Forward and inverse wave propagation using bandlimited functions and a fast reconstruction algorithm for electron microscopy

Kristian Sandberg
Department of Applied Mathematics
University of Colorado at Boulder[1]

July 25, 2003[2]

[1]`kristian.sandberg@colorado.edu`
[2]Re-formatted version of the PhD-thesis with the same name submitted by the author to Graduate School at University of Colorado on July 10, 2003.

Abstract

The thesis consists of three major parts. First, we develop an algorithm for solving the wave equation in two dimensions with spatially varying coefficients. In what is a new approach, we use the basis of approximate prolate spheroidal wavefunctions and construct derivative operators that incorporate boundary and interface conditions. Writing the wave equation as a first-order system, we evolve the equation in time using the matrix exponential. Computation of the matrix exponential requires efficient representation of operators in two dimensions and for this purpose we use short sums of one-dimensional operators. We also use a partitioned low-rank representation in one dimension to further speed up the algorithm. We demonstrate that the method significantly reduces numerical dispersion and the computational time in comparison with a fourth-order finite difference scheme in space with the explicit fourth-order Runge-Kutta solver in time.

Second, using efficient representations of spectral projectors, we develop a stable solver for initial value problems on space-like surfaces. By writing the Helmholtz equation in two dimensions as an initial value problem in space and using spectral projectors, we construct a numerically stable scheme for propagating the solution. This solver is a first step toward a fast algorithm for solving inverse scattering problems in two and higher dimensions.

Finally, we implement a fast Fourier summation algorithm for tomographic reconstruction of biological data sets obtained via transmission electron microscopy. For two-dimensional images, the new algorithm scales as $O(N_\theta M \log M) + O(MN \log N)$ operations, compared to $O(N_\theta MN)$ for the standard filtered back projection, where N_θ is the number of projection angles and $M \times N$ is the size of the reconstructed image. For typical data sets, the new algorithm is 1.5-2.5 times faster than computing the filtered backprojection using the direct summation. The speed advantage is greater as the size of the data sets grow. The new algorithm also allows us to use higher order spline interpolation of the data without additional computational cost.

Acknowledgments

This thesis could not have been completed without the help and support from several people. First, let me thank Gregory Beylkin for supervising the thesis, and for all the encouragement and help. I would also like to thank my committee members Bradley Alpert, James Curry, Keith Julien, and Richard McIntosh for their continuous interest in my research and valuable suggestions for improvements. I would like to thank Lucas Monzón for providing valuable help with some of the mathematical background for this thesis, Fernando Pérez for help with various computer issues, and our department secretary Laurie Conway for all the administrative help.

I would like to thank my dear parents Christina and Ulf Sandberg for all the encouragement, interest, support, and good advice over the years. My sisters Sofia, Katarina, and Ulrika have always been a source of support. Let me also mention my grandfather Bert Sandberg who taught me how to multiply at an early age.

Finally, my warmest thanks to Dawn Boiani for her endless support, encouragement, and inspiration throughout the completion of this thesis.

Contents

Chapter 1

Introduction

This thesis covers two main topics: numerical algorithms for wave propagation and a fast tomographic reconstruction algorithm for transmission electron microscopy. The main contributions of this thesis are:

- using bases for bandlimited functions as a numerical tool in algorithms of wave propagation,
- developing and using efficient representations of operators in two dimensions for purposes of wave propagation,

and

- constructing a fast algorithm for tomographic reconstruction of thick biological specimens for the transmission electron microscopy.

Using bases for bandlimited functions allows us to achieve a low oversampling rate while significantly reducing numerical dispersion. By using the operator representations introduced in this thesis for wave propagation, application of the matrix exponential for large time steps becomes feasible as a numerical propagation scheme.

1.1 Wave propagation

There are numerous applications of wave propagation in acoustics, elasticity and electro magnetics, including medical imaging, sonar, seismology, radar, and noise modeling in civil engineering. Few problems can be solved analytically, and almost all wave propagation problems in engineering and geophysics involve numerical solutions of equations of acoustics, elasticity, and other hyperbolic systems. Current methods have difficulties in applications involving varying material properties or complex geometries.

