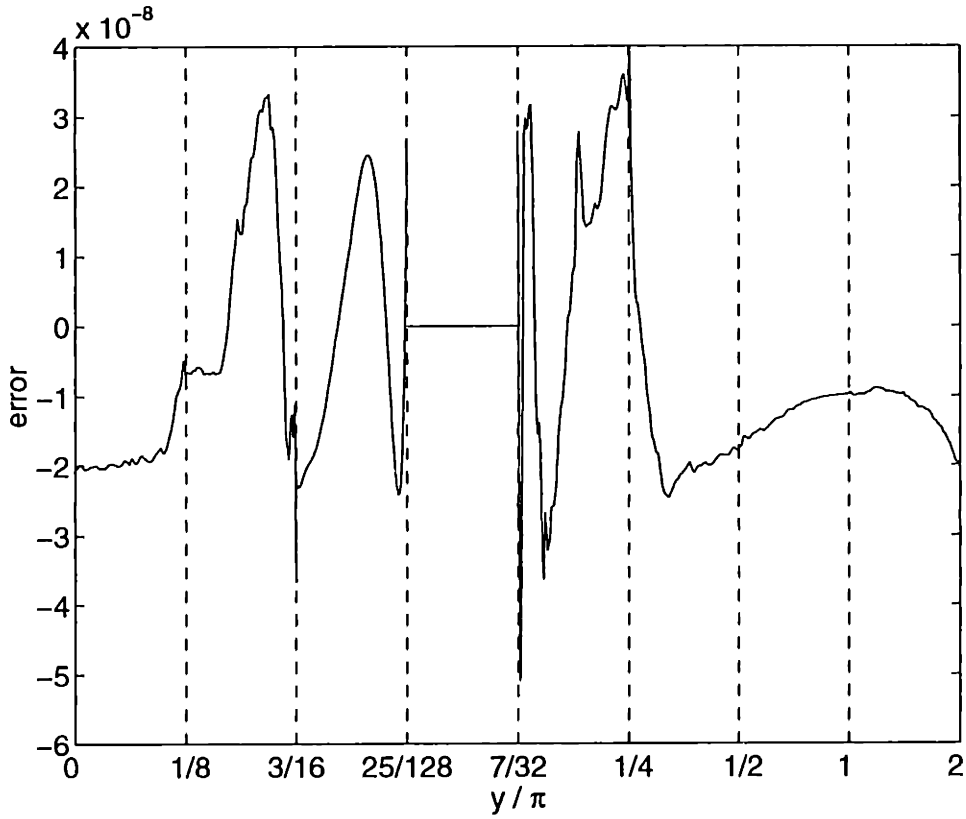


Figure 2-4: Value of the error $\cot((y-0.66)/2) - CIRC(10, 8, 0.66, y, 1)$. Between each pair of dashed vertical lines, the horizontal axis is scaled linearly with a different scaling. The maximum absolute difference between the exact error plotted here and Estimate 2.5.8 is 1.5×10^{-10}, which is too small to be distinguishable in the graphs. Also note that Estimate 2.6.11 gives a bound of 2×10^{-7} on the absolute error. This bound is within a factor of 4 of the observed maximum error.

has components $\widetilde{f}_1, \ldots, \widetilde{f}_N$, *then*

$$|f_j - \widetilde{f}_j| \lesssim (\sum_{k=1}^{K} |q_k|) \frac{8(5+4\sqrt{2})/3}{(17/3+4\sqrt{2})^p} \cdot 2^h.$$

Derivation. Use Estimate 2.6.11 and the fact that $CIRC(p, h, \mathbf{x}, \mathbf{y}, \mathbf{q})$ is linear in $\mathbf{q}$. □

Estimate 2.6.13 *If* $\mathbf{f}$ *has components*

$$f_j = \sum_{k=1}^{K} q_k \cot(\frac{y_j - x_k}{2})$$

for $j = 1, \ldots, N$, *then*

$$\|\mathbf{f} - CIRC(p, h, \mathbf{x}, \mathbf{y}, \mathbf{q})\|_2 \lesssim \|\mathbf{q}\|_2 \sqrt{KN} \cdot \frac{8(5 + 4\sqrt{2})/3}{(17/3 + 4\sqrt{2})^p} \cdot 2^h.$$

Derivation. This estimate is derived the same way as Estimate 2.5.11 for LIN, but using Estimate 2.6.11 instead of Estimate 2.5.9. □

The estimates in this section are for the worst-case error, which doubles with each successive level in $CIRC$, as it does in LIN. Also, Figure 2-3 illustrates that with our interpolation functions, the error for a single charge and evaluation point is roughly uniform across the possible relative positions.

2.7 Numerical experiments

Figure 2-5 shows results of an experiment with the $1/x$ potential on $[-1, 1]$. We chose 1024 evaluation positions and 1024 charge positions distributed uniformly on $[-1, 1]$, with the charge strengths being normally distributed with mean 0 and variance 1. Error is reported as the normalized ℓ^2 error, $\|f - \widetilde{f}\|_2 / \|f\|_2$.

The results of all eight algorithms described in the table in Section 2.3.1 are plotted in Figure 2-5. Each marker plots the observed error and the flop count for the EVAL portion of the corresponding algorithm. A marker containing a number, p, represents a data point for an algorithm that uses expansions of p terms, but does *not* use SVD compression. Such a marker is connected by thin black lines to smaller markers of the same shape, which represent data points for the same algorithms using the same initial lengths of expansions, but with SVD compression to smaller lengths.

The number of levels used by each algorithm was chosen by Equation (2.21) or (2.37) to be optimal if the points were equally spaced.

The figure shows that without compression of function spaces, the algorithms that use virtual charge interpolation functions v_k outperform those that use polynomial interpolation functions u_k, and those that use two-sided expansions outperform those that use one-sided expansions. Our algorithm LIN is the best of those that do not

use compression.

With compression, however, the errors in all of the algorithms converge to an envelope. This phenomenon can be explained by convergence to the (optimal) set of singular functions of the operator mapping charges to potentials, as discussed in Yarvin and Rokhlin [34].

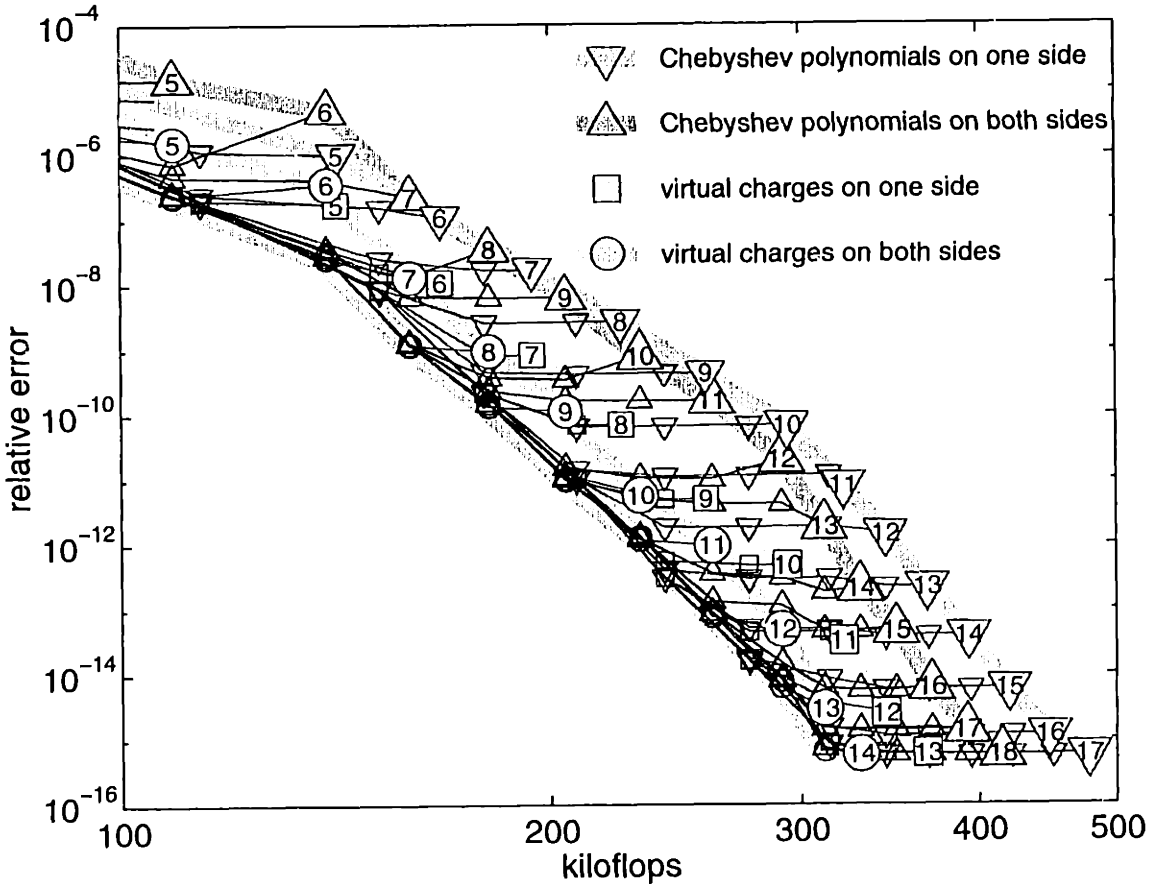


Figure 2-5: Plot of relative error vs. number of flops of the LIN family of algorithms, for a random distribution of 1024 charges and 1024 points with $1/x$ potential. Each marker labelled with a number p is for an algorithm using expansions of length p with no compression. It is connected by thin black lines to smaller markers of the same type, for algorithms with compression. Notice the convergence to an envelope.

In Figure 2-6 we have plotted the errors in LIN and $LINSVD$ (also in Figure 2-5), together with the error bound from Estimate 2.5.11. We see more clearly, from the difference between the graph of the error in LIN and the error envelope, that SVD compression contributes approximately one additional correct digit to the accuracy of

LIN. Also, the actual error in this example is about two orders of magnitude below the bound from Estimate 2.5.11.

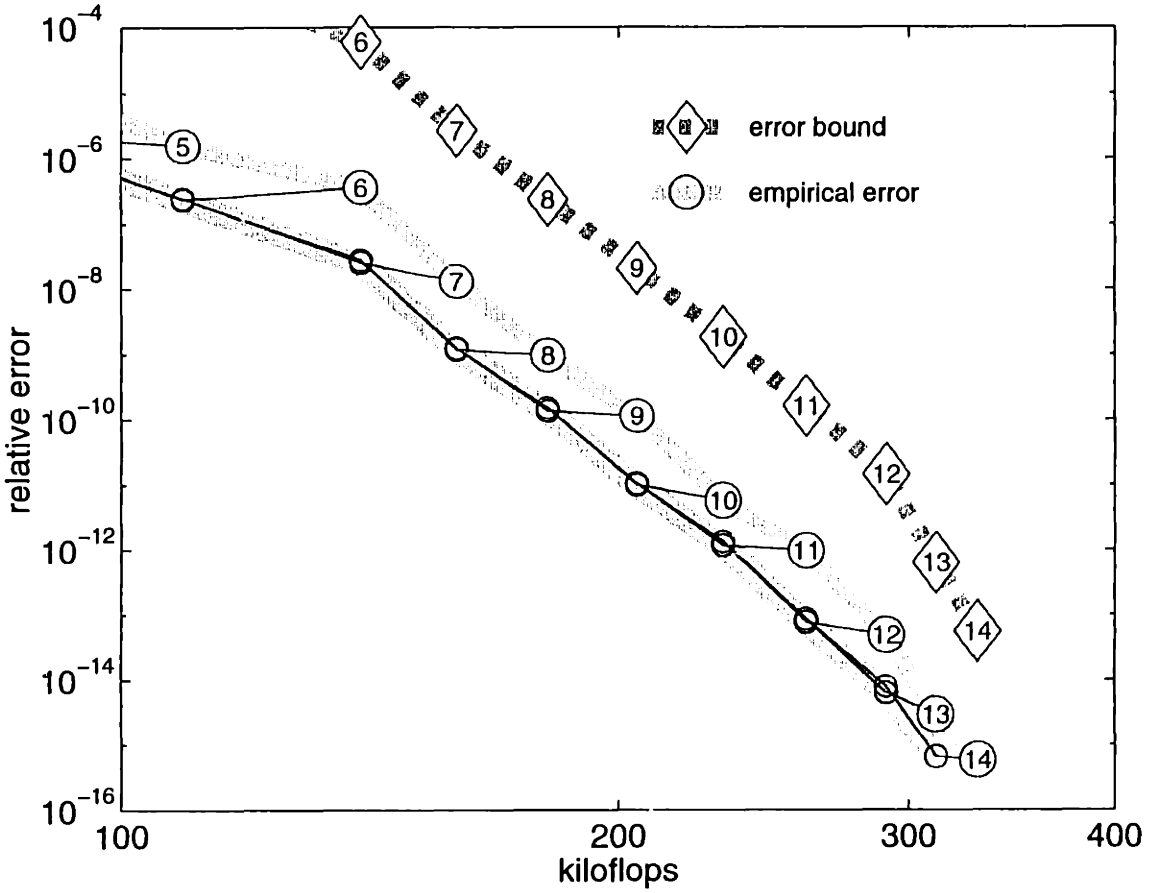


Figure 2-6: Relative error in LIN and $LINSVD$ from Figure 2-5, together with the error bound from Estimate 2.5.11

Figure 2-7 shows results of an experiment with the $\cot(\theta/2)$ potential on the circle. We chose 1024 evaluation positions and 1024 charge positions distributed uniformly on the circle, with the charge strengths again being normally distributed with mean 0 and variance 1. Error is reported as the normalized ℓ^2 error, $\|f - \tilde{f}\|_2/\|f\|_2$.

The results of all four algorithms described in the table in Section 2.4.1 are plotted in Figure 2-7. As in Figure 2-5, each marker plots the observed error and the flop count for the EVAL portion of the corresponding algorithm, and a marker containing a number, p, represents a data point for an algorithm that uses expansions of p terms, but does *not* use SVD compression. Such a marker is connected by thin black

lines to smaller markers of the same shape, which represent data points for the same algorithms using the same initial lengths of expansions, but with SVD compression.

The number of levels used by each algorithm was chosen by Equation (2.54) to be optimal if the points were equally spaced.

The figure shows again that without compression, the algorithms that use virtual charge interpolation functions v_k outperform those that use polynomial interpolation functions u_k. Our algorithm $CIRC$ is the best of those that do not use compression.

With compression, however, the errors in all of the algorithms converge to an envelope, as they do with LIN and its variants. Again, this can be explained by convergence to the (optimal) singular functions of the operator mapping charges to potentials.

In Figure 2-8 we have plotted the errors in $CIRC$ and $CIRCSVD$ (also in Figure 2-7), together with the error bound from Estimate 2.6.13. We see more clearly, from the difference between the graph of the error in $CIRC$ and the error envelope, that SVD compression contributes approximately one additional correct digit to the accuracy of $CIRC$. Also, the actual error in this example is about two or three orders of magnitude below the bound from Estimate 2.6.13.

2.8 Conclusions

In this chapter, we have described eight algorithms for evaluating one-dimensional potentials on the line. Four of the algorithms are new in that they use the new interpolation functions v_k. We have also described four algorithms, including two new ones, for evaluating potentials on the circle. Numerically we have found that with SVD compression, all of the algorithms on the line and all of the algorithms on the circle become indistinguishable in terms of the work-accuracy tradeoffs, because the basis of interpolation functions converges to the principal singular functions of the charges-to-potentials operator.

Our algorithms LIN and $CIRC$ are closest to representing the compressed versions, and so far are the best algorithms of this type for which useful error estimates

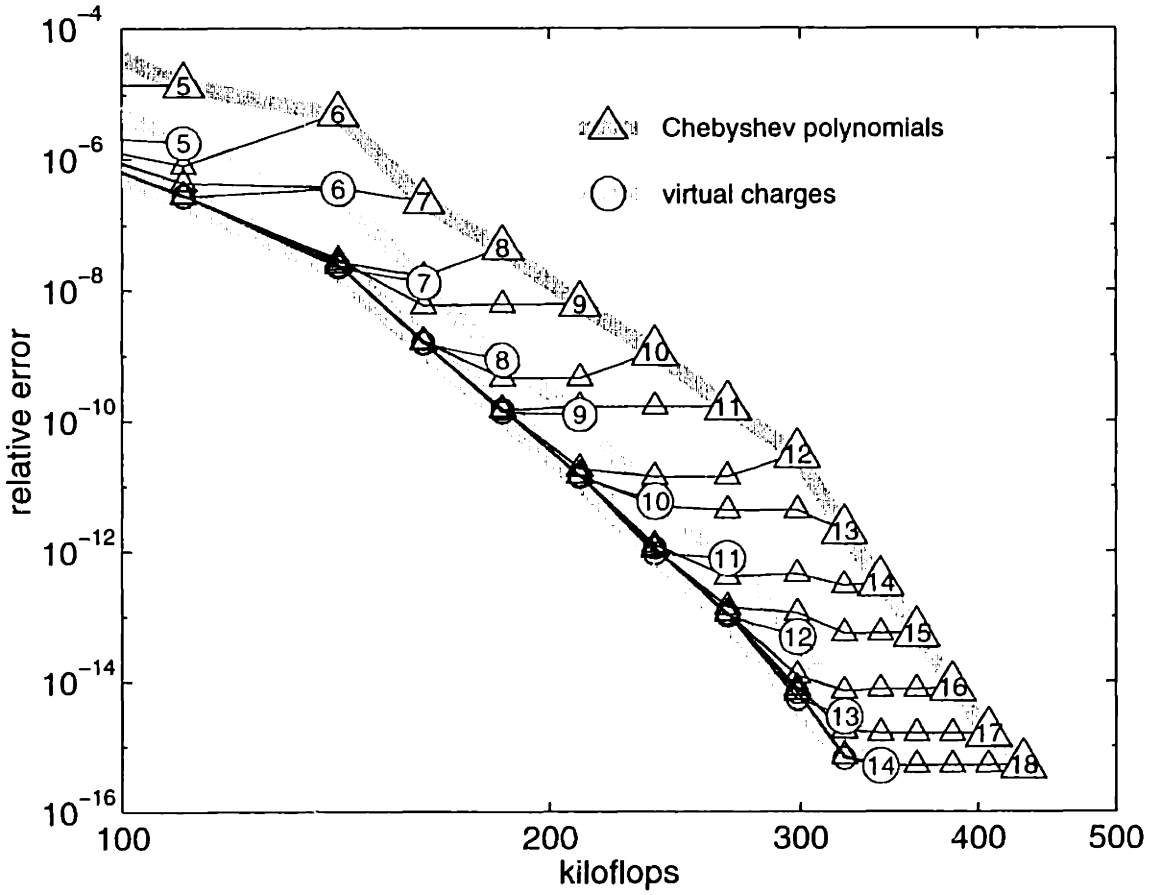


Figure 2-7: Plot of relative error vs. number of flops of the $CIRC$ family of algorithms, for a random distribution of 1024 charges and 1024 points with $\cot(\theta/2)$ potential. Each marker labelled with a number p is for an algorithm using expansions of length p with no compression. It is connected by thin black lines to smaller markers of the same type, for algorithms with compression. Notice the convergence to an envelope.

have been derived. Error bounds on SVD compressed methods have not been derived except numerically [34]. The derivations of our error estimates depend on the fact that the interpolation functions are rational, and it seems unlikely that other rational functions could be chosen that would be closer to the error envelope of singular functions shown in Figures 2-5 and 2-7.

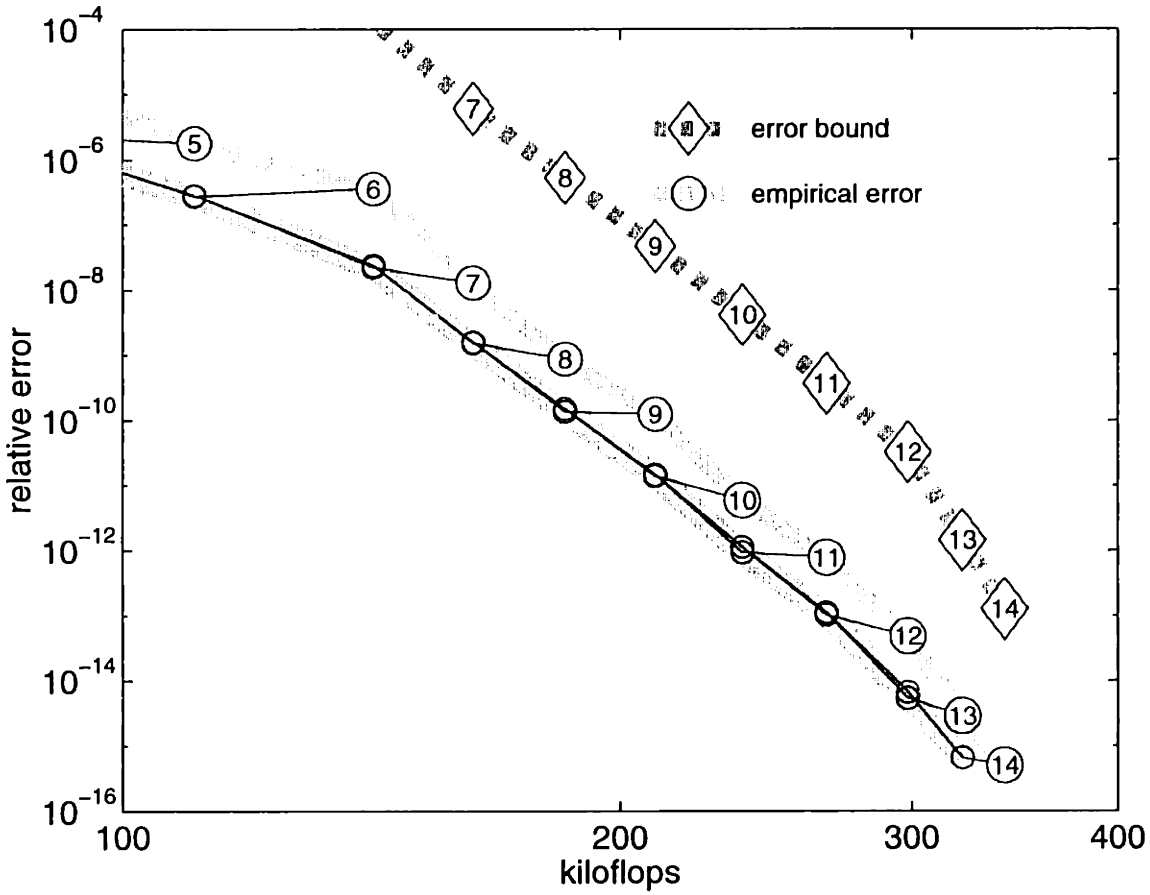


Figure 2-8: Relative error in $CIRC$ and $CIRCSVD$ from Figure 2-7, together with the error bound from Estimate 2.6.13.

Chapter 3

Low-Communication Parallel Fourier Transforms

3.1 Parallel Fourier Transforms

In future high-performance parallel computers, improvements in floating-point performance are likely to continue to outpace improvements in communication bandwidth. Therefore important algorithms for the future may trade off arithmetic for reduced communication. Indeed, with the increasing popularity of networks of workstations and clusters of symmetric multiprocessors, even on present machines it may be worthwhile to make this tradeoff.

Traditional research into algorithmic design for the Fast Fourier Transform focuses on memory and cache management and organization. All such algorithms are in effect variations of the original algorithm of Cooley and Tukey [8]. A few important variants are the Stockham framework [7], which reorders data at each step, the Bailey method [5], which minimizes the number of passes through external data sets, Swarztrauber's method [29] for hypercubes and vector supercomputers, and the recent algorithm by Cormen and Nicol [9] which reorganizes data for out-of-core algorithms. Many other important references may be found in Van Loan [31]. The methods described in this chapter depart from traditional FFT design in that they are *approximate* algorithms that would not be exact in the absence of roundoff error.

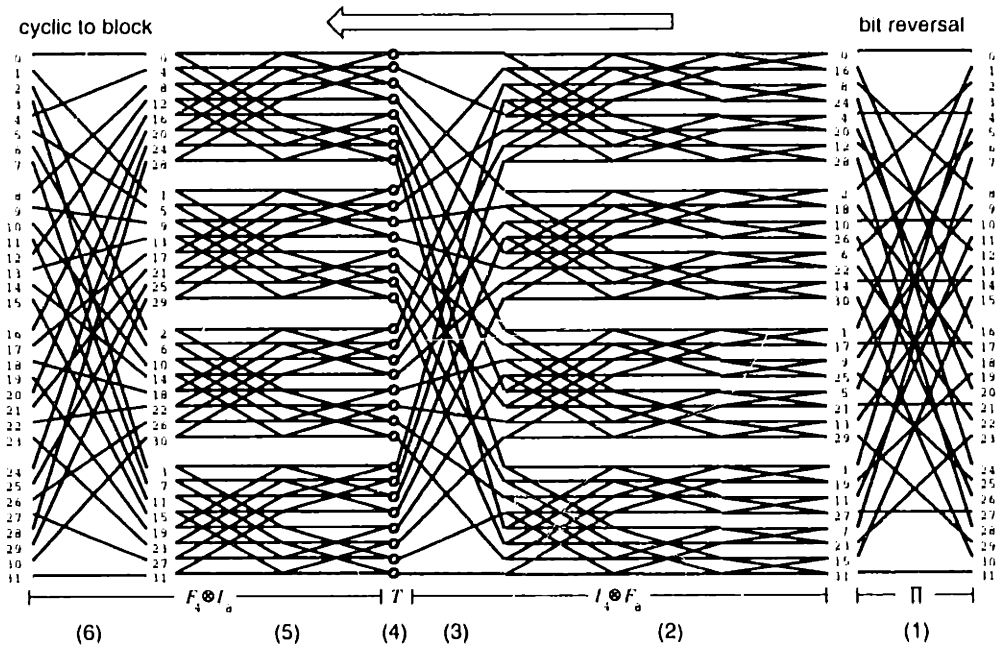


Figure 3-1: Communication pattern in parallel FFT of length 32 over 4 processors, using the six-step framework based on the factorization $F_{32} = (F_4 \otimes I_8)T(I_4 \otimes F_8)\Pi$ of equation (3.8) in Section 3.4. The step numbers are indicated at the bottom of the figure.

In our distributed-memory model of Section 1.3, we assume that the input vector is stored in natural order, with each processor holding a contiguous portion of the data. The output vector should be distributed the same way. In this model, the standard approach to the parallel FFT is known as the "six-step framework" [31, pages 173–174], consisting of: (1) a global bit reversal or shuffle, (2) local FFTs, (3) a global transpose, (4) multiplication by twiddle factors, (5) local FFTs, (6) a global shuffle or bit reversal. The global shuffles in steps (1) and (6) each require an amount of communication equivalent to the transpose in step (3). They may be saved if the order is not important. The communication pattern is as indicated in Figure 3-1, which is based on Gupta et al. [21].

This chapter presents a method which can save up to a factor of three in communication cost, by using an approximate algorithm that essentially combines the three global transposes into one. Accuracy can be extended to full machine precision with negligible effect on communication complexity.

This chapter is organized as follows. Section 3.2 contains a mathematical discussion of the singular value ideas that explain "why" a reduction in communication is possible. Section 3.3 introduces a matrix-vector multiply algorithm that uses an off-line singular value analysis. Section 3.4 introduces our parallel FMM-based algorithm, $FMDFT$, an equispaced variation of the non-equispaced Fourier transform proposed by Dutt and Rokhlin [11]. Section 3.5 discusses the results of numerical experiments which show that $FMDFT$ is in fact faster than a traditional parallel FFT on some current computer architectures.

3.2 Mathematical insights

In this section, we do not yet present our algorithms, but rather provide mathematical explanations as to why one expects that more communication-efficient algorithms may be found. The most important underlying critical idea is the notion of near-rank deficiency. The operators that represent the relationship between the input on one processor and the output on another processor are nearly rank-deficient. Therefore, this represents an opportunity to replace the operator with its more economical rank-deficient variant, thereby gaining speedups on parallel supercomputers. We shall see later that the existence of a multipole algorithm is really a way of taking advantage of this fact.

One can mathematically press further and ask for an explanation of why we are lucky enough to be in this near-rank deficiency situation at all. The answer to such a question may be found in an understanding of the link between our linear operator and its continuous limiting form. Such an understanding is closely related to the mathematics of prolate functions which we shall explain in this section.

Definition 3.2.1 *The DFT of* $\mathbf{x} \in \mathbb{C}^n$ *is*

$$\mathbf{y} = F_n \mathbf{x},$$

where

$$[F_n]_{jk} = \exp(-2\pi ijk/n) \qquad (0 \le j, k \le n-1).$$

Let $F_{n|p}$ denote the top left $m \times m$ submatrix of the unitary matrix $\frac{1}{\sqrt{n}}F_n$, where $m = n/p$ is an integer.

Proposition 3.2.2 *The singular values of $F_{n|p}$ are strictly between 0 and 1.*

Proof. Since $F_{n|p}$ is a submatrix of a unitary matrix, its singular values are at most 1. Moreover, $F_{n|p}$ is a Vandermonde matrix and hence nonsingular.

The CS decomposition [13, p. 77] of $\frac{1}{\sqrt{n}}F_n$ shows that if any singular value of $F_{n|p}$ is equal to 1, then 0 occurs as a singular value of a rectangular block of F_n, but this is not possible because the rectangular block would be a section of a Vandermonde matrix, which has full rank. □

Proposition 3.2.3 *If $p = 2$ then singular values occur in sine-cosine pairs, that is, $\sigma_j^2 + \sigma_{m+1-j}^2 = 1$.*

Proof.

If $\frac{1}{\sqrt{n}}F_n$ is split into four blocks, then all four blocks differ from $F_{n|2}$ by a diagonal unitary matrix, and hence they have the same singular values. Then the CS decomposition shows that the singular values must occur in sine-cosine pairs. □

For any p, the singular values of $F_{n|p}$ have an interesting property suggested by the plot in Figure 3-2: a fraction $1/p$ of them are close to 1, and the rest are close to 0. Toledo [30] observed this for $p = 2$.

This is a remarkable property of sections of the Fourier matrix. By contrast, if one takes a random unitary matrix (with Haar measure) and plots the singular values of a section, one finds that for $p = 4$ the singular values appear to be uniformly distributed on the interval $(0, \sqrt{3}/2)$, as shown in Figure 3-3.

An exact statement of the near-rank deficiency of a Fourier matrix section is obtained by bounding the number of singular values away from 0 and 1:

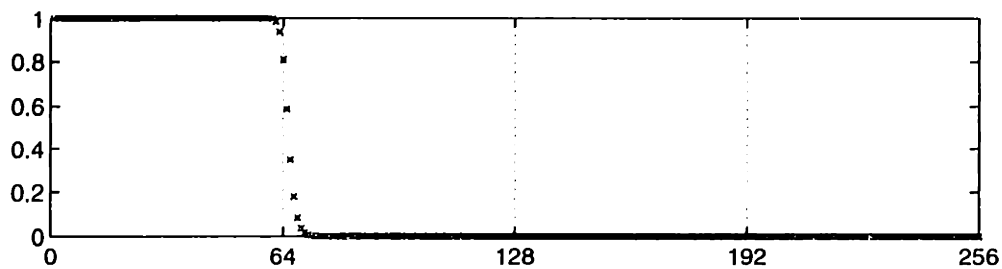


Figure 3-2: Singular values of $F_{1024|4}$

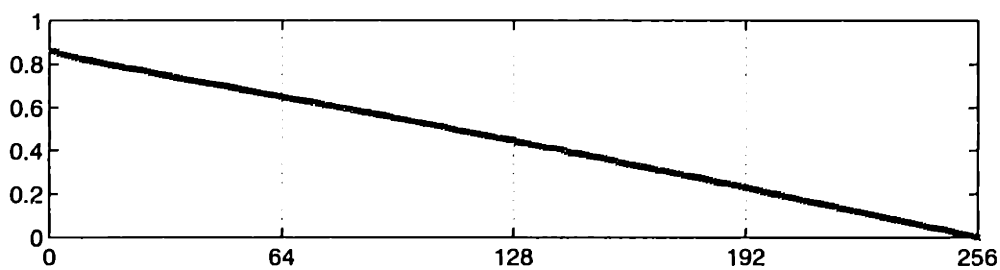


Figure 3-3: Singular values of a 256×256 section of a random 1024×1024 unitary matrix, computed with MATLAB.

Theorem 3.2.4 *For fixed p and $0 < \epsilon < \frac{1}{2}$, let $S_n(a,b)$ represent the number of singular values of $F_{n|p}$ in the interval (a,b). Then asymptotically, with $m = n/p$:*

$$\begin{aligned} S_n(0,\epsilon) &\sim \frac{1}{p}m, \\ S_n(\epsilon, 1-\epsilon) &\sim O(\log n), \\ S_n(1-\epsilon, 1) &\sim (1-\frac{1}{p})m. \end{aligned}$$

Proof. Up to unitary phase factors, $F^*_{n|p}F_{n|p}$ is equal to the $m \times m$ matrix $G_{n|p} = [g_{jk}]$ where

$$g_{jk} = \begin{cases} \frac{1}{p} & \text{if } j = k; \\ \frac{\sin(\pi(k-j)/p)}{n\sin(\pi(k-j)/n)} & \text{if } j \neq k. \end{cases} \tag{3.1}$$

The singular values $\sigma_1, \ldots, \sigma_m$ of $F_{n|p}$ are the positive square roots of the eigenvalues

of $G_{n|p}$. Taking the Frobenius norm of $G_{n|p}$,

$$\begin{aligned}
\|G_{n|p}\|_F^2 &= \sum_{j,k} |g_{jk}|^2 = \frac{m}{p^2} + \frac{1}{n^2}\sum_{j\neq k} \frac{\sin^2(\frac{\pi(k-j)}{p})}{\sin^2(\frac{\pi(k-j)}{n})} = \frac{m}{p^2} + \frac{2}{n^2}\sum_{l=1}^{m-1}(m-l)\frac{\sin^2(\frac{\pi l}{p})}{\sin^2(\frac{\pi l}{n})} \\
&= \frac{m}{p^2} + \frac{2}{n^2}m\sum_{l=1}^{m-1}\frac{\sin^2(\frac{\pi l}{p})}{\sin^2(\frac{\pi l}{n})} - \frac{2}{n^2}\sum_{l=1}^{m-1} l\frac{\sin^2(\frac{\pi l}{p})}{\sin^2(\frac{\pi l}{n})} \\
&\sim \frac{m}{p^2} + (\frac{m}{p} - \frac{m}{p^2}) - \frac{1}{\pi^2}\ln\frac{m}{p} \\
&\sim \frac{n}{p^2} - \frac{1}{\pi^2}\ln n.
\end{aligned}$$

Since the eigenvalues of $G_{n|p}$ are σ_j^2, we have, by the trace formula,

$$\sum_{j=1}^{m} \sigma_j^2 = m\frac{1}{p} = \frac{n}{p^2}. \tag{3.2}$$

But σ_j^2 are also the singular values of $G_{n|p}$, so

$$\sum_{j=1}^{m} \sigma_j^4 = \|G_{n|p}\|_F^2 \sim \frac{n}{p^2} - \frac{1}{\pi^2}\ln n. \tag{3.3}$$

Subtracting equation (3.3) from equation (3.2), then

$$\sum_{j=1}^{m} \sigma_j^2(1-\sigma_j^2) \sim \frac{1}{\pi^2}\ln n. \tag{3.4}$$

Since $0 < \sigma_j < 1$, equation (3.4) implies $S_n(\epsilon, 1-\epsilon) = O(\log n)$. And then from equations (3.2) and (3.3), we also have $S_n(0,\epsilon) \sim m/p$ and $S_n(1-\epsilon, 1) \sim m(1-1/p)$.
□

A better understanding of the transition region is suggested by examining the generating function [18], $g(x) = \sum_{l=-\infty}^{\infty} \widehat{g}(l)e^{2\pi i l x}$, of the *infinite* Toeplitz matrix with entries $g_{jk} = \widehat{g}(k-j)$ given by equation (3.1), for $j, k \in \mathbb{Z}$. In this case, we have

$$g(x) = \frac{1}{n}\sum_{j=1}^{m} \delta(x - \frac{2j-1-m}{2n}),$$

which is a comb function having $m = n/p$ spikes equally spaced in the interval $(-\frac{1}{2p}, \frac{1}{2p})$. This function can be viewed as a discrete version of the periodic window function defined on $[-\frac{1}{2}, \frac{1}{2}]$ as

$$w(x) = \begin{cases} 1 & \text{if } x \in [-\frac{1}{2p}, \frac{1}{2p}]; \\ 0 & \text{if } x \in [-\frac{1}{2}, -\frac{1}{2p}) \cup (\frac{1}{2p}, \frac{1}{2}]. \end{cases}$$

which is the generating function of an infinite Toeplitz matrix with entries

$$w_{jk} = \widehat{w}(k-j) = \frac{\sin(\pi(k-j)/p)}{\pi(k-j)}.$$

The finite version W_p of this matrix is known as the *prolate matrix* [32], and it has been well studied in the signal processing literature. Observe that $G_{n|p}$ converges elementwise to W_p:

$$\lim_{n\to\infty} g_{jk} = w_{jk}.$$

Slepian [27] observed that the prolate matrix W_p also has eigenvalues clustered near 0 and 1, with a fraction $1/p$ of them near 1 and the rest near 0. In the transition region near $j = m/p$, the jth largest eigenvalue is given asymptotically by

$$\lambda_j(W_p) \sim \frac{1}{1+\exp(\pi\beta)}$$

where $j = \lceil m/p + \frac{\beta}{\pi}\log m \rceil$. In the region $0.2 < \lambda_j < 0.8$, Slepian found that a good approximation [27, eqns. 61–62] is

$$\lambda_j(W_p) \sim \frac{1}{1+\exp(\pi\widehat{\beta})} \tag{3.5}$$

where

$$\widehat{\beta} = \frac{\pi(j-1/2-m/p)}{\log(8m|\sin(\pi/p)|)+\gamma}$$

and $\gamma = 0.5772156649$ is the Euler-Mascheroni constant. Table 3.1 shows that approximation (3.5) is a good predictor of the eigenvalues of the prolate matrix, W_p, which are also close to the eigenvalues of $G_{n|p}$.

Table 3.1: A listing of eigenvalues in a transition interval for the matrices $G_{n|p}$ and W_p, of order n/p, and the approximation (3.5).

p	n	n/p	j	$\lambda_j(G_{n\|p})$	$\lambda_j(W_p)$	approx
2	512	256	127	0.8774	0.8643	0.8588
			128	0.6545	0.6465	0.6460
			129	0.3455	0.3535	0.3540
			130	0.1226	0.1357	0.1412
2	1024	512	255	0.8575	0.8453	0.8408
			256	0.6425	0.6356	0.6352
			257	0.3575	0.3644	0.3648
			258	0.1425	0.1547	0.1592
4	1024	256	63	0.8774	0.8743	0.8681
			64	0.6545	0.6525	0.6521
			65	0.3454	0.3473	0.3479
			66	0.1226	0.1257	0.1319

These results on the eigenvalue distribution of the prolate matrix come from finding the asymptotics of discrete prolate spheroidal functions, which are solutions of a differential equation with a differential operator related to a tridiagonal matrix that commutes with the prolate matrix. A similar analysis may, in principle, be applicable to $G_{n|p}$. Grünbaum [19] makes the first step in this direction by finding a tridiagonal matrix that commutes with $G_{n|p}$.

3.3 *MVDFT*: A Matrix-Vector Algorithm

Given that the sections $F_{n|p}$ are nearly rank-deficient, we may borrow an idea from Section 2.3.5 and take a singular-value decomposition. Our first algorithm for the DFT involves nothing more than matrix-vector multiplication by SVD matrices. In the equation $\mathbf{y} = F_n\mathbf{x}$ from Definition 3.2.1, if we write F_n as p^2 blocks of size $m = n/p$ and if p^2 divides n, then

$$\begin{pmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{p-1} \end{pmatrix} = \sqrt{n} \begin{pmatrix} F_{n|p} & \cdots & D^{p-1}F_{n|p} \\ F_{n|p}D & \cdots & D^{p-1}F_{n|p}D \\ \vdots & & \vdots \\ F_{n|p}D^{p-1} & \cdots & D^{p-1}F_{n|p}D^{p-1} \end{pmatrix} \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{p-1} \end{pmatrix}, \tag{3.6}$$

where

$$D = \mathrm{diag}(1, \eta, \eta^2, \ldots, \eta^{m-1}), \qquad \eta = \exp(-2\pi i/p).$$

Since the number of significant singular values of $F_{n|p}$ is asymptotically only m/p, this suggests the idea of using compression to reduce communication.

The singular-value decomposition of the complex matrix $F_{n|p}$ is written

$$F_{n|p} = U\Sigma V^*,$$

where U and V are $m \times m$ unitary matrices, and Σ is a diagonal matrix of (non-negative) singular values. Let Σ_k denote the matrix consisting of the first k rows of Σ, containing the k largest singular values. The value of k depends on the accuracy desired, but for fixed accuracy and fixed p, the results of the previous section tell us $k = m/p + O(\log m)$.

Let U_k be the first k columns of U. We shall use the approximation

$$F_{n|p} \approx U_k \Sigma_k V^*. \tag{3.7}$$

If we precompute the $k \times m$ matrices

$$A^{(r)} = \Sigma_k V^* D^r \qquad (0 \le r \le p-1)$$

and the $m \times k$ matrices

$$B^{(r)} = D^r U_k \qquad (0 \le r \le p-1)$$

then we have the following algorithm.

The input and output vectors $\mathbf{x}$ and $\mathbf{y}$ are distributed across the p processors as in Equation (3.6), with $\mathbf{x}_\mu$ and $\mathbf{y}_\mu$ stored in processor μ. The intermediate vectors $\mathbf{u}_\mu^{(r)}$ and $\mathbf{v}_\mu^{(r)}$ are of length k and are stored in processor μ.

Algorithm 3.3.1

$\mathbf{y} = MVDFT(\mathbf{x})$ evaluates the DFT of $\mathbf{x}$ approximately, using matrix-vector multi-

plication.

1. **foreach** processor $\mu = 0, \ldots, p-1$:

   for $r = 1, \ldots, p$:

   $\mathbf{v}_\mu^{(r)} = A^{(r)}\mathbf{x}_\mu$

   end

2. **foreach** processor $\mu = 0, \ldots, p-1$:

   for $r = 1, \ldots, p$:

   Send $\mathbf{v}_\mu^{(r)}$ to processor r;

   Receive $\mathbf{u}_\mu^{(r)}$ from processor r.

   end

3. **foreach** processor $\mu = 0, \ldots, p-1$:

   $\mathbf{y}_\mu = \sum_{r=0}^{p-1} B^{(r)}\mathbf{u}_\mu^{(r)}$

3.3.1 Accuracy

Lemma 3.3.2 *When* $\mathbf{x}$ *has length* n *and is run on* p *processors with* k *of the* $m = n/p$ *singular values of* $F_{n|p}$ *being used, the relative error in* $MVDFT$ *is given by*

$$\frac{\|F_n\mathbf{x} - MVDFT(\mathbf{x})\|_2}{\|F_n\mathbf{x}\|_2} \leq p(\sum_{j=k+1}^{m} \sigma_j^2)^{1/2}.$$

Proof. We can write $MVDFT(\mathbf{x})$ as the linear operator $\widetilde{F}_n\mathbf{x}$. Then

$$\|F_n - \widetilde{F}_n\|_2 \leq \|F_n - \widetilde{F}_n\|_F = p\sqrt{n}\|U\Sigma V^* - U_k\Sigma_k V^*\|_F = p\sqrt{n}(\sum_{j=k+1}^{m} \sigma_j^2)^{1/2}.$$

For $\mathbf{x} \in \mathbb{C}^n$,

$$\|F_n\mathbf{x} - MVDFT(\mathbf{x})\|_2 \leq p\sqrt{n}(\sum_{j=k+1}^{m} \sigma_j^2)^{1/2}\|\mathbf{x}\|_2 = p(\sum_{j=k+1}^{m} \sigma_j^2)^{1/2}\|F_n\mathbf{x}\|_2.$$

□

Table 3.2: Number of significant singular values required for given relative accuracy.

	Size		Relative accuracy		
p	n	n/p^2	10^{-5}	10^{-8}	10^{-11}
2	1024	256	271	278	284
	2048	512	529	537	544
	4096	1024	1043	1051	1060
4	1024	64	78	84	89
	2048	128	144	151	157
	4096	256	274	282	289
	8192	512	532	540	549
8	1024	16	28	32	36
	2048	32	45	51	55
	4096	64	79	85	91
	8192	128	145	152	159

Table 3.2 shows k, the number of significant singular values, that must be used with several choices of n and p in order to obtain selected values of relative accuracy. These results were computed on a Sun Ultra Enterprise 5000 with the SunSoft Performance Library.

3.3.2 Complexity

In each of Steps 1 and 3, every processor performs p complex matrix-vector multiplications, each requiring $8km$ flops. Hence the total number of flops per processor is

$$16mkp = 16m^2 + O(m \log m).$$

All communication takes place in Step 2, where there is an all-to-all personalized communication with each processor sending k scalars to each other processor. The total number of scalars sent by each processor is

$$(p-1)k = m(1 - 1/p) + O(\log m).$$

The parallel Fast Fourier Transform, by comparison, has each processor sending $3m(1 - 1/p)$ scalars but uses only $5m \lg n$ flops. The matrix-vector multiplication algorithm using the SVD saves as much as a factor of three in communication at the

cost of greater arithmetic. In the next section, we show how a different algorithm using the fast multipole method can reduce the arithmetic but maintain this saving in communication.