We develop fast and accurate numerical algorithms to solve (as an example) the equation of acoustics

$$\kappa(x)u_{tt} = (\sigma(x)u_x)_x \tag{1.1}$$

with spatially varying material parameters $\kappa(x)$ and $\sigma(x)$. The coefficients $\kappa(x)$ and $\sigma(x)$ represent the compressibility and the specific volume of the medium, respectively. The goal is to construct schemes where we control the bandwidth and accuracy. Our approach is based on the following ideas:

- We choose a basis that works well for problems with variable medium parameters, non-periodic boundary conditions, and material interfaces.
- By constructing derivative operators with nearly uniform error distribution for frequencies within a given bandwidth we reduce numerical dispersion.
- We use integration by parts to incorporate boundary and interface conditions into the derivative operator.
- By formulating the acoustic equation as a first order system we use the operator exponential for time evolution. This requires an efficient operator representation in higher dimensions.

In this thesis, we choose so-called bandlimited functions (to be introduced in Chapter 2 below) as the function basis. We construct derivative operators and time evolution schemes for this basis. (Our construction, however, is applicable to any piecewise smooth basis.) The operator representations that we will use to efficiently compute the matrix exponential are useful also in other applications, such as computational quantum mechanics. Also, although in this thesis we consider problems in two dimensions, our approach can be generalized to three dimensions.

There is a wide literature devoted to the topic of wave propagation, see e.g. Durran [20] and references therein, and we refer only to a few sources that are related to our approach. For a general treatment of wave phenomena, see Whitham [55]. Acoustic waves are treated in Brekhovskiskh and Godin [13] which also contains an extensive bibliography on the subject. Iserles [33] gives an overview of finite difference methods for solving hyperbolic problems, and Fornberg [25] discusses the use of pseudo-spectral methods. Reformulation of a broad class of wave propagation problems as first order systems is considered in Bazer and Burridge [3]. Alpert et al. [2] use an integral evolution formula to solve the wave equation.

1.1.1 Bandlimited functions

The first step in our approach is to select a basis for representing the solution and operators. Traditionally, the trigonometric functions $\{e^{k\pi x}\}_{k=0}^{N}$ have been used for periodic problems, and Legendre and Chebyshev polynomials have been used for non-periodic problems.

In this thesis, we consider bandlimited functions restricted to an interval (see Section 2.5 for a precise definition). There are several bases available for computations using bandlimited functions (see Xiao et al. [56], and Beylkin and Monzon [9]). In this thesis we review the construction of three bases based on functions of the type $\{e^{c\theta_k x}\}_{k=1}^N$ where $|\theta_k| < 1$. Since we do not impose that $\theta_k = k\pi$, these functions are not necessarily periodic. The basis spanned by $\{e^{c\theta_k x}\}_{k=1}^N$ is usually not suitable for numerical computations since these functions are in general not orthogonal. Instead, we form linear combinations of these basis functions to construct approximations to the prolate spheroidal wave functions which were introduced as a basis for bandlimited functions by Slepian et al. in a series of papers [54], [38], [39], [51], and [52]. The resulting basis can be shown to be almost orthonormal and, therefore, suitable for numerical computations. In [9] it is shown that this choice of basis gives low sampling rates also for non-periodic functions and functions that are "almost" bandlimited.

1.1.2 Derivative operators with respect to bandlimited functions

The next step to solve the acoustic equation (1.1) is to discretize the equation in space. Since our basis functions are linear combinations of the exponentials $\{e^{c\theta_k x}\}_{k=1}^N$, we can differentiate these basis functions exactly. This is similar to what is done in pseudo-spectral methods [25] which are commonly based on periodic trigonometric functions or Chebyshev polynomials. One advantage of using bandlimited functions instead of polynomials as the basis is that the norm of the derivative matrix based on bandlimited functions is smaller than the norm of the derivative matrix based on polynomials.