3.4 *FMDFT*: Fast Multipole Approach

The near-rank deficiency shown in Section 3.2, together with the understanding that the Fourier transform is a highly structured problem, leads to the suggestion that a multipole-based algorithm may be appropriate. The suggestion is subtle, for it is the converse that is really correct. It is well known that one can cluster charges or particles when evaluating potential fields far away. In the language of linear algebra, the linear operator that transforms charge or mass to faraway potentials is approximately low-rank. It is therefore intuitively reasonable to attempt to identify nearly low-rank matrices that arise from highly structured mathematical problems with the evaluation of some sort of potential field. We shall see that commuting the so-called "twiddle factor" matrix through the "butterfly" operations, leads to just this sort of identification.

3.4.1 Matrix factorizations

For parallel FFT computations over p processors, the standard "six-step framework" [31, pages 173–174] is based on the radix-p splitting [31, eqn. (2.1.5)], a factorization of the Fourier matrix as

$$F_n = (F_p \otimes I_m)T(I_p \otimes F_m)\Pi \tag{3.8}$$

where again $m = n/p$ and T is a diagonal matrix of twiddle factors,

$$T = \mathrm{diag}(I_m, \Omega, \Omega^2, \ldots, \Omega^{p-1}),$$

$$\Omega = \mathrm{diag}(1, \omega, \omega^2, \ldots, \omega^{m-1}), \qquad \omega = \exp(-2\pi i/n),$$

and Π is a block-to-cyclic permutation that acts on the columns of the identity matrix as:

$$\Pi \mathbf{e_{j+kp}} = \mathbf{e_{k+jm}} \qquad (0 \le j \le p-1, 0 \le k \le m-1).$$

Our algorithm will use a factorization

$$F_n = (I_p \otimes F_m)(F_p \otimes I_m)M\Pi. \tag{3.9}$$

To solve for M, use the fact that $I_p \otimes F_m$ and $F_p \otimes I_m$ commute, so

$$(F_p \otimes I_m)T(I_p \otimes F_m)\Pi = F_n = (F_p \otimes I_m)(I_p \otimes F_m)M\Pi$$

which gives

$$\begin{aligned} M &= (I_p \otimes F_m)^{-1}T(I_p \otimes F_m) \\ &= \mathrm{diag}(I_m, F_m^{-1}\Omega F_m, \ldots, F_m^{-1}\Omega^{p-1}F_m) \\ &= \mathrm{diag}(I_m, C^{(1)}, \ldots, C^{(p-1)}) \end{aligned} \tag{3.10}$$

where the matrices $C^{(s)} = (c_{jk}^{(s)})$ have elements

$$c_{jk}^{(s)} = \gamma^{(s)}[\cot(\frac{\pi}{m}(k-j+\frac{s}{p})) + i] \qquad (j = 0, \ldots, m-1;\ k = 0, \ldots, m-1),$$

with $\gamma^{(s)} = \frac{1}{m}\exp(-i\pi s/p)\sin(\pi s/p)$. We shall define the matrices $G^{(s)} = (g_{jk}^{(s)})$ to have elements

$$g_{jk}^{(s)} = \cot(\frac{\pi}{m}(k-j+\frac{s}{p})) \qquad (j = 0, \ldots, m-1;\ k = 0, \ldots, m-1),$$

so that

$$C^{(s)} = \gamma^{(s)}(G^{(s)} + i). \tag{3.11}$$

For fast multiplication by $G^{(s)}$, we can use the algorithm $CIRC$ described in Section 2.4, with the potential function $\rho(x) = \cot(x/2)$.

3.4.2 General Approach

In evaluating the DFT $\mathbf{y} = F_n\mathbf{x}$, we assume that $\mathbf{x} \in \mathbb{C}^n$ is stored in block order and $\mathbf{y} \in \mathbb{C}^n$ should also end in block order. That is,

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{p-1} \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{p-1} \end{pmatrix}$$

where $\mathbf{x}_\mu$ and $\mathbf{y}_\mu$ are stored in processor μ.

One possible approach is as follows:

1. Perform the distributed permutation $\Pi\mathbf{x}$.

2. In processor $\mu = 1, \ldots, p-1$, multiply local vector by $C^{(\mu)}$.

3. Do m distributed FFTs of size p, with one element from each processor. (Corresponds to $F_p \otimes I_m$.)

4. In each processor, do a local FFT of size m. (Corresponds to $I_p \otimes F_m$.)

This method requires two distributed permutations, one in Step 1 and the other in the distributed FFT in Step 3.

We use an alternative approach that avoids performing the distributed permutation Π directly. Instead, we combine steps 1 and 2, doing each of the p multiplications by $C^{(s)}$ matrices in parallel. In terms of total number of scalars sent, the communication requirements are reduced by nearly half. In the description of the algorithm below, we take

$$\mathbf{x}^{(s)} = \begin{pmatrix} \mathbf{x}_0^{(s)} \\ \mathbf{x}_1^{(s)} \\ \vdots \\ \mathbf{x}_{p-1}^{(s)} \end{pmatrix},$$

where $\mathbf{x}_\mu^{(s)}$, of length m/p, is stored in processor μ and has components $x_k^{(s)} = x_{s+kp}$. We also use the intermediate vectors

$$\mathbf{v}^{(s)} = \begin{pmatrix} \mathbf{v}_0^{(s)} \\ \mathbf{v}_1^{(s)} \\ \vdots \\ \mathbf{v}_{p-1}^{(s)} \end{pmatrix},$$

where $\mathbf{v}_\mu^{(s)}$, of length m/p, is stored in processor μ. In $CIRC$, we store the "charge" positions in a fixed vector $\mathbf{z}$, of length m with components $z_k = -2\pi k/m$.

Algorithm 3.4.1 $\mathbf{y} = FMDFT(t, h, \mathbf{x})$ evaluates the DFT of $\mathbf{x}$ approximately, using a fast multipole method.

1. **for** $s = 1, \ldots, p-1$:
   $\sigma^{(s)} = \sum_{k=0}^{m-1} x^{(s)}(k)$
   end

2. **for** $s = 1, \ldots, p-1$:
   comment [Multiply by $G^{(s)}$ approximately]
   $\mathbf{v}^{(s)} = CIRC(t, h, \mathbf{z}, \mathbf{z} + 2\pi s/n, \mathbf{x}^{(s)})$ with potential $\rho(x) = \cot(x/2)$
   end

3. **for** $s = 1, \ldots, p-1$:
   $\mathbf{v}^{(s)} = \gamma^{(s)}(\mathbf{v}^{(s)} + i\sigma^{(s)})$
   end

4. **comment** [Multiply by $F_p \otimes I_m$ in Equation (3.9), as local FFTs of length p.]
   $(\mathbf{v}^{(0)}, \mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(p-1)}) = (\mathbf{x}^{(0)}, \mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(p-1)})F_p$;
   comment [Perform distributed transpose.]
   $$\mathbf{y} = \begin{pmatrix} \mathbf{v}^{(0)} \\ \mathbf{v}^{(1)} \\ \vdots \\ \mathbf{v}^{(p-1)} \end{pmatrix}$$

5. **comment** [Multiply by $I_p \otimes F_m$ in Equation (3.9), as local FFT of length m.]

   $\mathbf{y}_\mu = F_m \mathbf{y}_\mu$

3.4.3 Accuracy

The only part of $FMDFT$ that is not exact in exact arithmetic is Step 2, where $CIRC$ from Section 2.4 is used to approximate $\mathbf{v}^{(s)} = G^{(s)}\mathbf{x}^{(s)}$. In order to minimize arithmetic complexity of $CIRC$, we choose the number of levels in the algorithm to be $h \approx \lg(s_{\text{opt}})$ with s_{opt} from Equation (2.46).

Estimate 3.4.2 *If h is chosen so that*

$$|h - \frac{1}{2}| \leq \lg(\frac{n}{pt}\sqrt{\frac{3}{10}}),$$

then the relative error in $FMDFT$ is given by

$$\frac{\|F_n\mathbf{x} - FMDFT(t,h,\mathbf{x})\|_2}{\|F_n\mathbf{x}\|_2} \leq \frac{23}{11^t}(\frac{n}{pt}).$$

Derivation. The choice of h implies

$$2^h \leq \frac{n}{pt}\sqrt{\frac{3}{5}} = \frac{m}{t}\sqrt{\frac{3}{5}}.$$

Hence Estimate 2.6.13 gives, for $s = 1, \ldots, p-1$:

$$\begin{aligned} \|G^{(s)}\mathbf{x}^{(s)} - CIRC(t,h,\mathbf{z},\mathbf{z}+\frac{2\pi s}{n})\|_2 &\lesssim \|\mathbf{x}^{(s)}\|_2 \cdot m \cdot \frac{8(5+4\sqrt{2})/3}{(17/3+4\sqrt{2})^t} \\ &= (\sqrt{\frac{3}{5}})\frac{8(5+4\sqrt{2})/3}{(17/3+4\sqrt{2})^t}\frac{m^2}{t}\|\mathbf{x}^{(s)}\|_2. \quad (3.12) \end{aligned}$$

We may write as a linear operator

$$\widetilde{G}^{(s)}\mathbf{x}^{(s)} = CIRC(t,h,\mathbf{z},\mathbf{z}+\frac{2\pi s}{n},\mathbf{x}^{(s)}),$$

and then (3.12) gives

$$\|G^{(s)} - \widetilde{G}^{(s)}\|_2 \lesssim (\sqrt{\frac{3}{5}})\frac{8(5+4\sqrt{2})/3}{(17/3+4\sqrt{2})^t}\frac{m^2}{t}. \tag{3.13}$$

Using Equations (3.9)-(3.11) and properties of matrix norms,

$$\begin{aligned}
\|F_n\mathbf{x} &- FMDFT(t,h,\mathbf{x})\|_2 \\
&\leq \|I_p \otimes F_m\|_2 \|F_p \otimes I_m\|_2 \max_s\{|\gamma^{(s)}|\|G^{(s)} - \widetilde{G}^{(s)}\|_2\}\|\mathbf{x}\|_2 \\
&\lesssim \sqrt{m}\cdot\sqrt{p}\cdot\frac{1}{m}\cdot(\sqrt{\frac{3}{5}})\frac{8(5+4\sqrt{2})/3}{(17/3+4\sqrt{2})^t}\frac{m^2}{t}\|\mathbf{x}\|_2 \\
&= (\sqrt{\frac{3}{5}})\frac{8(5+4\sqrt{2})/3}{(17/3+4\sqrt{2})^t}(\frac{n}{pt})\|F_n\mathbf{x}\|_2
\end{aligned}$$

using $m = n/p$, $\gamma^{(s)} \leq 1/m$ and $\|F_n\mathbf{x}\|_2 = \sqrt{n}\|\mathbf{x}\|_2$. □

3.4.4 Complexity

In $FMDFT$ we run $CIRC$ in parallel $p-1$ times in Step 2, on problems of size m. Since we run $CIRC$ on complex data, the flop counts are doubled.

Here is a step-by-step analysis of the number of flops used by each processor in $FMDFT$.

1. $(p-1)2m/p$

2. $(p-1)(55\frac{mt}{p} + 20t^2\lg p - 76t^2)$ (at most, from (2.49))

3. $(p-1)8m/p$

4. $m \cdot 5p\lg p/p = 5m\lg p$

   There are m FFT evaluations of size p.

5. $5m\lg m$

   In each processor there is an FFT evaluation of size m.

Adding up and replacing $m = n/p$, the total number of flops per processor is bounded above by

$$\frac{n}{p}[5\lg n + (1 - \frac{1}{p})(10 + 55t)] + t^2(20\lg p - 76)(p-1). \tag{3.14}$$

For $t = 16$, the number of flops per processor is

$$\frac{n}{p}(5\lg n + 890(1 - \frac{1}{p})) + 256(20\lg p - 76)(p-1).$$

For communication, we count *complex* scalars instead of real scalars as in Chapter 2. When using $CIRC$, communication is reduced because the charge positions are known to all processors and do not need to be sent.

Here is a step-by-step analysis of the maximum number of scalars sent by each processor in $FMDFT$:

1. $2(p-1)$, in 2 messages
2. $(p-1)t(4\lg(m/t) + \lg p - 5)$ (from (2.51) for $CIRC$), in $4\lg p - 3$ messages
3. 0
4. $\frac{m}{p}(p-1)$, in $p-1$ messages
5. 0

So the total number of scalars sent by each processor is

$$(p-1)[\frac{n}{p^2} + t(4\lg n - 3\lg p - 4\lg t - 5)], \tag{3.15}$$

which for $t = 16$ is

$$(p-1)[\frac{n}{p^2} + 16(4\lg n - 3\lg p - 21)]$$

The total number of messages required to be sent from each processor is at most $p + 4\lg p - 2$.

3.5 Experimental Results

We have implemented both the algorithm $FMDFT$ and a conventional parallel FFT algorithm, $PFFT$, based on the six-step framework described in Section 3.1. We use our implementations to show below that $FMDFT$ is accurate and that it can outperform conventional parallel FFT algorithms.

Before we proceed to present our experimental results, we would like to state the precise goal of our performance experiments. The experiments are intended to show that the performance of the two algorithms is within a small factor of each other, and that the relative speed of the two algorithms is determined by the communication-to-computation-rates ratio of the parallel computer on which they are executed. When the ratio is high, $PFFT$ is faster. When the ratio is low, $FMDFT$ is faster.

Our experiments are *not intended* to show that either of our implementations is a state-of-the-art code that is better than other parallel FFT codes. We do believe, however, that if both implementations are improved to a state-of-the-art level, our $FMDFT$ algorithm would still prove faster on machines with fast processors and a relatively slow communication network.

3.5.1 Performance Results

This section compares the performance of our implementations of $FMDFT$ and $PFFT$. Both algorithms are coded in Fortran 77. We use a publicly available FFT package, FFTPACK [28], for performing local FFTs on individual processors, and the Message Passing Interface (MPI) for interprocessor communication. The software is portable and runs without modifications on both the IBM SP2 scalable parallel computer and a cluster of Sun UltraSparc symmetric multiprocessors (SMPs).

The first set of experiments were conducted on an IBM SP2 parallel computer [2]. The machine was configured with so-called thin nodes with 128 Mbytes of main memory. Thin nodes have a 66.7 MHz POWER2 processor [33], 64 Kbytes 4-way set associative level-1 data-cache, no level-2 cache, and a 64-bit-wide main memory bus. They have smaller data paths between the cache and the floating-point units than

all other POWER2-based SP2 nodes. The system software that we used includes the AIX version 4.1.3 operating system, Parallel Environment version 2.1 (this includes the message passing library), and the XLF version 3.2.5 Fortran compiler.

The computation-to-communication balance of the SP2 can be summarized as follows. The peak floating-point performance of POWER2-based nodes is 266 million flops per second, thanks to two floating-point functional units that can each execute a multiply-add operation in every cycle. While many dense matrix operations run on these nodes at close to peak performance [1], FFT codes run at lower rates. Large power-of-two one-dimensional FFTs from FFTPACK run at 20–30 megaflops per second, and similar routines from IBM's Engineering and Scientific Subroutine Library (ESSL) run at 75–100 megaflops per second. When the message passing libraries use the SP2's High-Performance Switch (a specialized interconnection network) using the so-called user-space communication protocol, which bypasses the operating system, the communication bandwidth they can achieve is at most 41 Mbytes per second per node. When the libraries use the High-Performance Switch using the internet protocol (IP), which does not bypass the operating system, the communication bandwidth is at most 13 Mbytes per second per node. When the libraries use IP over Ethernet rather than over the High-Performance switch, the bandwidth is even lower, 1.25 Mbytes per second for all the nodes combined.

The running times that are reported here are averages of 10 runs. We ran each experiment 11 times, always discarding the first run, which incurs various startup costs. We also discarded runs in which the running time was more than twice than the smallest running time for that experiment, which happened only once. We averaged the other 10 runs (9 in one case). The relative standard deviations were less than 3% when we used user-space communication over the High-Performance Switch, less than 9% when we used internet protocol over the High-Performance Switch, and less than 20% when we used internet protocol over Ethernet.

The results of our experiments are summarized in Table 3.3 and plotted in Figure 3-4. The results show that $PFFT$ is faster when we use the two faster communication mechanisms, and that $FMDFT$ is faster with the slowest communication

mechanism, IP over Ethernet. The absolute running times using Ethernet are very slow. Ethernet is also the only communication mechanism that does not allow additional processors to reduce the absolute running times, since it is a broadcast mechanism in which the total bandwidth does not grow with the number of processors. The High-Performance Switch allows additional processors to decrease the absolute running times of both algorithms.

Table 3.3 also shows that $PFFT$ is more sensitive to degradation in communication bandwidth. For example, on an FFT of 1048576 points on 4 processors, the running time of $PFFT$ increased by 0.932 seconds when we switched from user-space to IP communication over the HPS, but the running time of $FMDFT$ increased by only 0.423 seconds. The relative increases are 36% and 8%, respectively.

Table 3.4 describes the experiments with the best communication mechanism in more detail. The table shows that $PFFT$ achieves good speedups. The speedups for the largest problems on 2, 4, and 8 processors are 1.45, 2.66, and 4.77 respectively (where the speedup is defined as $p \cdot T_{\text{fft}}/T$). The volume of communication and the time spent in communication in $FMDFT$ are smaller by a factor of 2–3 than the volume and time spent by $PFFT$. The part spent in computations other than local FFTs is much larger, however, in $FMDFT$. The data from Table 3.4 are also graphed in Figure 3-5.

With a flop rate of FR (in flops per second) and a communications bandwidth of BW (in bytes per second), the times we should *expect* for $PFFT$ are:

$$T_{\text{fftloc}} = (5\frac{n}{p}\lg n)/FR \tag{3.16}$$

$$T_{\text{arith}} = (6\frac{n}{p})/FR \tag{3.17}$$

$$T_{\text{comm}} = [3\frac{n}{p}(1-\frac{1}{p})] \cdot (16 \text{ bytes})/BW \tag{3.18}$$

The last two rows of Table 3.4 show the minimum and maximum ratio of the actual times recorded to the times expected with $FR = 266$ Mflops/sec and $BW = 41$ Mbytes/sec.

For $FMDFT$, Equations (3.14) and (3.15) with $t = 16$ give expected times of:

$$T_{\text{fftloc}} = (5\frac{n}{p}\lg\frac{n}{p})/FR \tag{3.19}$$

$$T_{\text{arith}} = (890\frac{n}{p}(1-\frac{1}{p}))/FR \tag{3.20}$$

$$T_{\text{comm}} = [(p-1)(\frac{n}{p^2}+64\lg n-48\lg p-400)] \cdot (16 \text{ bytes})/BW \tag{3.21}$$

As with $PFFT$, the last two rows of Table 3.4 show the minimum and maximum ratios of actual to expected times with $FR = 266$ Mflops/sec and $BW = 41$ Mbytes/sec.

We have also conducted experiments on a cluster of nine Sun Ultra Enterprise 5000 servers connected by an Ethernet switch. These servers use UltraSPARC processors with a peak floating-point performance of 333 Mflops/sec. Although each server contains eight UltraSPARC processors, our experiments used only one processor per server. The maximum observed bandwidth of the Ethernet switch was approximately 1.25 Mbytes/second for all nodes.

Table 3.5 and Figure 3-6 summarize the results of our experiments on the Sun Ultra cluster. As on the SP2, we ran each experiment 11 times, discarding the first run, and averaged the other 10 runs. Relative standard deviations for the arithmetic portions were less than 15% in all but four cases, which ran as high as 30%. Because of fluctuations in traffic on the cluster, relative standard deviations in communication time were as high as 53%.

3.5.2 Extrapolation to Other Machines

Our results have shown that when we use Ethernet as an interconnect for SP2 nodes or UltraSPARC processor servers, $FMDFT$ outperforms a conventional FFT. While Ethernet cannot be considered an appropriate communication medium for high-performance scientific computing, high-performance machines with similar communication-to-computation-rates ratio do exist and are likely to be popular platforms in the future. Equations (3.16)–(3.21) show that the cutoff ratio is 0.036 bytes/flop.

Let us consider a cluster of symmetric multiprocessors connected with a fast com-

modity network. Such a configuration might consist, for example, of several Sun Ultra Enterprise servers using 8 UltraSparc processors each, connected by an ATM switch. The peak floating-point performance of each node (if all processors are used) is about 2.5 Gflops. Measurements made by Bobby Blumofe with Sun Sparc workstations connected by a Fore ATM switch have shown that the application-to-application communication bandwidth of the switch is about 5 Mbytes per second per node in one direction (the nominal peak bandwidth of this network is 155 Mbits per second). Even if the network can support 5 Mbytes/sec in both directions, the communication-to-computation-rates ratio is only 0.002 bytes/flop.

A cluster of Digital AlphaServers connected by Digital's GIGAswitch/FDDI network yields a similar communication-to-computation ratio. The nodes can have up to 12 processors each, with peak floating-point performance of 600-874 Mflops/sec each. Digital has measured the bandwidth of the network at about 11.9 Mbytes per second [6]. With nodes consisting of twelve 600 Mflops/sec processors each, the ratio is 0.0017 bytes/flop.

The ratio in our SP2 experiments with Ethernet is about 0.0022 bytes/flop when we use 2 nodes, 0.0010 with 4 nodes, and 0.0005 with 8 nodes. The peak performance of each node is 266 Mflops and the measured communication bandwidths are about 580, 270, and 132 Kbytes per second per node with 2, 4, and 8 nodes. In our Ultra cluster experiments, the ratio is approximately 0.002 bytes/flop with 2 nodes, 0.0009 with 4 nodes, and 0.0005 with 8 nodes. Each node has a peak performance of 333 Mflops/sec, and the communication bandwidth is approximately 1.25 Mbytes for all nodes combined.

Since $FMDFT$ outperformed the $PFFT$ by a large margin even on two processors on the SP2, when the ratio is 0.0022 bytes/flop, it seems safe to predict that $FMDFT$ would outperform a conventional FFT on the above-mentioned clusters where ratios are even lower.

If we assume that tuning both algorithms would improve the performance of their local computations by a factor of three, say, then $FMDFT$ would outperform a conventional FFT even if the networks of the clusters improved by a similar factor.

This assumption is supported by the fact that a tuned high-performance local FFT routine (in ESSL) is about 3.75 times faster than the publicly available package that we used (FFTPACK).

3.6 Conclusions

The results of our experiments on the SP2 and the Ultra cluster have shown that when the communication-to-computation-rates ratio is low, $FMDFT$ outperforms a conventional parallel FFT by more than a factor of two. Quantitative performance extrapolation indicates that the new algorithm would also be faster on state-of-the art clusters of symmetric multiprocessors.

The $FMDFT$ algorithm is faster when communication dominates the running time of conventional parallel FFTs. When communication is so expensive, neither $PFFT$ nor $FMDFT$ is likely to be very efficient when compared to a uniprocessor FFT. That is, their speedups are likely to be modest. There are at least two reasons to believe that the new algorithm would prove itself useful even when speedups are modest. First, in many applications the main motivation to use parallel machines is the availability of large memories, and not necessarily parallel speedups. In other words, it may be necessary to compute FFTs on multiple nodes because the data does not fit within the main memory of one node. Second, an FFT with a small or no speedup can be a part of a larger application which exhibits a good overall speedup. The application might include, for example, FFTs as well as grid computations, which require less communication per floating-point operation than the FFTs. In both cases, accelerating the parallel FFTs contributes to the performance of the application, whereas switching to a single-node FFT is not feasible.

We believe that improvements in the new algorithms that would reduce the amount of local arithmetic it performs are possible. Such improvements would make $FMDFT$ faster than a conventional parallel FFT on machines with higher communication-to-computation-rates ratios than those we have indicated in this chapter.

Table 3.3: A comparison of the performance of $PFFT$ and $FMDFT$ on an SP2 parallel computer using three communication mechanisms. Running times are in seconds. The three communication mechanisms that were used are user-space communication over the High-Performance Switch (US-HPS), internet protocol over the High-Performance Switch (IP-HPS), and internet protocol over Ethernet (IP-EN). The last two rows give the minimum and maximum ratios of the timings reported in the table to what one would expect from the sum of Equations (3.16)–(3.18) for T_C, or Equations (3.19)–(3.21) for T_N.

		Communication mechanism					
	Size	US-HPS		IP-HPS		IP-EN	
p	n	$PFFT$	$FMDFT$	$PFFT$	$FMDFT$	$PFFT$	$FMDFT$
2	32768	0.113	0.199	0.164	0.219	0.769	0.443
	65536	0.220	0.399	0.301	0.432	1.526	0.857
	131072	0.471	0.833	0.633	0.888	3.083	1.725
	262144	1.043	1.761	1.354	1.869	6.250	3.535
	524288	2.545	3.987	3.154	4.197	12.976	7.479
4	32768	0.059	0.128	0.109	0.152	1.213	0.602
	65536	0.116	0.268	0.199	0.302	2.368	1.198
	131072	0.220	0.563	0.355	0.614	5.928	2.528
	262144	0.469	1.171	0.719	1.264	11.474	4.902
	524288	1.033	2.441	1.504	2.605	18.540	8.726
	1048576	2.608	5.355	3.540	5.778	37.020	17.000
8	32768	0.031	0.077	0.061	0.101	1.708	1.263
	65536	0.070	0.150	0.114	0.179	3.166	2.117
	131072	0.140	0.296	0.266	0.358	7.225	3.196
	262144	0.265	0.593	0.446	0.681	12.983	5.691
	524288	0.556	1.288	0.866	1.410	22.165	10.097
	1048576	1.172	2.704	1.770	2.924	42.093	17.827
	2097152	2.823	5.926	3.926	6.320	85.428	33.783
	min ratio	5.503	4.984	3.482	4.597	1.193	1.356
	max ratio	10.060	6.890	5.405	6.087	1.639	1.918

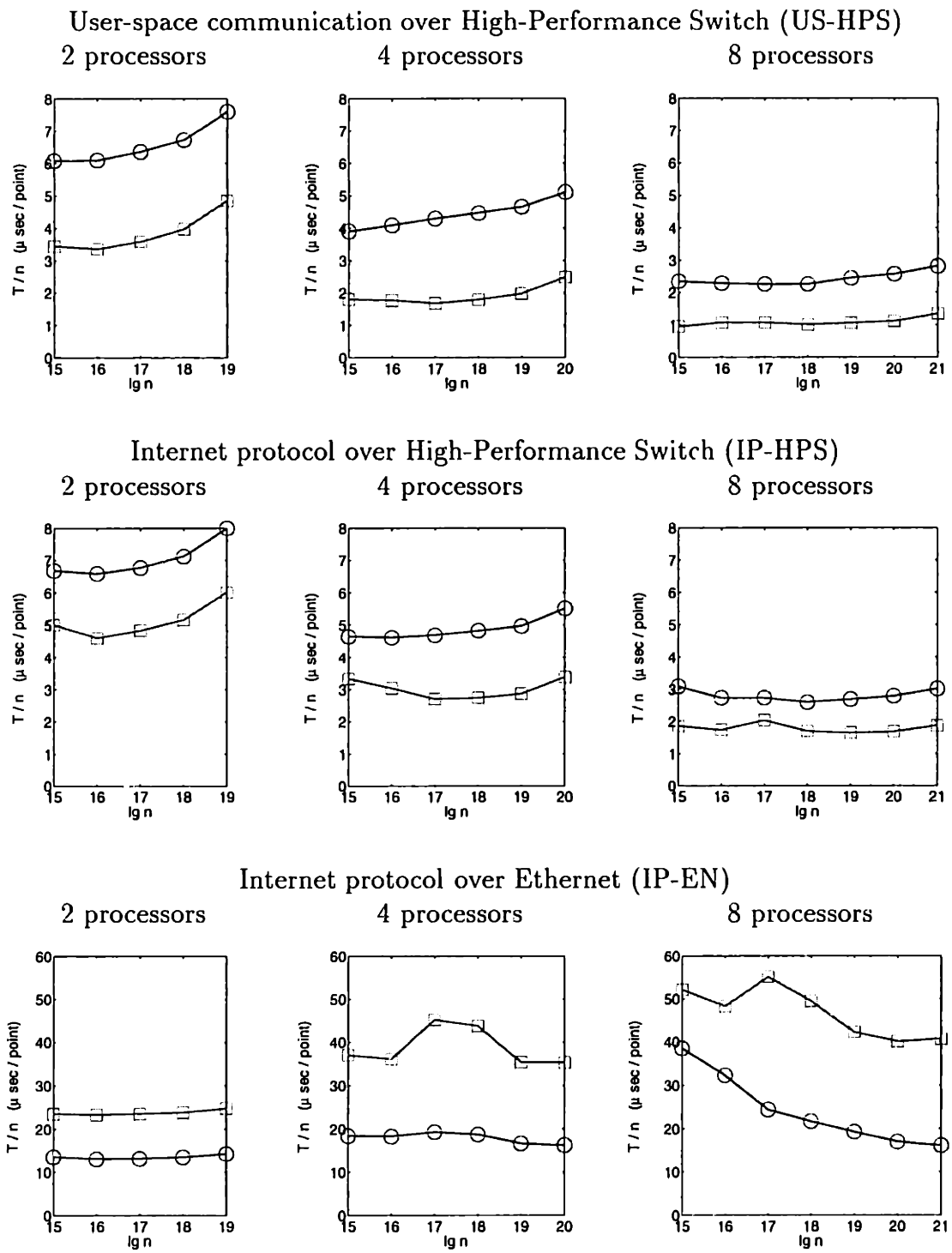


Figure 3-4: Normalized times on the SP2, in microseconds per point. The times for $PFFT$ are indicated with "□". The times for $FMDFT$ are indicated with "○". These graphs contain the same data as Table 3.3.

Table 3.4: A comparison of the performance of $PFFT$ and $FMDFT$ on an SP2 parallel computer. The communication software used the High-Performance Switch without operating system overhead (US-HPS). Mean times are reported in seconds. The total time is divided into three parts: T_{fftloc} spent in Netlib local FFTs, T_{comm} used for communication, and T_{arith} for other arithmetic. The last two rows give the minimum and maximum ratios of the timings reported in the table to what one would expect from Equations (3.16)–(3.21).

		$PFFT$				$FMDFT$			
Formula →			(3.16)	(3.17)	(3.18)		(3.19)	(3.20)	(3.21)
p	n	T	T_{fftloc}	T_{arith}	T_{comm}	T	T_{fftloc}	T_{arith}	T_{comm}
2	32768	0.113	0.067	0.018	0.029	0.199	0.046	0.141	0.012
	65536	0.220	0.129	0.035	0.055	0.399	0.086	0.292	0.021
	131072	0.471	0.295	0.070	0.105	0.833	0.211	0.583	0.039
	262144	1.043	0.690	0.141	0.212	1.761	0.521	1.166	0.075
	524288	2.545	1.846	0.282	0.417	3.987	1.502	2.338	0.147
4	32768	0.059	0.026	0.008	0.024	0.128	0.020	0.097	0.011
	65536	0.116	0.059	0.015	0.042	0.268	0.047	0.203	0.018
	131072	0.220	0.113	0.030	0.076	0.563	0.088	0.443	0.033
	262144	0.469	0.263	0.060	0.146	1.171	0.213	0.902	0.056
	524288	1.033	0.626	0.121	0.286	2.441	0.533	1.804	0.104
	1048576	2.608	1.724	0.305	0.580	5.355	1.524	3.635	0.197
8	32768	0.031	0.011	0.004	0.016	0.077	0.008	0.056	0.012
	65536	0.070	0.025	0.014	0.031	0.150	0.019	0.112	0.018
	131072	0.140	0.057	0.028	0.055	0.296	0.047	0.223	0.026
	262144	0.265	0.110	0.056	0.100	0.593	0.087	0.464	0.043
	524288	0.556	0.253	0.109	0.193	1.288	0.214	1.003	0.070
	1048576	1.172	0.608	0.219	0.344	2.704	0.519	2.052	0.133
	2097152	2.823	1.683	0.468	0.672	5.926	1.513	4.165	0.249
	min ratio	5.503	9.525	40.588	2.485	4.984	8.659	4.649	2.537
	max ratio	10.060	19.717	79.147	3.813	6.890	17.182	5.526	4.734

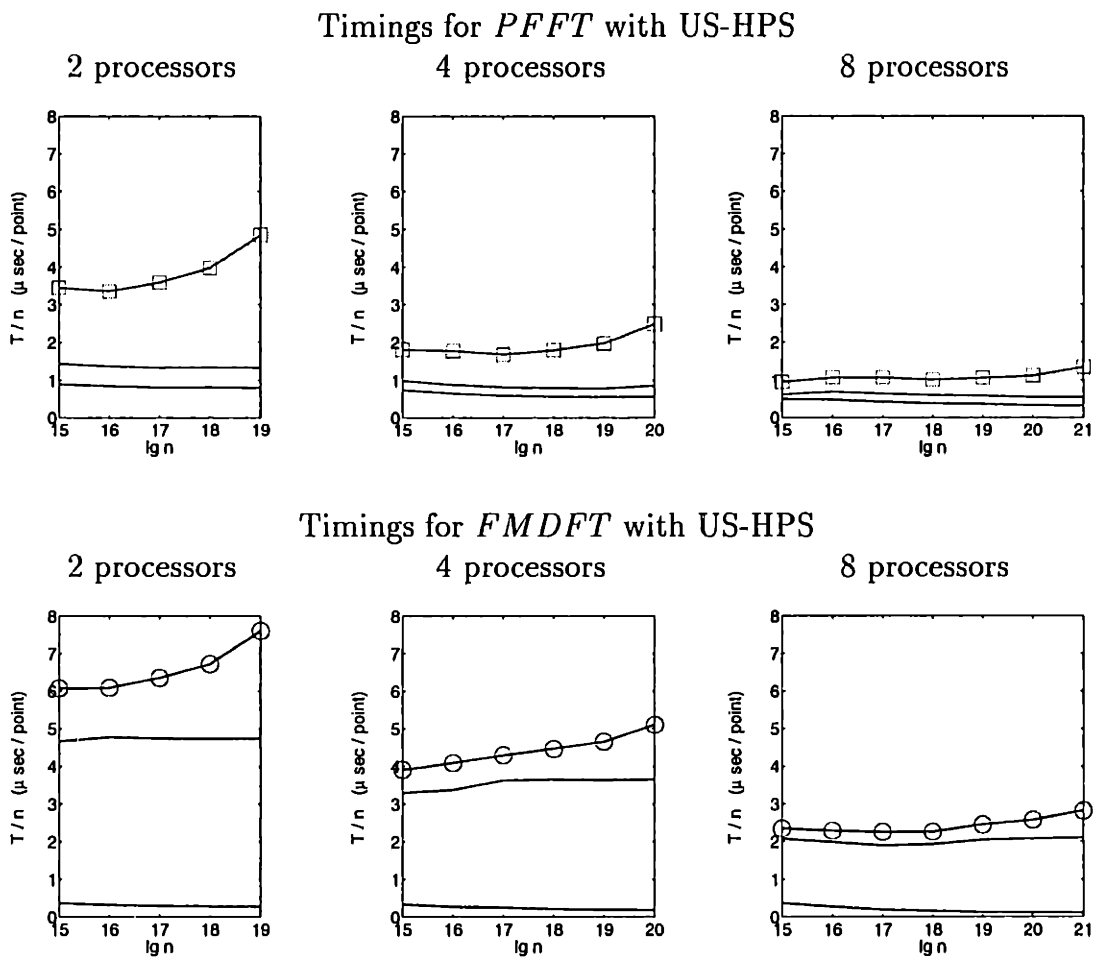
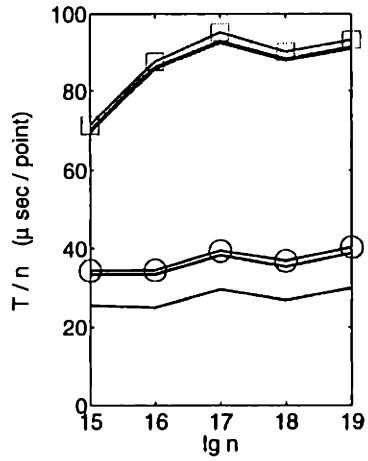


Figure 3-5: Normalized times on the SP2 (microseconds per point) with the High-Performance Switch without operating system overhead (US-HPS). The times for $PFFT$ are indicated with "□". The times for $FMDFT$ are indicated with "○". In each graph, the bottom component is the communication time, the top component is for the local FFT of size n/p, and the middle component is the time for all other arithmetic. These graphs contain the same data as Table 3.4.

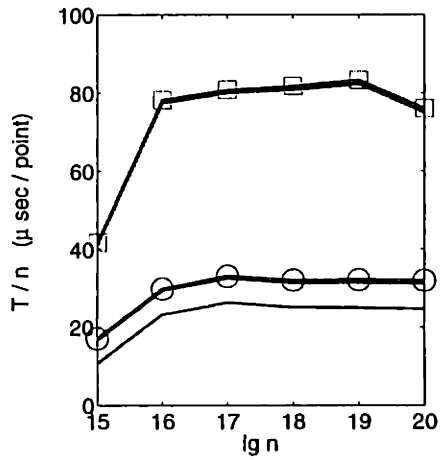
Table 3.5: A comparison of the performance of the two algorithms on a cluster of servers of UltraSPARC processors. Mean times are reported in seconds. The total time is divided into three parts: T_{fftloc} spent in Netlib local FFTs, T_{comm} used for communication, and T_{arith} for other arithmetic. The last two rows give the minimum and maximum ratios of the timings reported in the table to what one would expect from Equations (3.16)–(3.21).

		PFFT				*FMDFT*			
Formula →			(3.16)	(3.17)	(3.18)		(3.19)	(3.20)	(3.21)
p	n	T	T_{fftloc}	T_{arith}	T_{comm}	T	T_{fftloc}	T_{arith}	T_{comm}
2	32768	2.341	0.041	0.015	2.285	1.125	0.030	0.260	0.835
	65536	5.738	0.092	0.031	5.615	2.260	0.070	0.553	1.637
	131072	12.386	0.203	0.063	12.120	5.165	0.158	1.122	3.885
	262144	23.632	0.470	0.126	23.036	9.663	0.381	2.248	7.034
	524288	48.895	0.972	0.254	47.669	21.118	0.795	4.595	15.728
4	32768	1.366	0.016	0.007	1.343	0.558	0.015	0.193	0.351
	65536	5.127	0.041	0.015	5.071	1.957	0.032	0.402	1.524
	131072	10.603	0.092	0.031	10.479	4.341	0.077	0.818	3.447
	262144	21.447	0.200	0.062	21.185	8.403	0.168	1.654	6.581
	524288	43.711	0.444	0.124	43.143	16.878	0.392	3.368	13.119
	1048576	79.827	0.951	0.248	78.628	33.573	0.833	6.860	25.881
8	32768	1.677	0.007	0.003	1.667	1.028	0.006	0.113	0.909
	65536	5.460	0.019	0.006	5.435	2.274	0.014	0.226	2.034
	131072	12.537	0.039	0.014	12.484	5.945	0.031	0.459	5.455
	262144	23.825	0.089	0.031	23.706	10.781	0.076	0.974	9.732
	524288	33.214	0.196	0.062	32.956	13.682	0.167	1.950	11.565
	1048576	61.447	0.480	0.123	60.843	26.096	0.433	3.931	21.732
	2097152	115.096	0.953	0.250	113.893	41.130	0.855	7.896	32.379
	min ratio	1.444	7.588	40.649	1.423	1.384	8.130	11.753	0.910
	max ratio	4.887	13.268	53.776	4.816	5.563	12.942	13.117	4.660

2 processors
on Sun cluster



4 processors
on Sun cluster



8 processors
on Sun cluster

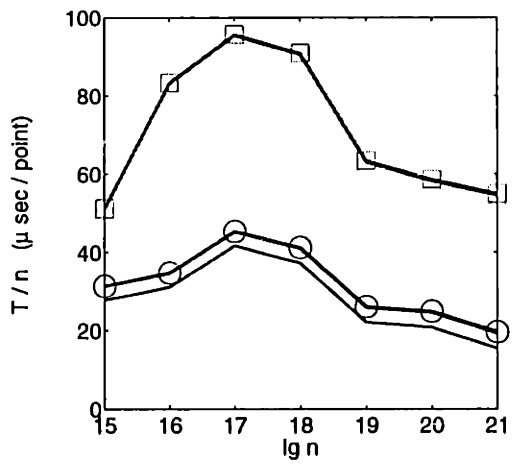


Figure 3-6: Normalized times on the Sun cluster, in microseconds per point. The times for $PFFT$ are indicated with "□". The times for $FMDFT$ are indicated with "○". For each algorithm in each graph, the bottom component is the communication time, the top component is for the local FFT of size n/p, and the middle component is the time for all other arithmetic. These graphs contain the same data as Table 3.5.

Chapter 4

Comparison of Two-Dimensional Methods

4.1 Approaches to the problem

This chapter is concerned with computing (1.1) in the complex plane with the two-dimensional Coulomb potential $\rho(z) = \log|z|$. We study the case in which the potential is to be evaluated at the charge positions; this corresponds to $\mathbf{x} = \mathbf{y}$ in (1.1).

With charges at positions represented by the complex numbers $z_1, \ldots, z_N$ the problem of this chapter is that of finding

$$f_j = \sum_{\substack{k=1 \\ k \neq j}}^{N} q_k \log|z_k - z_j| = \Re\mathrm{e} \sum_{\substack{k=1 \\ k \neq j}}^{N} q_k \log(z_k - z_j). \tag{4.1}$$

We compare three algorithms with fast multipole-type structure:

- the original Fast Multipole Method of Greengard and Rokhlin [14], described in Section 4.4;
- Anderson's method [3] based on Poisson's formula, described in Section 4.5;
- our new method based on complex polynomial interpolation, described in Section 4.6.

The common computational structure of these methods is described in Section 4.2, and a discussion of the general arithmetic complexity is given in Section 4.3. In Section 4.7, we show precisely how the approximations used in all three methods are obtained from Taylor series. In Section 4.8, we report the results of numerical experiments. These results indicate that, without optimizations, the Greengard–Rokhlin and Anderson methods have virtually identical performance. The performance of the polynomial interpolation method depends on the radii of the circles containing the interpolation points: if high accuracy is not needed, then the radii can be chosen so that the method is as efficient and accurate as Greengard–Rokhlin and Anderson. Numerical instabilities, however, result when a large number of points are used. This problem can be avoided by choosing different radii, but the resulting robust polynomial interpolation algorithm is less efficient for specified accuracy than the Greengard–Rokhlin and Anderson methods.

4.2 Structure of fast multipole-type methods in two dimensions

We follow the same FMM framework as outlined in Chapter 1.

The algorithms of this chapter take the form $POT2D(p, h, \mathbf{z}, \mathbf{q})$, where the parameters are:

- p is the length of expansions to be used in the algorithm;
- h is the maximum number of levels to be used in the algorithm;
- $\mathbf{z}$ is a complex vector containing the charge positions $z_1, \ldots, z_N$, which are taken to lie inside the square in the complex plane with corners $1+i, 1-i, -1+i, -1-i$;
- $\mathbf{q}$ is a real vector of length N with q_k being the strength of the charge at point z_k.

We use the following notation for each level $l = 2, \ldots, h$:

- $w_l = 1/2^l$ is the half-width of boxes at level l;
- for $a, b \in \{1, \ldots, 2^l\}$:
  - $B^l_{a,b}$ is the square box of side length $2w_l$ centered at

$$(2b-1)w_l - 1 + ((2a-1)w_l - 1)i$$

    and with sides parallel to the axes;
  - $J(l, a, b)$ is the set of indices of boxes adjacent to $B^l_{a,b}$:

$$J(l, a, b) = \{(a', b') \neq (a, b) : |a' - a| \leq 1 \& |b' - b| \leq 1\};$$

  - $I(l, a, b)$ is the it interaction list of boxes at level l that are the children of the neighbors of the parent of $B^l_{a,b}$ but are not themselves neighbors of $B^l_{a,b}$.

We define $s = 2^h$, and for $a, b \in \{1, \ldots, s\}$, we define the following for the boxes at level h:

- $c_{a,b} = (2b-1)w_h - 1 + ((2a-1)w_h - 1)i$ is the center of $B^h_{a,b}$;
- $\mathcal{Z}_{a,b} = \{k : z_k \in B^h_{a,b}\}$ is the set of indices of charges in $B^h_{a,b}$;

For $l = 2, \ldots, h$ and $a, b \in \{1, \ldots, 2^l\}$, the algorithm defines two expansions representing potentials:

- $\Phi^{(l)}_{a,b}$ is a p-vector containing coefficients of the far-field expansion for box $B^l_{a,b}$;
- $\Psi^{(l)}_{a,b}$ is a p-vector containing coefficients of the local expansion for box $B^l_{a,b}$.

If either a or b is not in $\{1, \ldots, 2^l\}$, then $\Phi^{(l)}_{a,b}$ and $\Psi^{(l)}_{a,b}$ are taken to be zero.

In approximating the function

$$f(z) = \sum_j q_j \log |z - z_j| \tag{4.2}$$

for a box centered at 0 at level l, the far-field expansion is valid for some $|z| \geq 3w_l$ when charge positions satisfy $|z_j| \leq w_l\sqrt{2}$, and the local expansion is valid for some $|z| \leq w_l\sqrt{2}$ when charge positions satisfy $|z_j| \geq 3w_l$. See Figure 4-1 for an illustration.

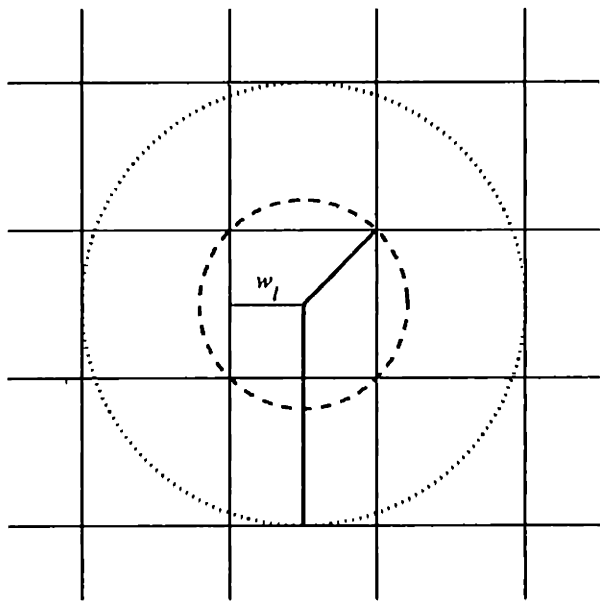


Figure 4-1: Points in a box at level l have maximum distance $w_l\sqrt{2}$ from the center. The closest point in the interaction list of a box has distance $3w_l$ from the center.