We incorporate boundary conditions into the derivative matrices using integration by parts. In the case of discontinuous interface conditions, interface conditions are also incorporated into the derivative matrix using integration by parts. This technique has previously been used by Alpert et al. in [1].

1.1.3 Time evolution and operator representations

To evolve the solution of the acoustic equation (1.1) in time, we first write the equation as a first order system in time. After discretizing the spatial operator of the equation, the equation takes the form of the system of linear first order ordinary differential equations

$$\begin{aligned} \mathbf{u}_t &= L\mathbf{u} + \mathbf{F}(t) \\ \mathbf{u}(0) &= \mathbf{u_0}. \end{aligned}$$

In the case of time independent material coefficients, the solution is given by

$$\mathbf{u}(t) = e^{tL}\mathbf{u_0} + e^{tL}\int_0^t e^{-\tau L}\mathbf{F}(\tau)\, d\tau.$$

If there is no time-dependent force, the solution can be computed by a sequence of matrix-vector multiplications,

$$\mathbf{u}(t_k) = e^{\Delta t L}\mathbf{u}(t_{k-1}), \tag{1.2}$$

where the time step Δt can be chosen arbitrary large without causing instabilities.

The computation of $e^{\Delta t L}$ and the matrix-vector multiplications in (1.2) are computationally costly in dimensions two and higher and, therefore, this approach is rarely used for numerical computations. We use the separated representation introduced by Beylkin and Mohlenkamp [7] to represent the operator L for problems in two or higher dimensions. This representation significantly reduces the computational cost for computing the matrix exponential and matrix-vector multiplications. The separated representation of an operator in two or higher dimensions is given by a sum of operators in one dimension. We refer to the number of terms in the separated representation as the separation rank. The separation rank for the matrix exponential $e^{\Delta t L}$ grows with the size of the time step Δt, and we will see that a time step between 1-2 temporal periods is appropriate to control both the separation rank and the number of time steps. We reduce the computational cost further by representing the operators in one dimension by introducing the so-called Partitioned Low Rank (PLR) representation which is similar to the partitioned singular value decomposition considered by Jones et al. [34], and by Beylkin et al. [10]. We note that both the separated representation and the PLR representation are interesting on their own, with applications in other areas, e.g., computational quantum mechanics (see Beylkin and Mohlenkamp [7] and [8]).

1.1.4 Wave propagation on space-like surfaces

As an application of the tools for wave propagation, we consider wave propagation on space-like surfaces. This problem appears in some approaches for solving the inverse problem where sound waves are propagated through a domain with an unknown scatterer. By measuring the scattered wave field at a surface outside the scatterer, the goal is to determine the structure of the scatterer. In particular, we consider ultrasound tomography and an approach by Natterer and Wübbeling [44] which involves repeated solution of equations on the form

$$\begin{aligned}
&u_{yy} = -u_{xx} - \kappa(x,y)\omega^2(1+f)u \equiv Au, \;\; (x,y)\in[-1,1]\times[-1,1]\\
&u(x,-1) = g(x)\\
&u_y(x,-1) = h(x)\\
&u(-1,y) = r(y)\\
&u(1,y) = s(y)
\end{aligned} \quad . \tag{1.3}$$

This equation is an initial value problem for the Helmholtz equation and we refer to it as wave propagation on space-like surfaces. As posed in (1.3), this equation is unstable. In many applications, for example underwater acoustics, seismics, and Synthetic Aperture Radar (SAR), this problem is stabilized by replacing the operator in (1.3) by one-way operators. Such approach excludes waves going in the opposite direction, e.g. multiple reflections. An alternative approach has been proposed in [44], where this problem is solved for constant coefficient κ by using Fourier-techniques to filter out the high frequencies of the solution. In this paper, we consider the case of x-dependent coefficient κ. We show that by forcing the spatial operator A to be negative definite, we obtain an equation which can be solved in a stable manner by computing the matrix exponential. In order to obtain a negative definite operator, we review the work by Beylkin et al. [10] who present fast algorithms for computing spectral projectors for self-adjoint operators. We extend this algorithm to diagonalizable matrices with pure real or imaginary spectrum and apply the technique to solve (1.3) for constant and x-dependent coefficients.