To convert between expansions at different levels, the algorithm uses the following operators which depend on the particular expansions used and are fixed by p:

- For $l = 2, \ldots, h-1$, $M_{SW}^{(l)}$, $M_{NW}^{(l)}$, $M_{SE}^{(l)}$ and $M_{NE}^{(l)}$ are $p \times p$ matrices for obtaining far-field expansions for boxes at level l from far-field expansions for their children at level $l+1$;

- For $l = 3, \ldots, h$, $S_{SW}^{(l)}$, $S_{NW}^{(l)}$, $S_{SE}^{(l)}$ and $S_{NE}^{(l)}$ are $p \times p$ matrices for obtaining local expansions for boxes at level l from local expansions for their parent at level $l-1$;

- For $l = 2, \ldots, h$ and for $\Delta a, \Delta b \in \{-3, -2, -1, 0, 1, 2, 3\}$, $T_{\Delta a, \Delta b}^{(l)}$ is a $p \times p$ matrix for obtaining a local expansion for a box at level l from the far-field expansion of a box in $I(l, a, b)$.

The function $POT2D_INIT(p, h, \mathbf{z})$ initializes matrices that depend on the charge positions $\mathbf{z}$. For each $a, b \in \{1, \ldots, s\}$:

- $G_{a,b}$ is a $p \times |\mathcal{Z}_{a,b}|$ matrix for obtaining far-field expansion coefficients from the strengths of the charges in $B^l_{a,b}$.
- $E_{a,b}$ is a $|\mathcal{Z}_{a,b}| \times p$ matrix for obtaining potentials at evaluation points in $B^l_{a,b}$ from local expansion coefficients.
- $D_{a,b}$ is a $|\mathcal{Z}_{a,b}| \times |\mathcal{Z}_{I(h,a,b)}|$ matrix for obtaining potentials at evaluation points in $B^h_{a,b}$ due to charges in $B^h_{a,b}$ and its neighbors:

$$D_{a,b}(j,k) = \log|z_j - z_k| \tag{4.3}$$

  $D_{a,b}$ uses only the rows with indices in $\mathcal{Z}_{a,b}$ and the columns with indices in $\mathcal{Z}_{J(h,a,b)}$.

Algorithm 4.2.1

$\mathbf{f} = POT2D(p, h, \mathbf{z}, \mathbf{q})$ evaluates $f_j = \sum_{k \neq j} q_k \log|z_j - z_k|$, using expansions with p terms and with h hierarchy levels.

function $POT2D_INIT(p, h, \mathbf{z})$:

$s = 2^h$.

for $a = 1, \ldots, s$ and $b = 1, \ldots, s$: Compute $G_{a,b}$, $E_{a,b}$, and $D_{a,b}$.

end function $POT2D_INIT$.

function $\mathbf{f} = POT2D_EVAL(\mathbf{q})$:

1. **comment** [Get coefficients of far-field expansions.]

$$\begin{pmatrix} \Phi^{(h)}_{1,1} & \cdots & \Phi^{(h)}_{1,s} \\ \vdots & \ddots & \vdots \\ \Phi^{(h)}_{s,1} & \cdots & \Phi^{(h)}_{s,s} \end{pmatrix} = \begin{pmatrix} G_{1,1}\mathbf{q}_{\mathcal{Z}_{1,1}} & \cdots & G_{1,s}\mathbf{q}_{\mathcal{Z}_{1,s}} \\ \vdots & \ddots & \vdots \\ G_{s,1}\mathbf{q}_{\mathcal{Z}_{s,1}} & \cdots & G_{s,s}\mathbf{q}_{\mathcal{Z}_{s,s}} \end{pmatrix}$$

2. **comment** [Get coefficients of local expansions.]

$$\begin{pmatrix} \Psi_{1,1}^{(h)} & \cdots & \Psi_{1,s}^{(h)} \\ \vdots & \ddots & \vdots \\ \Psi_{s,1}^{(h)} & \cdots & \Psi_{s,s}^{(h)} \end{pmatrix} = POT2D_RECUR(h, \begin{pmatrix} \Phi_{1,1}^{(h)} & \cdots & \Phi_{1,s}^{(h)} \\ \vdots & \ddots & \vdots \\ \Phi_{s,1}^{(h)} & \cdots & \Phi_{s,s}^{(h)} \end{pmatrix})$$

3. **comment** [Evaluate potentials from local expansions and direct interactions.]

   for $a = 1, \ldots, s$:

   for $b = 1, \ldots, s$:

   $f_{z_b} = \Re\mathrm{e}\,(E_{a,b}\Psi_{a,b}^{(h)}) + D_{a,b}\mathbf{q}_{z_{J(h,a,b)}}$

   end

   end

end function $POT2D_EVAL$.

function $\begin{pmatrix} \Psi_{1,1}^{(h)} & \cdots & \Psi_{1,s}^{(h)} \\ \vdots & \ddots & \vdots \\ \Psi_{s,1}^{(h)} & \cdots & \Psi_{s,s}^{(h)} \end{pmatrix} = POT2D_RECUR(h, \begin{pmatrix} \Phi_{1,1}^{(h)} & \cdots & \Phi_{1,s}^{(h)} \\ \vdots & \ddots & \vdots \\ \Phi_{s,1}^{(h)} & \cdots & \Phi_{s,s}^{(h)} \end{pmatrix})$:

if $l = 2$ **then**

comment [No parents to take from.]

for $a = 1, 2, 3, 4$:

for $b = 1, 2, 3, 4$:

$\Psi_{a,b}^{(2)} = \sum_{(a',b') \in I(2,a,b)} T_{a-a',b-b'}^{(2)} \Phi_{a',b'}^{(2)}$

end

end

else

1. **comment** [Compute coefficients of far-field expansions at level $l-1$.]

   for $a = 1, \ldots, 2^{l-1}$:

   for $b = 1, \ldots, 2^{l-1}$:

   $\tilde{\Phi}_{a,b} = M_{SW}^{(l-1)}\Phi_{2a-1,2b-1}^{(l)} + M_{NW}^{(l-1)}\Phi_{2a,2b-1}^{(l)} + M_{SE}^{(l-1)}\Phi_{2a-1,2b}^{(l)} + M_{NE}^{(l-1)}\Phi_{2a,2b}^{(l)}$

   end

end

2. **comment** [Get coefficients of local expansions at level $l-1$.]

$$\begin{pmatrix} \tilde{\Psi}_{1,1} & \cdots & \tilde{\Psi}_{1,2^l} \\ \vdots & \ddots & \vdots \\ \tilde{\Psi}_{2^l,1} & \cdots & \tilde{\Psi}_{2^l,2^l} \end{pmatrix} = POT2D_RECUR(l-1, \begin{pmatrix} \tilde{\Phi}_{1,1} & \cdots & \tilde{\Phi}_{1,2^l} \\ \vdots & \ddots & \vdots \\ \tilde{\Phi}_{2^l,1} & \cdots & \tilde{\Phi}_{2^l,2^l} \end{pmatrix}).$$

3. **comment** [Compute coefficients of local expansions at level l.]

for $a = 1, \ldots, 2^{l-1}$:

for $b = 1, \ldots, 2^{l-1}$:

$$\begin{aligned}
\Psi^{(l)}_{2a-1,2b-1} &= S^{(l)}_{SW}\tilde{\Psi}_{a,b} + \textstyle\sum_{(a',b')\in I(l,2a-1,2b-1)} T^{(l)}_{2a-1-a',2b-1-b'}\Phi^{(l)}_{a',b'} \\
\Psi^{(l)}_{2a,2b-1} &= S^{(l)}_{NW}\tilde{\Psi}_{a,b} + \textstyle\sum_{(a',b')\in I(l,2a,2b-1)} T^{(l)}_{2a-a',2b-1-b'}\Phi^{(l)}_{a',b'} \\
\Psi^{(l)}_{2a-1,2b} &= S^{(l)}_{SE}\tilde{\Psi}_{a,b} + \textstyle\sum_{(a',b')\in I(l,2a-1,2b)} T^{(l)}_{2a-1-a',2b-b'}\Phi^{(l)}_{a',b'} \\
\Psi^{(l)}_{2a,2b} &= S^{(l)}_{NE}\tilde{\Psi}_{a,b} + \textstyle\sum_{(a',b')\in I(l,2a,2b)} T^{(l)}_{2a-a',2b-b'}\Phi^{(l)}_{a',b'}
\end{aligned}$$

end

end

end

end function *POT2D_RECUR*.

4.3 Complexity of *POT2D_EVAL*

For $POT2D_RECUR(l, \Phi^{(l)}_1, \ldots, \Phi^{(l)}_{2^l})$, we count separately:

- $N_M(l)$, the number of multiplications by the matrices $M^{(l)}_{SW}, M^{(l)}_{NW}, M^{(l)}_{SE}$, and $M^{(l)}_{NE}$;
- $N_S(l)$, the number of multiplications by the matrices $S^{(l)}_{SW}, S^{(l)}_{NW}, S^{(l)}_{SE}$, and $S^{(l)}_{NE}$;
- $N_T(l)$, the number of multiplications by the matrices $T^{(l)}_{\Delta a,\Delta b}$.

For $l = 2$, we have $N_M(2) = 0$ and $N_S(2) = 0$. Note that

$$N_T(2) = \sum_{a,b\in\{1,2,3,4\}} |I(2,a,b)|.$$

Adding up over the types of boxes at level 2:

- If (a, b) is one of the four corner boxes, then $|I(2, a, b)| = 4^2 - 2^2 = 12$.
- If (a, b) is one of the eight other edge boxes, then $|I(2, a, b)| = 4^2 - 2 \cdot 3 = 10$.
- If (a, b) is one of the four interior boxes, then $|I(2, a, b)| = 4^2 - 3^2 = 7$.

Hence:

$$N_M(2) = 0, \quad N_S(2) = 0. \quad N_T(2) = 4 \cdot 12 + 8 \cdot 10 + 4 \cdot 7 = 156. \tag{4.4}$$

For $l > 2$, we have $N_M(l) = 4^l + N_M(l-1)$ from step 1 and the recursion, and $N_S(l) = 4^l + N_S(l-1)$ from step 3 and the recursion. For $N_T(l)$, add up over the types of boxes at level l:

- If (a, b) is one of the four corner boxes, then $|I(2, a, b)| = 4^2 - 2^2 = 12$.
- If (a, b) is one of the four interior boxes that are children of corner boxes at level $l - 1$, then $|I(2, a, b)| = 4^2 - 3^2 = 7$.
- If (a, b) is one of the eight other edge boxes that are children of corner boxes at level $l - 1$, then $|I(2, a, b)| = 4^2 - 2 \cdot 3 = 10$.
- If (a, b) is one of the $4 \cdot 2^l - 16$ other edge boxes, then $|I(2, a, b)| = 6 \cdot 4 - 2 \cdot 3 = 18$.
- If (a, b) is one of the $4 \cdot 2^l - 16$ interior boxes that are not edge boxes but are children of non-corner edge boxes at level $l - 1$, then $|I(2, a, b)| = 4 \cdot 6 - 3^2 = 15$.
- If (a, b) is one of the $4^l - 8 \cdot 2^l + 16$ remaining interior boxes, then $|I(2, a, b)| = 6^2 - 3^2 = 27$.

Hence:

$$N_T(l) = 27 \cdot 4^l - 84 \cdot 2^l + 60 + N_T(l-1) \tag{4.5}$$

$$N_M(l) = 4^l + N_M(l-1) \tag{4.6}$$

$$N_S(l) = 4^l + N_S(l-1) \tag{4.7}$$

Solving the relations in Equations (4.4)–(4.7), then the number of times each operation is used in $POT2D_RECUR$ comes to:

$$\begin{aligned} N_T(h) &= 36 \cdot 4^h - 168 \cdot 2^h + 60h + 132 = s^2 - 168s + 60h + 132; \\ N_M(h) &= \frac{16}{3}(4^{h-2} - 1) = (s^2 - 16)/3; \\ N_S(h) &= \frac{16}{3}(4^{h-2} - 1) = (s^2 - 16)/3. \end{aligned}$$

If F_T, F_M, and F_S are the numbers of flops required for each multiplication by a matrix of type T, M, and S, respectively, then we count the total flops in each step of $POT2D_EVAL(p, h, \mathbf{z}, \mathbf{q})$ as:

1. $F_G = \begin{cases} 2pN & \text{if the } G \text{ matrices are real;} \\ 4pN & \text{otherwise.} \end{cases}$

2. $(36s^2 - 168s + 60h + 132)F_T + \frac{s^2-16}{3}(F_M + F_S)$

3. $F_E + \sum_{a=1}^{s} \sum_{b=1}^{s} 2|\mathcal{Z}_{a,b}|| \cup_{(a',b') \in J(h,a,b)} \mathcal{Z}_{a',b'} \cup \mathcal{Z}_{a,b}|$,

   where
   $$F_E = \begin{cases} 2pN & \text{if the } E \text{ matrices are real;} \\ 4pN & \text{otherwise.} \end{cases}$$

If each box contains the same number of particles, N/s^2, then there is a simpler expression for the flop count in step 3. Since there are four corner boxes, each with three neighbors, $4s - 8$ other edge boxes, each with five neighbors, and $s^2 - 4s + 4$ interior boxes, each with eight neighbors, the number of flops in step 3 is

$$\begin{aligned} F_E &+ 2(N/s^2)(4 \cdot 4 + (4s - 8) \cdot 6 + (s^2 - 4s + 4) \cdot 9)(N/s^2) \\ &= F_E + \frac{2N^2}{s^4}(9s^2 - 12s + 4). \end{aligned}$$

4.4 Greengard–Rokhlin method

The method of Greengard and Rokhlin [14] approximates the *complex* function $f(z) = \sum_j q_j \log(z - z_j)$ instead of the real function in Equation (4.2). A far-field expansion

for a box centered at 0 takes the form of a truncated multipole series with a logarithmic term,

$$f(z) \approx a_1 \log(z) + \sum_{k=2}^{p} \frac{a_k}{z^{k-1}},$$

where a_1 is the sum of the strengths of the charges in the box. Local expansions are polynomials,

$$f(z) \approx \sum_{k=1}^{p} b_k z^{k-1}.$$

As described in Greengard and Rokhlin [14], the matrices in $POT2D$ are:

- $M_{SW}^{(l)}$, $M_{NW}^{(l)}$, $M_{NE}^{(l)}$, $M_{SE}^{(l)}$:
  Set $\Delta z = (-1-i)w_{l+1}, (-1+i)w_{l+1}, (1+i)w_{l+1}, (1-i)w_{l+1}$, respectively.

$$\begin{aligned}
M_{\Delta z}^{(l)}(1,1) &= 1; \\
M_{\Delta z}^{(l)}(j,1) &= -\Delta z^{j-1}/(j-1) \qquad (j = 2,\ldots,p) \\
M_{\Delta z}^{(l)}(j,k) &= \Delta z^{j-k}\binom{j-2}{k-2} \qquad (j = 2,\ldots,p;\ k = 2,\ldots,j) \\
M_{\Delta z}^{(l)}(j,1) &= 0 \qquad (j = 2,\ldots,p;\ k = j+1,\ldots,p)
\end{aligned}$$

  These matrices are lower triangular.

- $S_{SW}^{(l)}$, $S_{NW}^{(l)}$, $S_{NE}^{(l)}$, $S_{SE}^{(l)}$:
  Set $\Delta z = (+1+i)w_l, (+1-i)w_l, (-1-i)w_l, (-1+i)w_l$ respectively.

$$\begin{aligned}
S_{\Delta z}^{(l)}(j,k) &= (-\Delta z)^{k-j}\binom{k-1}{j-1} \qquad (j = 1,\ldots,p;\ k = 1,\ldots,j) \\
S_{\Delta z}^{(l)}(j,k) &= 0 \qquad (j = 1,\ldots,p;\ k = j+1,\ldots,p)
\end{aligned}$$

  These matrices are upper triangular.

- $T^{(l)}_{\Delta a,\Delta b}$: Set $\Delta z = (\Delta a + i\Delta b)2w_l$.

$$\begin{aligned}
T^{(l)}_{\Delta a,\Delta b}(1,1) &= \log(-\Delta z) \\
T^{(l)}_{\Delta a,\Delta b}(j,1) &= \frac{-1}{(j-1)\Delta z^{j-1}} \qquad (j = 2,\ldots,p) \\
T^{(l)}_{\Delta a,\Delta b}(j,k) &= \frac{(-1)^{k-1}\binom{j+k-3}{k-2}}{\Delta z^{j+k+2}} \qquad (j = 1,\ldots,p;\ k = 2,\ldots,p).
\end{aligned}$$

- $G_{a,b}$: For $k \in \mathcal{Z}_{a,b}$,

$$\begin{aligned}
G_{a,b}(1,k) &= 1 \\
G_{a,b}(j,k) &= -\frac{(z_k - c_{a,b})^{j-1}}{j-1} \qquad (j = 2,\ldots,p)
\end{aligned}$$

- $E_{a,b}$: For $j \in \mathcal{Z}_{a,b}$,

$$E_{a,b}(j,k) = (z_j - c_{a,b})^{k-1} \qquad (k = 1,\ldots,p)$$

Multiplying matrices by the direct method, the flop counts are:

- $F_G = 4pN$
- $F_T = 8p^2$
- $F_M = 4p^2$ (triangular matrix)
- $F_S = 4p^2$ (triangular matrix)
- $F_E = 4pN$.

Greengard and Rokhlin [15] use convolution and fast Fourier transforms to improve upon these flop counts, though at the cost of reduced accuracy. In our tests, we use only the direct method of matrix multiplication.

4.5 Anderson's method

Anderson [3] proposed a method based on interpolation with Poisson's integral formula. This formula applies because the real potential f from Equation (4.2) is a harmonic function outside the region containing the charges. If the charges are contained inside a disk of radius a, then the value of the potential f at any point outside the disk is determined by the values of f on its circumference:

$$f(re^{i\theta}) = \kappa \log r + \frac{1}{2\pi}\int_0^{2\pi} (f(ae^{is}) - \kappa \log a) \cdot (\frac{1-(a/r)^2}{1-2(a/r)\cos(\theta - s) + (a/r)^2})ds, \tag{4.8}$$

where κ is the sum of the strengths of the charges inside the disk $\{z : |z| < a\}$. Inside the disk, the potential due to charges outside the disk is

$$f(re^{i\theta}) = \frac{1}{2\pi}\int_0^{2\pi} f(ae^{is})(\frac{1-(r/a)^2}{1-2(r/a)\cos(\theta - s) + (r/a)^2})ds. \tag{4.9}$$

Anderson's method employs discretized versions of these integrals, with $p-1$ interpolation points chosen on each circle.

Approximating the integrals in Equations (4.8)-(4.9) by the trapezoidal rule with $p-1$ points on a circle of radius a, and taking only the first $p/2-1$ Fourier modes, as discussed in Anderson [3], we have the approximations

$$f(re^{i\theta}) \approx \kappa \log r + \frac{1}{p-1}\sum_{k=1}^{p-1}(f(ae^{i\beta_k}) - \kappa \log a)\chi(a/r, \theta - \beta_k) \tag{4.10}$$

and

$$f(re^{i\theta}) \approx \frac{1}{p-1}\sum_{k=1}^{p-1} f(ae^{i\beta_k})\chi(r/a, \theta - \beta_k), \tag{4.11}$$

using the angles

$$\beta_k = \frac{2\pi}{p-1}(k-1) \qquad (k = 1, \ldots, p-1)$$

and the kernel

$$\chi(r,\alpha) = \sum_{j=1-p/2}^{p/2-1} r^{|j|} e^{ij\alpha} = 1 + 2\sum_{j=1}^{p/2-1} r^j \cos(j\alpha). \tag{4.12}$$

The far-field expansion for a box centered at 0 has p coefficients and takes the form

$$f(z) \approx a_1 \log|z| + \frac{1}{p-1}\sum_{k=2}^{p} a_k \cdot \chi(\frac{2w_l R_O}{|z|}, \arg z - \beta_{k-1}),$$

where $2w_l R_O$ is the radius of the circle containing the interpolation points. Since the far-field expansion is evaluated only at points with distance at least $3w_l$ from the center of the box, we normally pick $R_O \gtrsim 3/2$.

The local expansion for the same box has $p-1$ coefficients, $b_2, \ldots, b_p$:

$$f(z) \approx \frac{1}{p-1}\sum_{k=2}^{p} b_k \cdot \chi(\frac{|z|}{2w_l R_I}, \arg z - \beta_{k-1}),$$

where $2w_l R_I$ is the radius of the circle containing the interpolation points. Since the local expansion is evaluated only at points with distance at most $w_l\sqrt{2}$ from the center of the box, we normally pick $R_I \lesssim \sqrt{2}/2$.

Anderson recommends fixing $R_O = \sqrt{2}$ and $R_I = \sqrt{2}/2$. With these settings, far-field expansions are evaluated only outside the circle containing the interpolation points, and local expansions are evaluated only inside the circle containing the interpolation points.

The matrices in *POT2D* are as follows.

- $M_{SW}^{(l)}$, $M_{NW}^{(l)}$, $M_{NE}^{(l)}$, $M_{SE}^{(l)}$:
  Set $\Delta z = w_{l+1}(-1-i), w_{l+1}(-1+i), w_{l+1}(1+i), w_{l+1}(1-i)$ respectively, and define r_k and θ_k for $k = 1, \ldots, p-1$ by

$$r_k e^{i\theta_k} = (2w_l R_O)e^{i\beta_k} - \Delta z.$$

Then

$$
\begin{aligned}
M^{(l)}_{\Delta z}(1,1) &= 1; \\
M^{(l)}_{\Delta z}(1,k) &= 0 \qquad (k=2,\ldots,p) \\
M^{(l)}_{\Delta z}(j,1) &= \log(\frac{r_{j-1}}{2w_l R_O}) \qquad (j=2,\ldots,p) \\
M^{(l)}_{\Delta z}(j,k) &= \frac{1}{p-1}\chi(\frac{2w_{l+1}R_O}{r_{j-1}}, \theta_{j-1}-\beta_{k-1}) \qquad (j=2,\ldots,p;\ k=2,\ldots,p).
\end{aligned}
$$

Notice that these matrices are independent of l.

- $S^{(l)}_{SW}$, $S^{(l)}_{NW}$, $S^{(l)}_{NE}$, $S^{(l)}_{SE}$:
  Set $\Delta z = (+1+i)w_l, (+1-i)w_l, (-1-i)w_l, (-1+i)w_l$ respectively, and define r_k and θ_k for $k=1,\ldots,p-1$ by

$$r_k e^{i\theta_k} = (2w_l R_I)e^{i\beta_k} - \Delta z.$$

Then

$$
S^{(l)}_{\Delta z}(j,k) = \frac{1}{p-1}\chi(\frac{r_{j-1}}{2w_{l-1}R_I}, \theta_{j-1}-\beta_{k-1}) \qquad (j=2,\ldots,p;\ k=2,\ldots,p).
$$

These matrices are independent of l.

- $T^{(l)}_{\Delta a,\Delta b}$: Set $\Delta z = (\Delta a + i\Delta b)2w_l$, and define r_k and θ_k for $k=1,\ldots,p-1$ by

$$r_k e^{i\theta_k} = (2w_l R_I)e^{i\beta_k} - \Delta z.$$

$$
\begin{aligned}
T^{(l)}_{\Delta a,\Delta b}(j,1) &= \log(r_{j-1}) \qquad (j=2,\ldots,p) \\
T^{(l)}_{\Delta a,\Delta b}(j,k) &= \frac{1}{p-1}\chi(\frac{2w_l R_O}{r_{j-1}}, \theta_{j-1}-\beta_{k-1}) \qquad (j=2,\ldots,p;\ k=2,\ldots,p) \\
T^{(l)}_{\Delta a,\Delta b}(1,k) &= 0 \qquad (k=1,\ldots,p).
\end{aligned}
$$

Except for the first column, these matrices are independent of l.

- $G_{a,b}$: For $k \in \mathcal{Z}_{a,b}$,

$$\begin{aligned} G_{a,b}(1,k) &= 1 \\ G_{a,b}(j,k) &= \log|\frac{z_k - c_{a,b}}{2w_h R_O} - e^{i\beta_{j-1}}| \qquad (j = 2, \ldots, p) \end{aligned}$$

- $E_{a,b}$: For $j \in \mathcal{Z}_{a,b}$, set r_j and θ_j by

$$r_j e^{i\theta_j} = z_j - c_{a,b}.$$

Then

$$\begin{aligned} E_{a,b}(j,k) &= \frac{1}{p-1}\chi(\frac{r_j}{2w_h R_I}, \theta_j - \beta_{k-1}) \qquad (k = 2, \ldots, p) \\ E_{a,b}(j,1) &= 0. \end{aligned}$$

The flop counts for multiplying by the matrices as follows.

- $F_G = 2pN$
- $F_T = 2p(p-1)$
- $F_M = 2p(p-1)$
- $F_S = 2(p-1)^2$
- $F_E = 2(p-1)N$.

These flop counts may possibly be reduced by using fast Fourier or cosine transforms. In our tests, we use the direct method of matrix multiplication.

4.6 Polynomial interpolation method

Our third algorithm is based on interpolation of the complex potential of Equation (4.2) by polynomials.

We make use of the $p-1$ interpolating polynomials $u_1, u_2, \ldots, u_{p-1}$ of degree $p-2$:

$$u_k(z) = \prod_{\substack{j=1 \\ j \neq k}}^{p} \frac{z - t_j}{t_k - t_j},$$

where

$$t_j = e^{2\pi i(j-1)/(p-1)}$$

are $p-1$ interpolation points on the unit circle.

The domain containing the points at which the far-field or local potential is to be evaluated is mapped to a disk of fixed size centered at the origin.

- $T_O^{(l)}(z)$ maps z with $|z| \geq 3w_l$ to the disk of radius $2R_O/3$, by

$$T_O^{(l)}(z) = \frac{2w_l R_O}{z}.$$

  The inverse function to $T_O^{(l)}$ is $Z_O^{(l)}(t) = 2w_l R_O/t$.

- $T_I^{(l)}(z)$ maps z with $|z| \leq w_l\sqrt{2}$ to the disk of radius $1/(R_I\sqrt{2})$, by

$$T_I^{(l)}(z) = \frac{z}{2w_l R_I}.$$

  The inverse function to $T_I^{(l)}$ is $Z_I^{(l)}(t) = 2w_l R_I t$.

The functions T_O, Z_O, T_I and Z_I are analogous to X_O, Y_O, X_I and Y_I (respectively) from Chapter 2.

We approximate the complex function $f(z) = \sum_j q_j \log(z - z_j)$. The coefficients that are stored are the values of f, or an approximation of f, at the interpolation points.

Far-field expansions centered at 0 take the form of polynomials in $1/z$, with a logarithmic term:

$$f(z) \approx a_1 \log(z) + \frac{1}{z} \sum_{k=2}^{p} a_k u_{k-1}(T_O^{(l)}(z)).$$

These functions are interpolated at the points $z = Z_O^{(l)}(t_{k-1})$, so

$$a_k = Z_O^{(l)}(t_{k-1}) \cdot [f(Z_O^{(l)}(t_{k-1})) - a_1 \log(Z_O^{(l)}(t_{k-1}))].$$

Local expansions centered at 0 are polynomials of degree $p-2$:

$$f(z) \approx \sum_{k=2}^{p} b_k u_k(T_I^{(l)}(z)),$$

containing $p-1$ coefficients $b_2, \ldots, b_p$. These functions are interpolated at the points $z = Z_I^{(l)}(t_{k-1})$, so

$$b_k = f(Z_I^{(l)}(t_{k-1})).$$

Figure 4-2 illustrates the locations of the interpolation points for far-field and local expansions for $p = 10$.

The matrices used in *POT2D* are:

- $M_{SW}^{(l)}$, $M_{NW}^{(l)}$, $M_{NE}^{(l)}$, $M_{SE}^{(l)}$:
  Set $\Delta z = (-1-i)w_{l+1}, (-1+i)w_{l+1}, (1+i)w_{l+1}, (1-i)w_{l+1}$ respectively.

$$\begin{aligned}
M_{\Delta z}^{(l)}(1,1) &= 1; \\
M_{\Delta z}^{(l)}(1,k) &= 0 \qquad (k = 2, \ldots, p) \\
M_{\Delta z}^{(l)}(j,1) &= Z_O^{(l)}(t_{j-1}) \cdot (\log(Z_O^{(l)}(t_{j-1}) - \Delta z) - \log(Z_O^{(l)}(t_{j-1}))) \\
&= \frac{2w_l R_O}{t_{j-1}} \cdot \log(1 - \frac{t_{j-1}\Delta z}{2w_l R_O}) \qquad (j = 2, \ldots, p) \\
M_{\Delta z}^{(l)}(j,k) &= \frac{Z_O^{(l)}(t_{j-1})}{Z_O^{(l)}(t_{j-1}) - \Delta z} \cdot u_{k-1}(T_O^{(l+1)}(Z_O^{(l)}(t_{j-1}) - \Delta z)) \\
&= \frac{u_{k-1}(\frac{t_{j-1}/2}{1 - t_{j-1}\Delta z/(2w_l R_O)})}{1 - t_{j-1}\Delta z/(2w_l R_O)}) \qquad (j = 2, \ldots, p;\ k = 2, \ldots, p).
\end{aligned}$$

  Note that these matrices are independent of l.

- $S_{SW}^{(l)}$, $S_{NW}^{(l)}$, $S_{NE}^{(l)}$, $S_{SE}^{(l)}$:

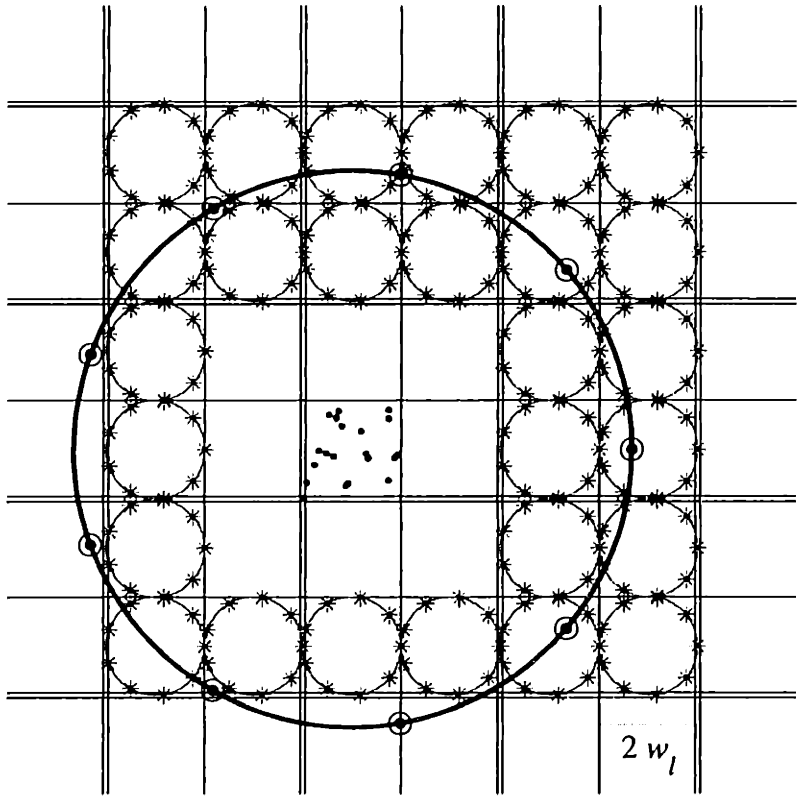


Figure 4-2: An illustration of the polynomial interpolation method, with $p = 10$. Double lines indicate boundaries of boxes at the next higher level. The coefficients of the far-field expansion for the charges in the center box, at level l, are the potentials evaluated at the $p-1$ circled points on the large circle of radius $2w_l R_O$, where here $R_O = 2\sqrt{2}$. The box containing the charges has 27 other boxes in its interaction list. The local expansions at these boxes are found by interpolating at the $p-1$ asterisked points on the smaller circles of radius $2w_l R_I$, where here $R_I = 1/2$.

Set $\Delta z = (+1+i)w_l, (+1-i)w_l, (-1-i)w_l, (-1+i)w_l$ respectively.

$$\begin{aligned} S^{(l)}_{\Delta z}(j,k) &= u_{k-1}(T_I^{(l-1)}(Z_I^{(l)}(t_{j-1}) - \Delta z)) \\ &= u_{k-1}(\frac{t_{j-1}}{2} - \frac{\Delta z}{2w_{l-1}R_I}) \qquad (j = 2,\ldots,p;\ k = 2,\ldots,p) \end{aligned}$$

These matrices are also independent of l.

- $T^{(l)}_{\Delta a,\Delta b}$: Set $\Delta z = (\Delta a + i\Delta b)2w_l$.

$$T^{(l)}_{\Delta a,\Delta b}(j,1) = \log(Z_I^{(l)}(t_{j-1}) - \Delta z) \qquad (j = 2,\ldots,p)$$

$$
\begin{aligned}
T^{(l)}_{\Delta a,\Delta b}(j,k) &= \frac{u_{k-1}(T^{(l)}_O(Z^{(l)}_I(t_{j-1}) - \Delta z))}{Z^{(l)}_I(t_{j-1}) - \Delta z} \\
&= \frac{u_{k-1}(\frac{R_O}{R_I t_{j-1} - \Delta z/(2w_l)})}{2w_l R_I t_{j-1} - \Delta z} \qquad (j = 2,\ldots,p;\ k = 2,\ldots,p)
\end{aligned}
$$

Except for the first column, matrices with the same Δa and Δb, but different l, are constant multiples of each other.

- $G_{a,b}$: For $k \in \mathcal{Z}_{a,b}$,

$$
\begin{aligned}
G_{a,b}(1,k) &= 1 \\
G_{a,b}(j,k) &= Z^{(h)}_O(t_{j-1})(\log(Z^{(h)}_O(t_{j-1}) - (z_k - c_{a,b})) - \log(Z^{(h)}_O(t_{j-1}))) \\
&= \frac{2w_h R_O}{t_{j-1}} \log(1 - \frac{t_{j-1}}{2w_h R_O}(z_k - c_{a,b})) \qquad (j = 2,\ldots,p)
\end{aligned}
$$

- $E_{a,b}$: For $j \in \mathcal{Z}_{a,b}$,

$$
\begin{aligned}
E_{a,b}(j,k) &= u_{k-1}(T^{(h)}_I(z_j - c_{a,b})) \\
&= u_{k-1}((z_j - c_{a,b})/2w_h R_I) \qquad (k = 2,\ldots,p)
\end{aligned}
$$

The flop counts are:

- $F_G = 4pN$
- $F_T = 8p(p-1)$
- $F_M = 8p(p-1)$
- $F_S = 6(p-1)^2$
- $F_E = 4(p-1)N$.

In our tests, we use two settings for R_O and R_I.

1. $R_O = \sqrt{2}, R_I = 1/\sqrt{2}$.

2. $R_O = 2\sqrt{2}, R_I = 1/2$.

Reports of tests are given in Section 4.8. We find that in case 1, the method is numerically stable, but it is not as accurate as Anderson's method with the same settings and the same complexity. This difference may be explained partially by the fact that since Anderson's method works in real arithmetic, it uses twice as many interpolation points for the same complexity. As we shall see in Section 4.7, aliasing increases the error above that of the Greengard–Rokhlin method.

In case 2, the effects of aliasing are reduced, and polynomial interpolation is as accurate as Greengard–Rokhlin or Anderson, up to a certain accuracy, but it is numerically unstable if more interpolation points are taken. This instability occurs because, for local expansions, the interpolating polynomial must be evaluated at some points outside the disk bounded by the circle containing the interpolation points. The effects of roundoff error are magnified when interpolating from a polynomial with a large number of points on a circle, at points outside the circle. The same phenomenon occurs with far-field expansions, with the "inside" and "outside" of the circle interchanged in this description.

4.7 Coefficients from Taylor series

The algorithms described in Sections 4.4–4.6 are based on approximating Taylor series of the analytic functions g listed in Table 4.1, where $f(z) = \sum_j q_j \log(z - z_j)$. For far-field expansions, we maintain the coefficient κ, the sum of the strengths of charges inside the box.

Table 4.1: Analytic functions used for each approximation method.

Method	far-field expansions	local expansions
Greengard–Rokhlin	$g(z) = \frac{1}{z}[f(\frac{1}{z}) - \kappa \log(\frac{1}{z})]$	$g(z) = f(z)$
Polynomial interpolation	$g(z) = \frac{1}{z}[f(\frac{1}{z}) - \kappa \log(\frac{1}{z})]$	$g(z) = f(z)$
Anderson	$g(z) = f(\frac{1}{z}) - \kappa \log(\frac{1}{z})$	$g(z) = f(z)$

Theorem 4.7.1 *The analytic function*

$$g(z) = \sum_{k=0}^{\infty} a_k z^k$$

from Table 4.1 is approximated with m terms by

$$h(z) = \sum_{k=0}^{m-1} b_k z^k$$

in the Greengard–Rokhlin and polynomial interpolation methods, or by $\Re\mathrm{e}\ h(z)$ in Anderson's method. The coefficients b_k are given by:

- *Greengard–Rokhlin: $b_k = a_k$.*
- *Polynomial interpolation: $b_k = \sum_{l=0}^{\infty} a_{k+ml} R^{ml}$,*
  where R is the radius of the circle containing interpolation points for g.
- *Anderson: $b_0 = \sum_{l=0}^{\infty} a_{(2m-1)l} R^{(2m-1)l}$;*
  for $k = 1, \ldots, m$: $b_k = \sum_{l=0}^{\infty} (a_{k+(2m-1)l} + \bar{a}_{(2m-1)(l+1)-k} R^{2m-1-2k}) R^{(2m-1)l}$,
  where R is the radius of the circle containing interpolation points for g.

Proof.

- Greengard–Rokhlin: Immediate from Section 4.4. Here $m = p - 1$ for far-field expansions, $m = p$ for local expansions.
- Polynomial interpolation: When $\omega_m = e^{2\pi i/m}$,

$$g(R\omega_m^j) = \sum_{k=0}^{\infty} a_k R^k \omega_m^{jk} = \sum_{k=0}^{m-1} (\sum_{l=0}^{\infty} a_{k+ml} R^{k+ml}) \omega_m^{jk} = h(R\omega_m^j).$$

  Since h is a complex polynomial of degree $m - 1$ that agrees with the interpolating polynomial at the m interpolation points, h is equal to the interpolating polynomial.

- Anderson: From Equations (4.10)–(4.12), Anderson's approximations for $p = 2m$ give an expression of the form

$$h_1(re^{i\theta}) = \sum_{k=1-m}^{m-1} d_k r^{|k|} e^{ik\theta} = d_0 + \sum_{k=1}^{m-1} d_k(z^k + \bar{z}^k) = \Re\text{e}\ \{d_0 + 2\sum_{k=1}^{m-1} d_k z^k\}$$

with real coefficients $d_{1-m}, \ldots, d_{m-1}$. The interpolation points are at $R\omega_{2m-1}^j$ for $j = 0, 2, \ldots, 2m-2$, where $\omega_{2m-1} = e^{2\pi i/(2m-1)}$. Therefore, $h_1(R\omega_{2m-1}^j) = \Re\text{e}\ h_2(R\omega_{2m-1}^j)$, where h_2 is a complex polynomial interpolating g at the $2m-1$ points. According to the previous part of this theorem,

$$h_2(z) = \sum_{k=0}^{2m-2} c_k z^k,$$

where $c_k = \sum_{l=0}^{\infty} a_{k+(2m-1)l} R^{(2m-1)l}$. Observe that

$$\begin{aligned}
\Re\text{e}\ h(R\omega_{2m-1}^j) &= \Re\text{e}\ \{c_0 + \sum_{k=1}^{m-1}(c_k + \bar{c}_{2m-1-k}R^{2m-1-2k})R^k\omega_{2m-1}^{jk}\} \\
&= \frac{c_0 + \bar{c}_0}{2} + \frac{1}{2}\sum_{k=1}^{m-1}(c_k R^k \omega_{2m-1}^{jk} + \bar{c}_k R^k \omega_{2m-1}^{-jk}) + \\
&\quad \frac{1}{2}\sum_{k=1}^{m-1}(\bar{c}_{2m-1-k}R^{2m-1-k}\omega_{2m-1}^{jk} + c_{2m-1-k}R^{2m-1-k}\omega_{2m-1}^{-jk}) \\
&= \Re\text{e}\ c_0 + \Re\text{e}\ \sum_{k=1}^{2m-2} c_k R^k \omega_{2m-1}^{jk} \\
&= \Re\text{e}\ h_2(R\omega_{2m-1}^j)
\end{aligned}$$

Since the real parts are the same at $2m-1$ points, h differs from h_2 by only a constant.

□

In Theorem 4.7.1, note that mathematically, the Greengard–Rokhlin method is the same as polynomial interpolation with $R = 0$. The effects of aliasing are increased with increasing R, although interpolation with small R leads to numerical instabilities, as discussed in Section 4.6.

The following theorem expresses the approximating function h in terms of the convolution of a kernel with g.

Theorem 4.7.2 *For analytic functions g from Table 4.1, in the Greengard-Rokhlin and polynomial interpolation methods the approximating functions are given by:*

$$h(re^{i\theta}) = \frac{1}{2\pi}\int_0^{2\pi} K(r,\phi,\theta)g(re^{i\phi})d\phi.$$

In Anderson's method, the approximating functions are:

$$h(re^{i\theta}) = \frac{1}{2\pi}\int_0^{2\pi} K_1(r,\phi,\theta)g(re^{i\phi})d\phi + \frac{1}{2\pi}\int_0^{2\pi} K_2(r,\phi,\theta)\bar{g}(re^{i\phi})d\phi.$$

The kernels are as follows.

- *Greengard-Rokhlin:*

$$\begin{aligned} K(r,\phi,\theta) &= \sum_{k=0}^{m-1} e^{ik(\theta-\phi)} \\ &= e^{i(\theta-\phi)(m-1)/2}\frac{\sin(m(\theta-\phi)/2)}{\sin((\theta-\phi)/2)}. \end{aligned}$$

  This is a Dirichlet kernel.