1.2 Biological imaging

There is a significant interest in studying the fine structure of cells and tissues. By using high voltage Transmission Electron Microscopy, biologists reconstruct three-dimensional images of cell structures. A large amount of information is collected in the form of three-dimensional images, and an important task is to use this information to build three-dimensional models of cell structures. Image analysis is an important tool for gaining a deeper understanding of biological structures of cells and tissues (Ladinsky et al. [37]).

The modeling process includes specimen preparation followed by imaging of the specimen using electron microscopy. The three-dimensional density of the specimen is computed by tomographic reconstruction. The resulting data set is segmented (boundaries of certain objects are labeled) and rendered into surfaces that can be visualized in three dimensions (Kremer et al. [36]). The modeling process is time consuming and it is desirable to make some of the steps automatic. This requires careful understanding of the decisions that are currently being made by biologists.

The reconstruction process and the image segmentation stages involve a number of challenging mathematical problems. Tomographic reconstruction is of great importance in many fields, including seismic imaging, Magnetic Resonance Imaging (MRI), x-ray tomography, and in electron microscopy. We consider the study of thick biological specimen using transmission electron microscopy. This application involves some difficulties that has to be addressed by a successful reconstruction algorithm. The range of angles that can be used for illuminating the specimen is limited to a range of angles, typically between $\pm 70^0$, and the measurement usually contain a significant amount of noise. Also, the specimen is significantly deformed during the experiment, which means that the angles usually are not equally spaced.

In this thesis we develop a fast algorithm for tomographic reconstructions of transmission electron microscopy data. The goal is to construct a fast reconstruction algorithm that produces the same results as the direct summation technique [28] traditionally used for electron microscopy tomography of thick specimen. The goal is to construct an algorithm that scales as $O(N^2 \log N)$ compared to $O(N^3)$ for the direct summation technique.

Our approach is based on the reconstruction technique known as filtered backprojection or direct summation (see, e.g., Deans [18] and Gilbert [28]), where the reconstruction formula takes the form of a sum. Instead of summing in the space domain, we express the sum in Fourier space where we can use the Unequally Spaced Fast Fourier Transform (USFFT) introduced by Dutt and Rokhlin [21], and by Beylkin [4], for efficient summation. The resulting algorithm has been incorporated into the software package IMOD [32],[36] developed by The Boulder Laboratory for 3-D Electron Microscopy of Cells at University of Colorado at Boulder.

1.3 New results

Although this thesis combines tools from a number of papers, in particular [9], [1], [7], [10], and [4], there are some results and applications that are new:

- The use of bandlimited functions for numerical solutions of partial differential equations.
- The construction of an algorithm for solving the acoustic equation in two dimensions with variable coefficients with a significant reduction of numerical dispersion compared to a fourth order finite difference scheme. The time evolution method allows large time steps and is significantly faster than using the explicit Runge-Kutta 4 solver.
- The use of spectral projectors for numerical solutions of wave propagation problems on space-like surfaces.
- The construction of fast operator calculus algorithms for matrices represented in the PLR form. The PLR form is demonstrated to be an efficient representation for many matrices that are not compressible with wavelet-techniques and the singular value decomposition.
- The construction of spectral derivative matrices incorporating boundary and interface conditions. The thesis generalizes existing methods to non-orthogonal bases and demonstrates how the use of spectral projectors improves the conditioning of the derivative matrices.
- Miscellaneous results for the approximate prolate spheroidal wave functions introduced in [9].

- The construction of a fast algorithm for tomographic reconstruction of thick biological specimen using transmission electron microscopy. The new algorithm is shown to be faster and to provide more flexibility than the commonly used technique filtered back projection (a.k.a. direct summation).