- *Polynomial interpolation:*

$$\begin{aligned} K(r,\phi,\theta) &= [\sum_{k=0}^{m-1} e^{ik(\theta-\phi)}]\cdot[\sum_{l=0}^{\infty}(\frac{R}{re^{i\phi}})^{ml}] \\ &= e^{i(\theta-\phi)(m-1)/2}\frac{\sin(m(\theta-\phi)/2)}{\sin((\theta-\phi)/2)}\cdot\frac{1}{1-(R/(re^{i\phi}))^m}. \end{aligned}$$

- *Anderson:*

$$\begin{aligned} K_1(r,\phi,\theta) &= [\sum_{k=0}^{m-1} e^{ik(\theta-\phi)}]\cdot[\sum_{l=0}^{\infty}(\frac{R}{re^{i\phi}})^{(2m-1)l}] \\ &= e^{i(\theta-\phi)(m-1)/2}\frac{\sin(m(\theta-\phi)/2)}{\sin((\theta-\phi)/2)}\cdot\frac{1}{1-(R/(re^{i\phi}))^{2m-1}}; \end{aligned}$$

$$K_2(r,\phi,\theta) = [\sum_{k=m}^{2m-2}((\frac{R}{r})^2 e^{i(\phi-\theta)})^k] \cdot [\sum_{l=0}^{\infty}(\frac{Re^{i\phi}}{r})^{(2m-1)l}] \cdot (\frac{re^{i\theta}}{R})^{2m-1}.$$

Proof. We use the expansions from Theorem 4.7.1. If

$$g(re^{i\theta}) = \sum_{j=0}^{\infty} a_j r^j e^{ij\theta}$$

then

$$\frac{1}{2\pi}\int_0^{2\pi} K(r,\phi,\theta)g(re^{i\phi})d\phi = \sum_{j=0}^{\infty} a_j r^j \frac{1}{2\pi}\int_0^{2\pi} K(r,\phi,\theta)e^{ij\phi}d\phi.$$

- Greengard–Rokhlin:

$$\begin{aligned}\frac{1}{2\pi}\int_0^{2\pi} K(r,\phi,\theta)e^{ij\phi}d\phi &= \sum_{k=0}^{m-1} e^{ik\theta}\frac{1}{2\pi}\int_0^{2\pi} e^{-ik\phi}e^{ij\phi}d\phi \\ &= \begin{cases} e^{ij\theta} & \text{if } j = 0,1,\ldots,m-1; \\ 0 & \text{otherwise.}\end{cases}\end{aligned}$$

Hence

$$\sum_{j=0}^{\infty} a_j r^j \frac{1}{2\pi}\int_0^{2\pi} K(r,\phi,\theta)e^{ij\phi}d\phi = \sum_{j=0}^{m-1} a_j r^j e^{ij\theta} = h(re^{i\theta}).$$

- Polynomial interpolation:

$$\begin{aligned}\frac{1}{2\pi}\int_0^{2\pi} K(r,\phi,\theta)e^{ij\phi}d\phi &= \sum_{k=0}^{m-1}\sum_{l=0}^{\infty} e^{ik\theta}(\frac{R}{r})^{ml}\frac{1}{2\pi}\int_0^{2\pi} e^{-ik\phi}e^{-iml\phi}e^{ij\phi}d\phi \\ &= e^{ik\theta}(\frac{R}{r})^{ml},\end{aligned}$$

where $j = k + ml$ and $0 \le k \le m-1$. Hence

$$\begin{aligned}\sum_{j=0}^{\infty} a_j r^j \frac{1}{2\pi}\int_0^{2\pi} K(r,\phi,\theta)e^{ij\phi}d\phi &= \sum_{k=0}^{m-1}\sum_{l=0}^{\infty} a_{k+ml} r^{k+ml} e^{ik\theta}(\frac{R}{r})^{ml} \\ &= \sum_{k=0}^{m-1}(\sum_{l=0}^{\infty} a_{k+ml}R^{ml})r^k e^{ik\theta} \\ &= h(re^{i\theta}).\end{aligned}$$

- Anderson:

$$\begin{aligned}\frac{1}{2\pi}\int_0^{2\pi} K_1(r,\phi,\theta)e^{ij\phi}d\phi &= \sum_{k=0}^{m-1}\sum_{l=0}^{\infty} e^{ik\theta}(\frac{R}{r})^{(2m-1)l}\,. \\ &\quad \frac{1}{2\pi}\int_0^{2\pi} e^{-ik\phi}e^{-i(2m-1)l\phi}e^{ij\phi}d\phi \\ &= e^{ik\theta}(\frac{R}{r})^{(2m-1)l},\end{aligned}$$

if $j = k + (2m-1)l$ with $0 \le k \le m-1$ and integer l, and zero otherwise. Hence

$$\begin{aligned}\sum_{j=0}^{\infty} a_j r^j \quad &\cdot \quad \frac{1}{2\pi}\int_0^{2\pi} K_1(r,\phi,\theta)e^{ij\phi}d\phi \\ &= \sum_{k=0}^{m-1}\sum_{l=0}^{\infty} a_{k+(2m-1)l}r^{k+(2m-1)l}e^{ik\theta}(\frac{R}{r})^{(2m-1)l} \\ &= \sum_{k=0}^{m-1}(\sum_{l=0}^{\infty} a_{k+(2m-1)l}R^{(2m-1)l})r^k e^{ik\theta}. \qquad (4.13)\end{aligned}$$

Also:

$$\begin{aligned}\frac{1}{2\pi}\int_0^{2\pi} K_2(r,\phi,\theta)e^{-ij\phi}d\phi &= \sum_{k=m}^{2m-2}\sum_{l=0}^{\infty} e^{i(2m-1-k)\theta}(\frac{R}{r})^{2k+(2m-1)(l-1)}\,. \\ &\quad \frac{1}{2\pi}\int_0^{2\pi} e^{ik\phi}e^{i(2m-1)l\phi}e^{-ij\phi}d\phi \\ &= e^{i(2m-1-k)\theta}(\frac{R}{r})^{2k+(2m-1)(l-1)},\end{aligned}$$

if $j = k + (2m-1)l$ with $m \le k \le 2m-2$ and integer l, and zero otherwise. Hence

$$\begin{aligned}\sum_{j=0}^{\infty} \bar{a}_j r^j \quad &\cdot \quad \frac{1}{2\pi}\int_0^{2\pi} K_2(r,\phi,\theta)e^{-ij\phi}d\phi \\ &= \sum_{k=m}^{2m-2}\sum_{l=0}^{\infty} \bar{a}_{k+(2m-1)l}r^{k+(2m-1)l}e^{i(2m-1-k)\theta}(\frac{R}{r})^{2k+(2m-1)(l-1)}\end{aligned}$$

$$
\begin{aligned}
&= \sum_{k=m}^{2m-2}\left(\sum_{l=0}^{\infty}\bar{a}_{k+(2m-1)l}R^{(2m-1)(l-1)+2k}\right)r^{2m-1-k}e^{i(2m-1-k)\theta} \\
&= \sum_{k=1}^{m-1}\left(\sum_{l=0}^{\infty}\bar{a}_{(2m-1)(l+1)-k}R^{(2m-1)l+(2m-1-2k)}\right)r^{k}e^{ik\theta}. \qquad (4.14)
\end{aligned}
$$

Combining Equations (4.13)–(4.14), then

$$
\begin{aligned}
&\frac{1}{2\pi}\int_0^{2\pi} K_1(r,\phi,\theta)g(re^{i\phi})d\phi + \frac{1}{2\pi}\int_0^{2\pi} K_2(r,\phi,\theta)\bar{g}(re^{i\phi})d\phi \\
&= \sum_{k=0}^{m-1}\left(\sum_{l=0}^{\infty}\left(a_{k+(2m-1)l}R^{(2m-1)l} + \bar{a}_{(2m-1)(l+1)-k}R^{(2m-1)l+(2m-1-2k)}\right)\right)r^{k}e^{ik\theta} \\
&= h(re^{i\theta}).
\end{aligned}
$$

□

For more on the mathematical theory of complex approximation, the reader is referred to Gaier [12].

4.8 Comparison of algorithms

In this section we report results of tests of the three algorithms. These computations were performed in double-precision arithmetic in MATLAB on a Digital AlphaServer 8400. We report the error, compared with direct evaluation, and the number of floating-point operations for *POT2D_EVAL*.

Two different problem sizes were tried:

- $10,240$ particles, with $h = 4$ hierarchy levels in *POT2D*;
- $40,960$ particles, with $h = 5$ hierarchy levels in *POT2D*.

In all cases, each leaf-level box contains 40 particles. We used three types of distributions of particle positions within leaf-level boxes, as illustrated in Figure 4-3:

- UNIFORM distribution: the 40 particles are distributed uniformly within each box.

- CORNER distribution: there are 10 particles distributed uniformly within each of four sub-boxes in the corners of each leaf-level box. Each sub-box has one quarter the width of the leaf-level boxes.

- CENTER distribution: the 40 particles are distributed uniformly within a sub-box concentric with the leaf-level box, but half as wide.

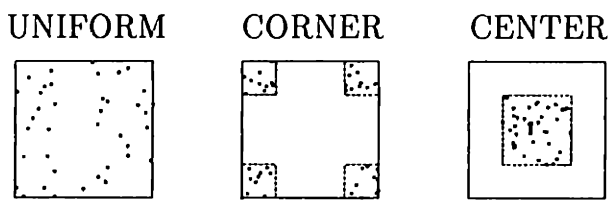


Figure 4-3: The three types of particle distributions. Each leaf-level box contains 40 particles.

These distributions were suggested by Hu and Johnsson [24].

For each of the two problem sizes (10K particles or 40K particles) and for each of the three particle distributions (UNIFORM, CORNER, and CENTER), we report results for one random problem instance. In all cases, the charge strengths are chosen from a normal distribution with mean 0 and variance 1.

Error is reported as the normalized ℓ^2 error,

$$\frac{\|f - \tilde{f}\|_2}{\|f\|_2},$$

in Tables 4.2–4.7. In Figures 4-4–4-7, the error is plotted against the flop count of *POT2D_EVAL* for different choices of lengths of expansions.

We see from Figures 4-4 and 4-6 that the Greengard–Rokhlin and Anderson algorithms have practically indistinguishable performance on the examples that were tested. For all methods, error is greatest for the CORNER distribution and least for the CENTER distribution, with error for the UNIFORM distribution being in between. This is expected, since the far-field and local approximations for all three methods are most accurate when charges are concentrated near the center of each

box. Figures 4-5 and 4-7 compare the errors with each charge distribution with each method. Since the distribution of the particle positions affects primarily the accuracy of calculations on the leaf level, it follows that the more levels used, the less significant the effects of different distributions. The relative effects with four and five levels can be seen by comparing Figures 4-5 and 4-7.

Complex polynomial interpolation does not perform as well as the other two methods when highest accuracy is needed. With $R_O = 2\sqrt{2}$ and $R_I = 1/2$, polynomial interpolation achieves minimum error on the order of 10^{-11} to 10^{-10}, with about 22 interpolation points for the examples tested here. In this range, the algorithm is competitive with Greengard–Rokhlin and Anderson's method. However, with more interpolation points, the polynomial interpolation method loses accuracy due to roundoff error. A choice of $R_O = 2\sqrt{2}$ and $R_I = 1/2$ removes the problem with roundoff error, and allows up to machine precision. However, the method then becomes less accurate than Greengard–Rokhlin or Anderson's algorithm by one or two orders of magnitude.

4.9 Conclusions

We find that by taking simple flop counts for the operations in $POT2D_EVAL$, all three algorithms show comparable performance up to a certain accuracy. Depending on a particular implementation and how it takes advantage of efficiencies in the matrix multiplications, any one of the three algorithms may turn out to be superior to the others. The most work so far has been on the Greengard–Rokhlin method [26], although Anderson's method has attracted recent attention [24], and economizations may also be found for this approach, as well as for complex polynomial interpolation.

Table 4.2: Normalized ℓ^2 error in the Greengard-Rokhlin algorithm with 10K random charges and four hierarchy levels.

p	megaflops	UNIFORM	CORNER	CENTER
7	10.081	2.25E-05	4.31E-05	1.15E-05
8	11.000	7.72E-06	1.53E-05	3.54E-06
9	12.032	2.66E-06	5.05E-06	1.05E-06
10	13.174	8.06E-07	1.77E-06	2.89E-07
11	14.429	3.38E-07	7.20E-07	9.43E-08
12	15.795	1.19E-07	2.75E-07	3.22E-08
13	17.273	4.60E-08	9.96E-08	9.56E-09
14	18.863	1.81E-08	4.05E-08	3.03E-09
15	20.564	7.55E-09	1.61E-08	1.02E-09
16	22.377	2.77E-09	6.35E-09	3.28E-10
17	24.302	1.04E-09	2.49E-09	1.12E-10
18	26.338	4.36E-10	1.03E-09	3.54E-11
19	28.486	1.80E-10	4.61E-10	1.24E-11
20	30.746	7.63E-11	1.91E-10	4.12E-12
21	33.117	2.85E-11	7.06E-11	1.44E-12
22	35.600	1.19E-11	2.97E-11	4.78E-13
23	38.195	4.63E-12	1.29E-11	1.80E-13
24	40.901	2.11E-12	5.28E-12	6.54E-14
25	43.719	8.07E-13	2.18E-12	2.04E-14
26	46.649	3.70E-13	9.69E-13	7.66E-15
27	49.690	1.46E-13	4.04E-13	3.28E-15
28	52.844	6.82E-14	1.68E-13	1.64E-15
29	56.108	2.77E-14	8.04E-14	1.23E-15
30	59.485	1.24E-14	3.43E-14	1.20E-15
31	62.973	5.25E-15	1.48E-14	1.19E-15
32	66.573	2.65E-15	6.18E-15	1.19E-15
33	70.284	1.57E-15	3.27E-15	1.19E-15
34	74.108	1.25E-15	1.77E-15	1.19E-15
35	78.042	1.20E-15	1.33E-15	1.19E-15
36	82.089	1.19E-15	1.24E-15	1.19E-15
37	86.247	1.18E-15	1.23E-15	1.19E-15
38	90.517	1.19E-15	1.22E-15	1.19E-15
39	94.899	1.18E-15	1.22E-15	1.19E-15
40	99.392	1.18E-15	1.22E-15	1.19E-15

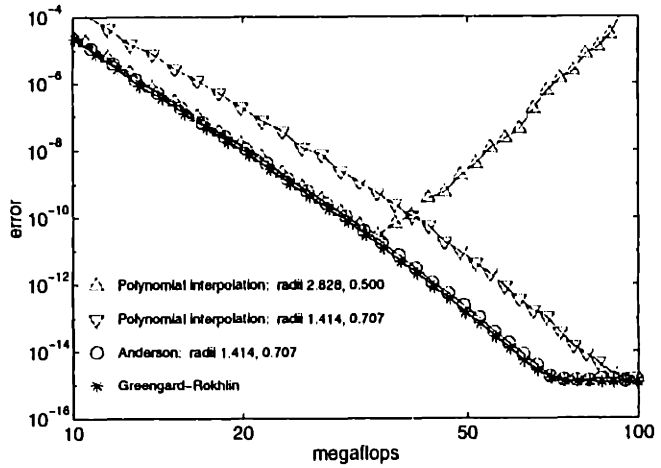
Table 4.3: Normalized ℓ^2 error in Anderson's algorithm with 10K random charges and four hierarchy levels.

p	megaflops	UNIFORM	CORNER	CENTER
14	9.892	3.56E-05	6.61E-05	1.90E-05
16	10.792	1.19E-05	2.17E-05	5.65E-06
18	11.806	3.97E-06	7.63E-06	1.52E-06
20	12.932	1.28E-06	2.79E-06	4.73E-07
22	14.172	5.07E-07	1.07E-06	1.49E-07
24	15.524	1.80E-07	3.69E-07	5.20E-08
26	16.990	6.41E-08	1.49E-07	1.26E-08
28	18.568	2.57E-08	5.73E-08	4.68E-09
30	20.259	1.09E-08	2.33E-08	1.70E-09
32	22.064	4.20E-09	8.65E-09	5.51E-10
34	23.981	1.48E-09	3.78E-09	1.42E-10
36	26.011	6.23E-10	1.48E-09	5.44E-11
38	28.154	2.70E-10	6.75E-10	2.10E-11
40	30.410	1.11E-10	2.59E-10	6.53E-12
42	32.779	3.99E-11	1.06E-10	1.91E-12
44	35.261	1.73E-11	4.41E-11	7.21E-13
46	37.856	7.26E-12	1.98E-11	3.15E-13
48	40.564	2.99E-12	7.85E-12	9.40E-14
50	43.385	1.12E-12	3.25E-12	2.71E-14
52	46.319	5.38E-13	1.36E-12	1.14E-14
54	49.366	2.34E-13	6.14E-13	5.42E-15
56	52.525	1.01E-13	2.59E-13	2.12E-15
58	55.798	3.81E-14	1.15E-13	1.42E-15
60	59.184	1.79E-14	5.06E-14	1.29E-15
62	62.682	8.38E-15	2.24E-14	1.21E-15
64	66.294	3.90E-15	9.71E-15	1.24E-15
66	70.018	1.81E-15	4.22E-15	1.18E-15
68	73.856	1.41E-15	2.40E-15	1.21E-15
70	77.806	1.35E-15	1.77E-15	1.24E-15
72	81.870	1.30E-15	1.54E-15	1.22E-15
74	86.046	1.46E-15	1.76E-15	1.35E-15
76	90.336	1.49E-15	1.78E-15	1.37E-15
78	94.738	1.26E-15	1.45E-15	1.21E-15
80	99.253	1.40E-15	1.58E-15	1.32E-15

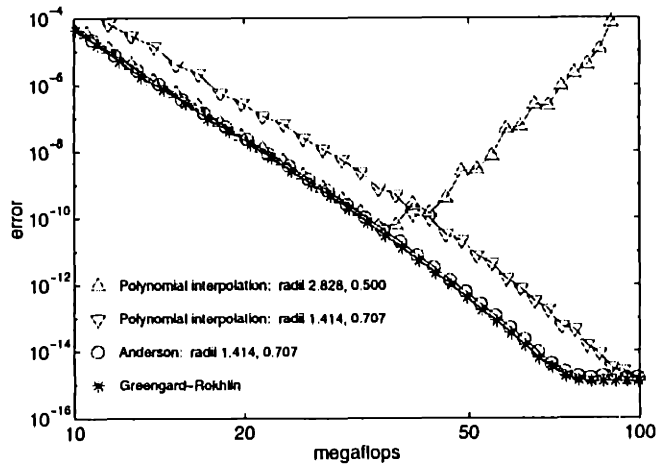
Table 4.4: Normalized ℓ^2 error in polynomial interpolation algorithm with 10K random charges and four hierarchy levels.

		$R_O = \sqrt{2}, R_I = 1/\sqrt{2}$			$R_O = 2\sqrt{2}, R_I = 1/2$		
p	Mflops	UNIF.	CORNER	CENTER	UNIF.	CORNER	CENTER
7	9.672	3.24E-04	4.32E-04	3.00E-04	6.90E-05	1.18E-04	5.01E-05
8	10.544	1.15E-04	1.61E-04	1.01E-04	1.89E-05	3.49E-05	1.16E-05
9	11.529	4.67E-05	6.97E-05	4.09E-05	6.34E-06	1.23E-05	3.15E-06
10	12.627	1.62E-05	3.03E-05	1.39E-05	2.17E-06	4.13E-06	9.54E-07
11	13.838	8.06E-06	1.46E-05	5.31E-06	7.26E-07	1.38E-06	2.80E-07
12	15.162	2.78E-06	4.09E-06	2.11E-06	2.81E-07	5.84E-07	9.27E-08
13	16.598	1.24E-06	2.27E-06	9.79E-07	1.01E-07	2.39E-07	3.47E-08
14	18.148	5.15E-07	5.76E-07	3.01E-07	3.65E-08	7.83E-08	8.13E-09
15	19.811	1.89E-07	2.84E-07	1.11E-07	1.56E-08	2.87E-08	2.73E-09
16	21.587	7.53E-08	1.17E-07	4.78E-08	6.49E-09	1.26E-08	9.01E-10
17	23.475	3.73E-08	6.95E-08	2.16E-08	2.21E-09	5.28E-09	2.79E-10
18	25.477	1.22E-08	2.52E-08	7.17E-09	8.70E-10	1.94E-09	8.70E-11
19	27.591	7.84E-09	1.18E-08	2.66E-09	3.57E-10	7.94E-10	3.08E-11
20	29.819	2.53E-09	5.69E-09	1.27E-09	1.52E-10	3.78E-10	1.03E-11
21	32.159	1.19E-09	2.42E-09	4.60E-10	6.01E-11	1.52E-10	3.85E-12
22	34.613	5.48E-10	7.68E-10	2.08E-10	2.88E-11	5.66E-11	3.05E-12
23	37.179	2.52E-10	5.01E-10	6.39E-11	6.49E-11	6.11E-11	7.03E-12
24	39.859	1.11E-10	1.35E-10	3.97E-11	1.17E-10	2.80E-10	5.26E-11
25	42.651	5.59E-11	9.69E-11	1.58E-11	3.75E-10	1.43E-10	8.59E-12
26	45.556	1.80E-11	3.23E-11	5.59E-12	5.66E-10	3.89E-10	1.71E-11
27	48.574	9.59E-12	2.14E-11	2.43E-12	1.85E-09	2.50E-09	1.03E-10
28	51.706	4.27E-12	7.59E-12	1.33E-12	4.04E-09	2.86E-09	6.08E-11
29	54.950	2.11E-12	4.43E-12	5.67E-13	1.17E-08	7.29E-09	1.52E-10
30	58.307	8.73E-13	1.57E-12	1.70E-13	2.24E-08	4.84E-08	1.54E-09
31	61.777	3.65E-13	7.88E-13	8.61E-14	4.30E-08	5.63E-08	9.68E-10
32	65.360	1.65E-13	3.31E-13	4.80E-14	1.61E-07	2.57E-07	5.08E-09
33	69.056	1.09E-13	1.82E-13	1.80E-14	5.73E-07	2.37E-07	1.86E-09
34	72.865	3.74E-14	5.60E-14	5.31E-15	1.34E-06	9.15E-07	9.78E-09
35	76.787	1.46E-14	4.62E-14	3.19E-15	2.24E-06	2.20E-06	1.74E-08
36	80.822	8.17E-15	1.52E-14	1.88E-15	7.80E-06	4.02E-06	2.55E-08
37	84.970	4.64E-15	8.65E-15	1.51E-15	1.24E-05	1.17E-05	7.05E-08
38	89.230	2.44E-15	3.20E-15	1.19E-15	2.91E-05	7.04E-05	3.21E-07
39	93.604	1.61E-15	2.66E-15	1.36E-15	1.31E-04	1.58E-04	6.37E-07
40	98.091	1.68E-15	1.71E-15	1.25E-15	3.99E-04	2.99E-04	1.12E-06
41	102.691	1.60E-15	1.73E-15	1.31E-15	1.05E-03	6.35E-04	1.96E-06

UNIFORM distribution



CORNER distribution



CENTER distribution

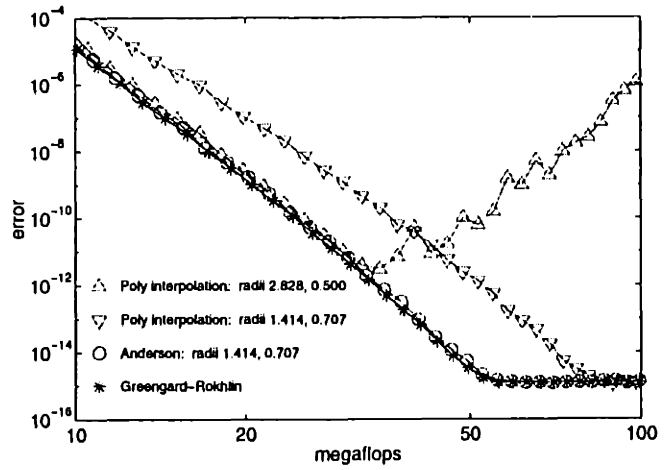


Figure 4-4: Plot of normalized ℓ^2 error against flop count for *POT2D_EVAL* with four hierarchy levels and 10,240 charges. There are 40 random charges in each of the 256 leaf-level boxes. These graphs contain the same information as Tables 4.2, 4.3, and 4.4.

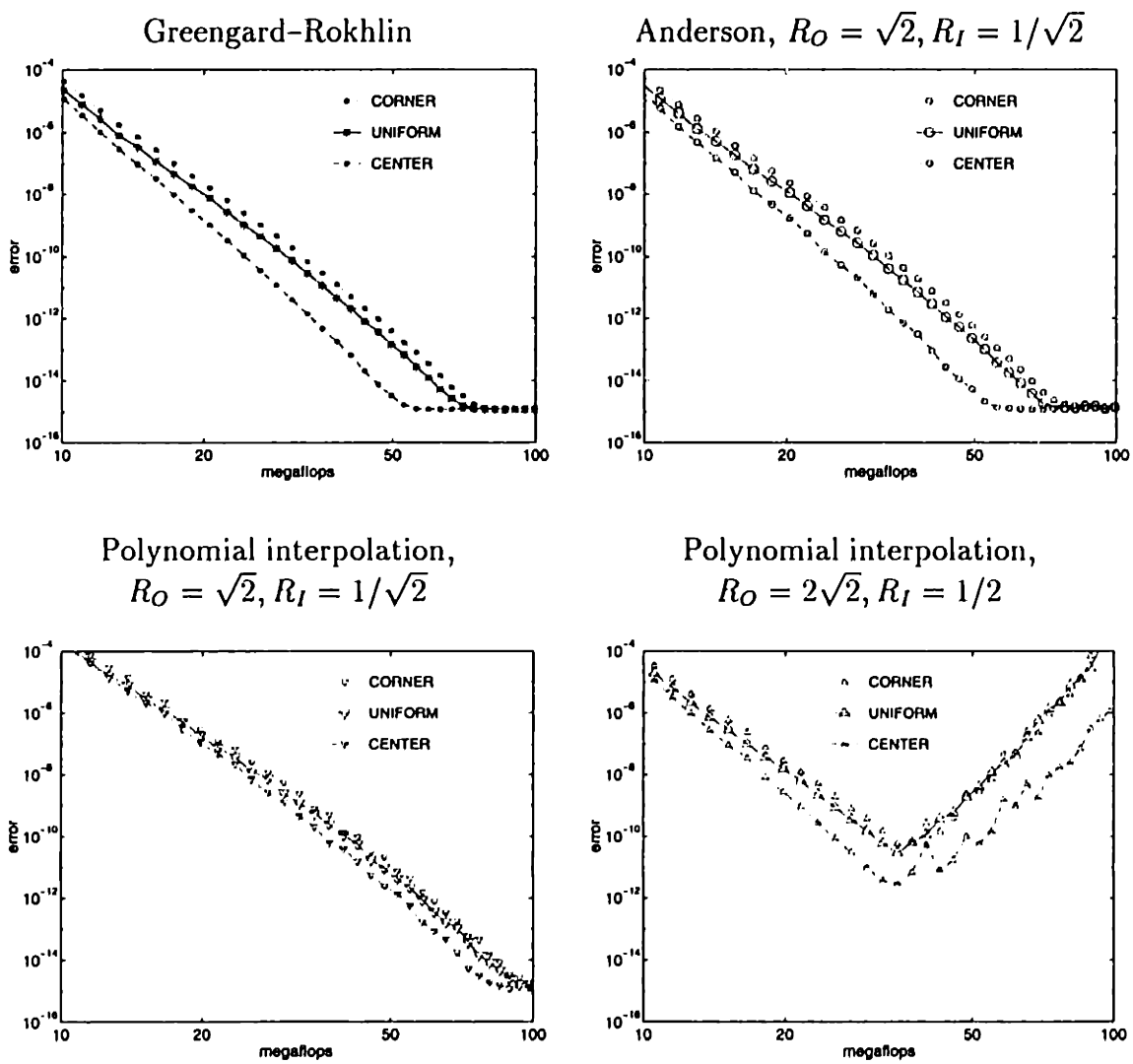


Figure 4-5: Plot of normalized ℓ^2 error against flop count for $POT2D_EVAL$ with four hierarchy levels and 10,240 charges. Each graph is for a separate method. There are 40 random charges in each of the 256 leaf-level boxes.