1.4 Speed comparisons

For the speed comparisons in this thesis, we used a Dell computer running Red Hat Linux 8.0 with a Pentium 4 2.56GHz processor and a memory of 1GB RAM with 512kB cache. The programs were written in Fortran 77 using (non-optimized) BLAS and LAPACK routines for linear algebra operations. We used the the g77 compiler with the compiler flags `-O3 -march=pentium3 -mmmx -msse -malign-double -funroll-loops`.

1.5 Outline of the thesis

We begin with a review of the bandlimited functions in Chapter 2 where we also include a few new results. In the third chapter, we construct derivative matrices incorporating boundary and interface conditions with respect to an arbitrary set of (smooth) basis functions. We apply the tools from Chapter 3 to bandlimited functions in Chapter 4 where we provide several numerical examples demonstrating the accuracy of the derivative matrices based on bandlimited functions. The chapter also includes a section on the construction of integration matrices with respect to bandlimited functions. In Chapter 5 we review the separated representation representation used for representing operators in two or higher dimensions. We provide algorithms for linear algebra operations for operators in this representation. We also introduce the PLR representation, provide algorithms for linear algebra operations for operators in this representation, and also demonstrate the efficiency of this representation for computing matrix products.

The tools from Chapter 2-5 are applied to solving the acoustic equation in two dimensions in Chapter 6 where we give a number of numerical examples and comparisons with a fourth order finite difference scheme. We consider inverse problems and computations of spectral projectors in Chapter 7.

Finally, we construct a fast algorithm for tomographic reconstruction of thick biological specimen using transmission electron microscopy in Chapter 8. The paper "A fast algorithm for electron microscopy tomography" by Beylkin, Mastronarde, and Sandberg is included in Appendix E.

Chapter 2

Bandlimited functions

In this chapter we study bandlimited functions and present numerical tools for using them. Our main motivation for using bandlimited functions for numerical analysis is that solutions of PDEs behave more like exponentials than polynomials which comprise the main tool used today in computations. Using bandlimited functions allows us to significantly reduce the computational cost and achieve any desired accuracy. In this thesis we use bandlimited functions for studying wave propagation.

Waves are often decomposed into frequency components, with the frequency content of the wave given by its Fourier transform. In physical phenomena there is a bound for the frequency range we can expect from the solution. It is therefore natural to describe such phenomena by functions with compactly supported Fourier transforms. We refer to such functions as bandlimited functions; a precise definition will be given later. If the Fourier transform of a function is supported on $[-c, c]$, we refer to c as the bandwidth of the function.

For applications in this thesis, we will study bandlimited functions restricted to a finite interval, e.g., $[-1, 1]$. Since a function cannot have compact support in both the space and the frequency domains, it is important to fully understand how functions on an interval can be extended onto the real line. Slepian et al. in a series of papers [54], [38], [39], [51], and [52], introduced the prolate spheroidal wave functions as an eigensystem which can be shown to be bandlimited and maximally concentrated within the interval $[-1, 1]$. Although these functions were introduced for signal processing, their use has been limited, perhaps due to lack of fast or reliable algorithms. In this chapter, we will see how these functions can be approximated with a finite number of exponentials of the type $\{e^{ic\theta_k x}\}_k$ where $|\theta_k| < 1$.

Exponentials also provide a natural basis for wave phenomena. Bandlimited periodic waves are typically expanded into the Fourier basis $e^{ik\pi x}$ for $k = 0, 1, \ldots, N$ or, equivalently, $\{\cos k\pi x, \sin k\pi x\}$, for $k = 0, 1, \ldots, N$. If we consider bandlimited waves with zero boundary conditions, it is natural to expand such waves into the functions $\sin \frac{k\pi(x+1)}{2}$ for $k = 1, 2, \ldots, N$. In this thesis we will study wave phenomena on domains where we expect piecewise smooth solutions. We divide the domain into subdomains, and on each subdomain we expect the solution to be bandlimited. The global solution may not be differentiable and thus cannot be truly bandlimited, but can be described as bandlimited on subdomains. In such cases, we need to allow arbitrary boundary conditions on the subdomains, and neither the Fourier nor the sin basis will be efficient. This motivates the introduction of a basis that can represent functions of the type e^{ibx} for an arbitrary (real) value of b such that $|b| < c$, where c is a maximum allowable bandwidth. These functions are not generally periodic. Bandlimited functions with arbitrary boundary conditions also appear when studying problems on a finite domain with absorbing boundary conditions.