Table 4.5: Normalized ℓ^2 error in Greengard–Rokhlin algorithm with 40K random charges and five hierarchy levels.

p	megaflops	UNIFORM	CORNER	CENTER
8	47.412	9.53E-06	1.34E-05	6.89E-06
9	52.126	3.19E-06	4.60E-06	2.24E-06
10	57.357	1.01E-06	1.53E-06	6.33E-07
11	63.103	3.91E-07	6.39E-07	2.42E-07
12	69.366	1.56E-07	2.50E-07	9.47E-08
13	76.145	6.02E-08	9.73E-08	3.42E-08
14	83.440	2.05E-08	3.40E-08	9.25E-09
15	91.251	8.41E-09	1.53E-08	4.03E-09
16	99.578	3.12E-09	6.28E-09	1.44E-09
17	108.422	1.26E-09	2.45E-09	5.40E-10
18	117.781	5.36E-10	8.82E-10	1.85E-10
19	127.656	2.26E-10	4.34E-10	7.98E-11
20	138.048	8.45E-11	1.93E-10	2.81E-11
21	148.956	3.51E-11	7.13E-11	1.09E-11
22	160.379	1.63E-11	2.64E-11	3.94E-12
23	172.319	6.60E-12	1.30E-11	1.65E-12
24	184.775	2.52E-12	5.63E-12	6.19E-13
25	197.747	1.07E-12	2.31E-12	2.54E-13
26	211.235	5.17E-13	9.37E-13	1.03E-13
27	225.240	2.24E-13	4.44E-13	4.37E-14
28	239.760	8.55E-14	1.84E-13	1.53E-14
29	254.796	3.60E-14	8.16E-14	6.66E-15
30	270.349	1.78E-14	3.40E-14	3.79E-15
31	286.417	8.18E-15	1.65E-14	2.84E-15
32	303.002	3.86E-15	7.07E-15	2.62E-15
33	320.103	2.91E-15	3.90E-15	2.60E-15
34	337.720	2.70E-15	2.89E-15	2.60E-15
35	355.853	2.62E-15	2.70E-15	2.60E-15
36	374.502	2.60E-15	2.66E-15	2.62E-15
37	393.667	2.61E-15	2.64E-15	2.59E-15
38	413.348	2.59E-15	2.63E-15	2.59E-15

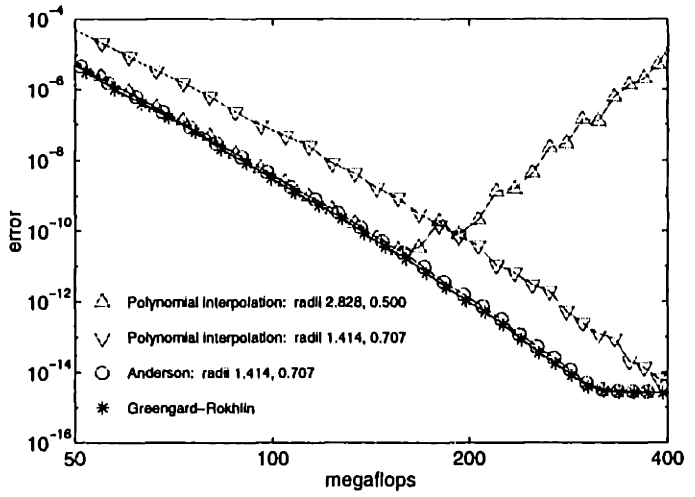
Table 4.6: Normalized ℓ^2 error in Anderson's algorithm with 40K random charges and five hierarchy levels.

p	megaflops	UNIFORM	CORNER	CENTER
16	46.449	1.49E-05	2.05E-05	1.09E-05
18	51.077	4.67E-06	6.75E-06	3.22E-06
20	56.227	1.55E-06	2.38E-06	9.30E-07
22	61.899	6.02E-07	9.57E-07	3.79E-07
24	68.092	2.20E-07	3.61E-07	1.32E-07
26	74.806	8.15E-08	1.39E-07	4.47E-08
28	82.042	2.95E-08	5.05E-08	1.32E-08
30	89.799	1.23E-08	2.22E-08	6.07E-09
32	98.078	4.70E-09	9.30E-09	2.22E-09
34	106.878	1.76E-09	3.74E-09	7.75E-10
36	116.200	8.05E-10	1.34E-09	2.72E-10
38	126.043	3.25E-10	6.34E-10	1.17E-10
40	136.408	1.26E-10	2.71E-10	4.27E-11
42	147.294	5.12E-11	1.10E-10	1.64E-11
44	158.701	2.40E-11	4.10E-11	6.10E-12
46	170.631	9.52E-12	1.91E-11	2.53E-12
48	183.081	3.63E-12	8.01E-12	9.51E-13
50	196.053	1.59E-12	3.47E-12	3.81E-13
52	209.547	7.29E-13	1.41E-12	1.49E-13
54	223.562	3.18E-13	6.46E-13	6.44E-14
56	238.098	1.19E-13	2.68E-13	2.30E-14
58	253.156	5.25E-14	1.17E-13	9.80E-15
60	268.735	2.55E-14	4.96E-14	4.92E-15
62	284.836	1.17E-14	2.36E-14	3.13E-15
64	301.459	4.86E-15	9.83E-15	2.81E-15
66	318.602	3.24E-15	5.05E-15	2.61E-15
68	336.268	2.91E-15	3.55E-15	2.64E-15
70	354.454	2.85E-15	3.20E-15	2.72E-15
72	373.163	2.77E-15	3.00E-15	2.66E-15

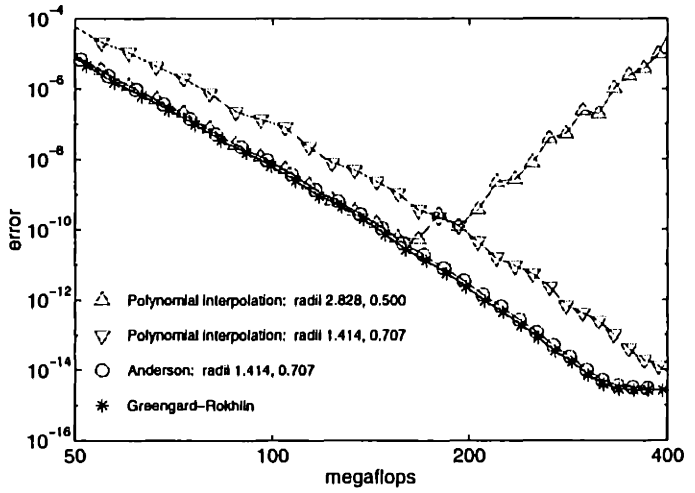
Table 4.7: Normalized ℓ^2 error in polynomial interpolation algorithm with 40K random charges and five hierarchy levels.

		$R_O = \sqrt{2}, R_I = 1/\sqrt{2}$			$R_O = 2\sqrt{2}, R_I = 1/2$		
p	Mflops	UNIF.	CORNER	CENTER	UNIF.	CORNER	CENTER
9	49.812	5.28E-05	6.20E-05	4.33E-05	7.05E-06	9.77E-06	5.12E-06
10	54.830	2.16E-05	2.11E-05	1.88E-05	2.50E-06	3.52E-06	1.76E-06
11	60.370	9.02E-06	1.09E-05	8.54E-06	8.52E-07	1.24E-06	6.01E-07
12	66.431	3.60E-06	4.33E-06	2.85E-06	3.03E-07	5.05E-07	1.90E-07
13	73.014	1.56E-06	1.98E-06	1.34E-06	1.27E-07	2.12E-07	8.02E-08
14	80.118	6.32E-07	7.61E-07	4.66E-07	4.70E-08	7.53E-08	2.47E-08
15	87.743	2.30E-07	2.28E-07	1.92E-07	1.55E-08	2.60E-08	7.34E-09
16	95.891	9.56E-08	1.37E-07	7.09E-08	6.17E-09	1.18E-08	2.95E-09
17	104.559	4.78E-08	8.18E-08	3.33E-08	2.52E-09	5.23E-09	1.18E-09
18	113.749	2.51E-08	2.29E-08	1.28E-08	9.76E-10	1.87E-09	3.88E-10
19	123.461	8.27E-09	7.79E-09	6.20E-09	4.08E-10	6.64E-10	1.44E-10
20	133.694	4.40E-09	4.93E-09	2.65E-09	1.66E-10	3.33E-10	5.80E-11
21	144.448	1.71E-09	2.30E-09	9.50E-10	6.47E-11	1.52E-10	2.20E-11
22	155.724	9.00E-10	1.09E-09	3.69E-10	2.88E-11	5.70E-11	9.54E-12
23	167.521	3.16E-10	3.62E-10	1.73E-10	3.37E-11	5.37E-11	1.57E-11
24	179.840	1.47E-10	2.27E-10	8.41E-11	1.85E-10	2.77E-10	1.01E-10
25	192.680	6.90E-11	1.27E-10	3.11E-11	7.86E-11	1.26E-10	2.83E-11
26	206.042	3.92E-11	4.54E-11	1.32E-11	2.09E-10	3.68E-10	7.33E-11
27	219.926	1.08E-11	1.56E-11	6.02E-12	1.36E-09	2.24E-09	3.99E-10
28	234.330	6.66E-12	9.34E-12	3.16E-12	1.61E-09	2.71E-09	4.75E-10
29	249.256	2.97E-12	5.45E-12	1.24E-12	4.48E-09	7.91E-09	9.56E-10
30	264.704	1.99E-12	2.30E-12	4.36E-13	2.30E-08	3.92E-08	5.74E-09
31	280.673	5.43E-13	6.87E-13	2.21E-13	3.01E-08	5.31E-08	6.15E-09
32	297.164	2.50E-13	4.17E-13	1.06E-13	1.41E-07	2.44E-07	3.01E-08
33	314.176	1.22E-13	2.49E-13	3.97E-14	1.23E-07	2.00E-07	2.11E-08
34	331.710	9.39E-14	1.04E-13	1.82E-14	6.15E-07	1.05E-06	1.06E-07
35	349.765	2.10E-14	4.07E-14	7.47E-15	1.39E-06	2.47E-06	2.48E-07
36	368.341	1.40E-14	1.90E-14	5.01E-15	2.08E-06	3.86E-06	3.67E-07
37	387.439	6.60E-15	1.28E-14	3.58E-15	5.20E-06	9.63E-06	9.18E-07
38	407.059	5.00E-15	6.17E-15	2.69E-15	3.53E-05	6.27E-05	4.64E-06

UNIFORM distribution



CORNER distribution



CENTER distribution

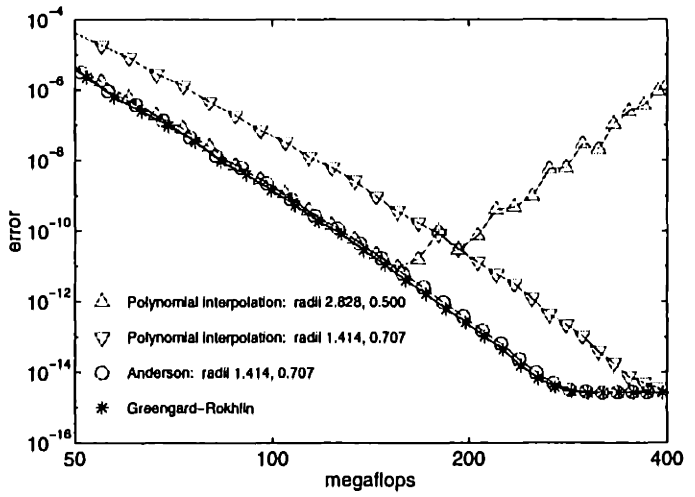


Figure 4-6: Plot of normalized ℓ^2 error against flop count for *POT2D_EVAL* with five hierarchy levels and 40,960 charges. There are 40 random charges in each of the 1024 leaf-level boxes. These graphs contain the same information as Tables 4.5, 4.6, and 4.7.

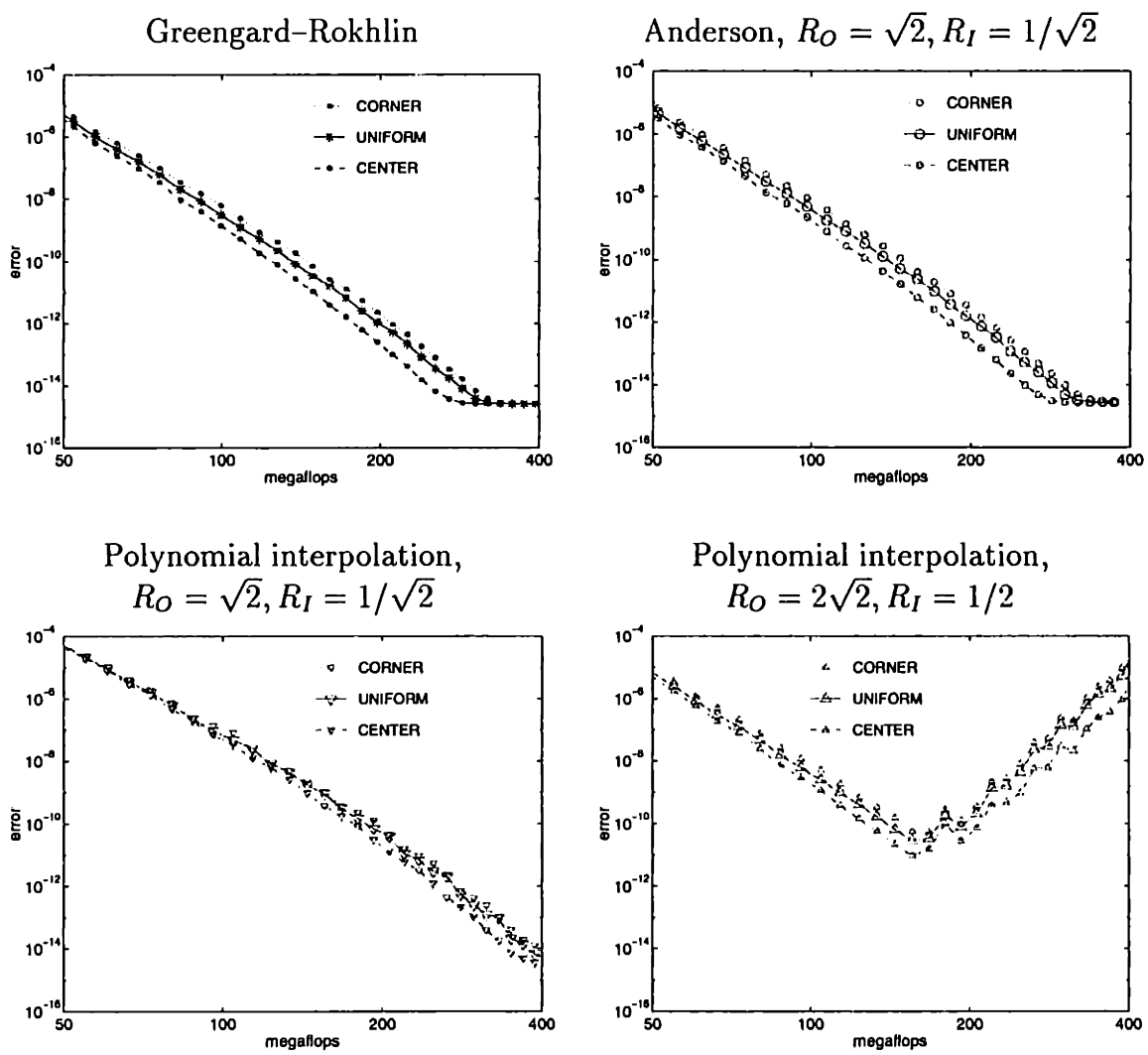


Figure 4-7: Plot of normalized ℓ^2 error against flop count for $POT2D_EVAL$ with five hierarchy levels and 40,960 charges. Each graph is for a separate method. There are 40 random charges in each of the 1024 leaf-level boxes.

Bibliography

[1] R. C. Agarwal, F. G. Gustavson, and M. Zubair. Exploiting functional parallelism of POWER2 to design high-performance numerical algorithms. *IBM Journal of Research and Development*, 38(5):563–576, 1994.

[2] T. Agerwala, J. L. Martin, J. H. Mirza, D. C. Sadler, D. M. Dias, and M. Snir. SP2 system architecture. *IBM Systems Journal*, 34(2):152–184, 1995.

[3] Christopher R. Anderson. An implementation of the fast multipole method without multipoles. *SIAM J. Sci. Stat. Comput.*, 13(4):923–947, 1992.

[4] Harry C. Andrews and C. L. Patterson. Singular value decomposition (SVD) image coding. *IEEE Trans. Commun.*, COM-24:425–432, 1976.

[5] David H. Bailey. FFTs in external or hierarchical memory. *Journal of Supercomputing*, 4(1):23–35, 1990.

[6] Edward G. Benson, David C. P. LaFrance-Linden, Richard A. Warren, and Santa Wiryaman. Design of Digital's Parallel Software Environment. *Digital Technical Journal*, 7(3):24–38, 1995.

[7] W. T. Cochrane, J. W. Cooley, J. W. Favin, D. L. Helms, R. A. Kaenel, W. W. Lang, G. C. Maling, D. E. Nelson, C. M. Rader, and P. D. Welch. What is the fast Fourier transform? *IEEE Trans. Audio and Electroacoustics*, AU-15:45–55, 1967.

[8] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19:297–301, 1965.

[9] Thomas H. Cormen and David M. Nicol. Performing out-of-core FFTs on parallel disk systems. Technical Report PCS-TR96-294, Dartmouth College, September 1996.

[10] A. Dutt, M. Gu, and V. Rokhlin. Fast algorithms for polynomial interpolation, integration and differentiation. *SIAM J. Numer. Anal.*, 33(5):1689–1711, 1996.

[11] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data, II. *Appl. Comput. Harmon. Anal.*, 2:85–100, 1995.

[12] Dieter Gaier. *Lectures on Complex Approximation.* Birkhäuser, Boston, 1987.

[13] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore and London, 3rd edition, 1996.

[14] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, 1987.

[15] L. Greengard and V. Rokhlin. On the efficient implementation of the fast multipole algorithm. Technical Report RR-602, Yale University, Department of Computer Science, November 1988.

[16] Leslie Greengard and Vladimir Rokhlin. A new version of the Fast Multipole Method for the Laplace equation in three dimensions. In *Acta Numerica*, volume 6, pages 229–269. Cambridge University Press, 1997.

[17] Leslie F. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems.* The MIT Press, Cambridge, MA, 1988.

[18] Ulf Grenander and Gábor Szegő. *Toeplitz Forms and their Applications.* University of California Press, Berkeley and Los Angeles, 1958.

[19] F. Alberto Grünbaum. Eigenvectors of a Toeplitz matrix: discrete version of the prolate spheroidal wave functions. *SIAM J. Alg. Disc. Meth.*, 2:136–141, 1981.

[20] Ming Gu. *Studies in Numerical Linear Algebra.* PhD thesis, Yale University, November 1993.

[21] S. K. S. Gupta, C.-H. Huang, P. Sadayappan, and R. W. Johnson. Implementing fast Fourier transforms on distributed-memory multiprocessors using data redistributions. *Parallel Processing Letters*, 4(4):477–488, 1994.

[22] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann, San Francisco, 2nd edition, 1996.

[23] Tomasz Hrycak and Vladimir Rokhlin. An improved fast multipole algorithm for potential fields. Research Report YALEU/DCS/RR-1089, Yale University, Department of Computer Science, November 1995.

[24] Y. Charlie Hu and S. Lennart Johnsson. On the accuracy of Poisson's formula based fast N-body algorithms. Technical Report TR-06-96, Harvard University, Center for Research in Computing Technology, January 1996.

[25] M. J. D. Powell. *Approximation theory and methods.* Cambridge University Press, 1981.

[26] John H. Reif and Stephen R. Tate. N-body simulation I: Fast algorithms for potential field evaluation and Trummer's problem. WWW page, `http://www.cs.unt.edu/home/srt/papers/nbody1.ps`, 1997.

[27] D. Slepian. Prolate spheroidal wave functions, Fourier analysis, and uncertainty V: The discrete case. *Bell System Tech. J.*, 57:1371–1430, 1978.

[28] Paul N. Swarztrauber. Vectorizing the FFT. In G. Rodrigue, editor, *Parallel Computations*, pages 51–83. Academic Press, 1982.

[29] Paul N. Swarztrauber. Multiprocessor FFTs. *Parallel Computing*, 5:197–210, 1987.

[30] Sivan Toledo. On the communication complexity of the discrete Fourier transform. *IEEE Signal Processing Letters*, 3(6):171–172, 1996.

[31] Charles F. Van Loan. *Computational Frameworks for the Fast Fourier Transform.* SIAM, Philadelphia, 1992.

[32] J. M. Varah. The prolate matrix. *Linear Algebra Appl.*, 187:269–278, 1993.

[33] S. W. White and S. Dhawan. POWER2: Next generation of the RISC System/6000 family. *IBM Journal of Research and Development*, 38(5):493–502, 1994.

[34] N. Yarvin and V. Rokhlin. An improved fast multipole algorithm for potential fields on one-dimensional structures. Research Report YALEU/DCS/RR-1119, Yale University, Department of Computer Science, January 1997.