The chapter is organized as follows. In the first section we give some preliminary definitions and results, and in the following section we introduce bandlimited functions defined on the real line. In Section 2.3 we review the prolate spheroidal wave functions and state some of their properties, and then in Section 2.4 we introduce bandlimited functions on an interval. In Section 2.5 we review quadratures for bandlimited functions on an interval and show how a finite number of exponentials can approximate any bandlimited function of a fixed bandwidth within a fixed but arbitrary precision. The following section considers three bases for approximating bandlimited functions on an interval. We show that one such basis approximates the prolate spheroidal wave functions. We conclude this chapter by approximating trigonometric functions, Chebyshev polynomials, and Gaussian bell functions using a finite number of exponentials.

2.1 Preliminaries

Throughout this thesis, the Fourier transform of $u \in L^1(\mathbb{R})$ is defined as

$$\hat{u}(\omega) = \int_{-\infty}^{\infty} u(x) e^{-ix\omega} \, dx.$$

The inverse Fourier transform of $\hat{u} \in L^1(\mathbb{R})$ is defined by $\frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{u}(\omega) e^{i\omega x} \, d\omega$. For functions such that $u, \hat{u} \in L^1(\mathbb{R})$ we have that

$$u(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{u}(\omega) e^{i\omega x} \, d\omega \tag{2.1}$$

almost everywhere. If $u \in L^2(\mathbb{R})$ then there exists a sequence of functions $\{u_n\}_n \subseteq L^2(\mathbb{R}) \cap L^1(\mathbb{R})$ such that $\lim_{n\to\infty} u_n = u$ in the L^2-norm. We define the Fourier transform of such functions as $\hat{u} = \lim_{n\to\infty} \hat{u}_n$ in the L^2-norm.

For functions defined on the interval $[-1, 1]$ we will usually use the norms

$$\|u\|_2 = \sqrt{\int_{-1}^{1} |u(x)|^2 \, dx}$$

and

$$\|u\|_\infty = \max_{x\in[-1,1]} |u(x)|.$$

Occasionally, we extend these functions to functions on $\mathbb{R}$ and use the norms

$$\|u\|_{L^2(\mathbb{R})} = \sqrt{\int_{-\infty}^{\infty} |u(x)|^2 \, dx}$$

and

$$\|u\|_{L^\infty(\mathbb{R})} = \max_{x\in\mathbb{R}} |u(x)|$$

for the extended functions.

2.2 The space of bandlimited functions

In this section we introduce bandlimited functions on the real line. We review properties of such functions and state a bound for the derivative. Following Landau and Pollak [38], the bandlimited functions are defined as

Definition 1 (Bandlimited functions) *Let $c > 0$ and define the space*

$$\mathcal{B}_c = \{f \in L^2(\mathbb{R}) \mid \hat{f}(\omega) = 0 \text{ for } |\omega| > c\}.$$

We refer to $\mathcal{B}_c$ as the space of bandlimited functions of bandwidth c.

We note that the space of bandlimited functions equipped with the standard inner product is a closed linear subspace of $L^2(\mathbb{R})$ and, therefore, a Hilbert space. As an example, consider the function

$$u(x) = \frac{c}{\pi}\operatorname{sinc}(cx) = \frac{\sin cx}{\pi x}. \tag{2.2}$$

This function is square integrable on the real line and its Fourier transform is given by

$$\hat{u}(\omega) = \left\{ \begin{array}{ll} 1, & \omega \in [-c, c] \\ 0, & \omega \notin [-c, c] \end{array} \right. .$$

In some situations, we will use the following space which is dense in $\mathcal{B}_c$.

Definition 2 *Let $c > 0$ and define the space*

$$\mathcal{F}_c = \mathcal{B}_c \cap L^1(\mathbb{R}).$$

As an example, consider the function

$$v(x) = \frac{c}{2\pi}\operatorname{sinc}^2(\frac{cx}{2}).$$

This function is absolutely and square integrable on the real line. The Fourier transform is given by

$$\hat{v}(\omega) = \begin{cases} 1 - \frac{|\omega|}{c}, & \omega \in [-c, c] \\ 0, & \omega \notin [-c, c] \end{cases}.$$

Hence, $v \in \mathcal{F}_c$. Note that the function u in (2.2) is not absolutely integrable, and hence u does not belong to $\mathcal{F}_c$.

The space $\mathcal{B}_c$ has the following properties.

Proposition 3 *The space of bandlimited functions $\mathcal{B}_c$ has the following properties:*

1. *Every function $u \in \mathcal{B}_c$ can be written as*

$$u(x) = \frac{1}{2\pi}\int_{-c}^{c} \hat{u}(\omega)e^{i\omega x}\, d\omega$$

almost everywhere.

2. *Let $u \in \mathcal{B}_c$ and define $\alpha > 0$ by*

$$\alpha = \frac{\|u\|_{L^2([-1,1])}}{\|u\|_{L^2(\mathbb{R})}}.$$

Then

$$\left\|\frac{du}{dx}\right\|_{L^2([-1,1])} \le c\,\|u\|_{L^2(\mathbb{R})} = \frac{c}{\alpha}\|u\|_{L^2([-1,1])}.$$

3. *For every $u \in \mathcal{B}_c$, there exists a $\tilde{u} \in C^\infty(\mathbb{R})$ such that $u = \tilde{u}$ almost everywhere.*

Proof. (1) Let $\{u_n\}_n \subseteq \mathcal{F}_c$ be a sequence converging to $u \in \mathcal{B}_c$. Since $\hat{u}$ is defined as a limit in the L^2-norm of functions $\hat{u}_n \in L^2(\mathbb{R})$ and is supported on $[-c, c]$, Hölder's inequality gives that $\hat{u} \in L^1(\mathbb{R})$. Hence,

$$\tilde{u}(x) = \frac{1}{2\pi}\int_{-c}^{c} \hat{u}(\omega)e^{i\omega x}\, d\omega$$

is well-defined. We have that

$$\begin{aligned} \|u - \tilde{u}\|_{L^2(\mathbb{R})} &= \|u - u_n + u_n - \tilde{u}\|_{L^2(\mathbb{R})} \le \|u - u_n\|_{L^2(\mathbb{R})} + \|u_n - \tilde{u}\|_{L^2(\mathbb{R})} \\ &\le \|u - u_n\|_{L^2(\mathbb{R})} + \frac{1}{2\pi}\|\hat{u}_n - \hat{u}\|_{L^2(\mathbb{R})} \end{aligned}$$

and since $u_n \to u$ and $\hat{u}_n \to \hat{u}$ in the L^2-norm, $u = \tilde{u}$ almost everywhere.

(2) This follows from Bernstein's inequality. (See, e.g., Meyer [43, Ch. 2.5] and references therein.)

(3) This follows from the second part of the proposition. $\square$

2.3 The Prolate Spheroidal Wave Functions

In this section we review the prolate spheroidal wave functions and consider some of their properties. These functions have been studied by, e.g., Flammer [24] and Slepian et al. [54],[38]. Properties of the functions are reviewed by Slepian [53]. Numerical tools for these functions are given by Bouwkamp [11], Xiao et al. [56], and by Beylkin and Monzon [9].

The prolate spheroidal wave functions are defined as

Definition 4 (Prolate spheroidal wave functions) *Consider the bandwidth $c > 0$ and define the operator $F_c : L^2([-1,1]) \to L^2([-1,1])$ by*

$$F_c(\psi)(\omega) = \int_{-1}^{1} e^{icx\omega}\psi(x)\ dx, \tag{2.3}$$

and the operator $Q_c : L^2([-1,1])] \to L^2([-1,1])]$ by

$$Q_c(\psi)(\omega) = \frac{1}{\pi}\int_{-1}^{1} \frac{\sin(c(\omega - x))}{\omega - x}\psi(x)\ dx = \frac{c}{2\pi}F_c^* F_c \psi.$$

The eigenfunctions ψ of Q_c and F_c are called prolate spheroidal wave functions .

Each eigenvalue λ of F_c corresponds to an eigenvalue μ of Q_c by

$$\mu = \frac{c|\lambda|^2}{2\pi}. \tag{2.4}$$

The prolate spheroidal wave functions depend on the bandwidth c for which they are constructed, but we will suppress this dependence in our notation.

Let us state some properties of the prolate spheroidal wave functions.

Theorem 5 *The prolate spheroidal wave functions are complete in $L^2([-1,1])$ and $\mathcal{B}_c$.*

For a proof, see [54]. The eigenfunctions $\psi_j(x)$ are real and orthogonal on both $[-1,1]$ and $\mathbb{R}$,

$$\int_{-1}^{1} \psi_i(x)\psi_j(x)dx = \delta_{ij}, \tag{2.5}$$

and

$$\int_{-\infty}^{\infty} \psi_i(x)\psi_j(x)dx = \frac{1}{\mu_i}\delta_{ij} \tag{2.6}$$

where μ_i is the eigenvalue of the operator Q_c. This normalization is more convenient for our purposes, although in the original paper by Slepian and Pollak [54], the prolate spheroidal wave functions are normalized on $\mathbb{R}$ instead. The prolate spheroidal wave functions are uniformly bounded on $[-1,1]$. More precisely, there exists a bandwidth dependent constant K_c such that

$$\|\psi_j\|_\infty \leq K_c \tag{2.7}$$

for all $j = 0, 1, \dots$. As discussed in [9], the existence of K_c can be proved by using that the prolate spheroidal wave functions approach the Legendre polynomials for $j \gg c$.

The eigenvalues of Q_c are real and the spectrum is naturally divided into three parts. For large bandlimits c, the first $2\pi/c$ eigenvalues μ_i of Q_c are close to 1. The next $\log c$ eigenvalues decay exponentially fast to zero and the remaining eigenvalues are very close to zero. By construction, we have

$$e^{ictx} = \sum_{j=0}^{\infty} \lambda_j \psi_j(t)\psi_j(x) \tag{2.8}$$

for all $t, x \in [-1,1]$. This is the optimal separated representation for e^{ictx}. For a given c, once $j > 2c/\pi$, the series can be truncated due to the exponential decay of λ_j.

For the prolate spheroidal wave functions we establish a bound for their derivatives on the interval $[-1,1]$. It is a simple property, although we have not seen it stated explicitly elsewhere.

Theorem 6 *Let ψ_j be the j:th prolate spheroidal wave function with bandwidth c and let μ_j be the corresponding eigenvalue to Q_c defined in Definition 4. Then*

$$\left\|\frac{d\psi_j}{dx}\right\|_2 \leq c\|\psi_j\|_{L^2(\mathbb{R})} = c\frac{1}{\sqrt{\mu_j}}.$$

Proof. The proof follows from Bernstein's inequality in Proposition 3 and from $\|\psi_j\|_{L^2(\mathbb{R})} = \frac{1}{\sqrt{\mu_j}}$. □

It is interesting to compare this bound to another version of Bernstein's inequality (see, e.g., Rivlin [47, Ch. 2.4]), which states that if $p(x)$ is an n:th degree polynomial and $|p(x)| \leq 1$ for $x \in [-1,1]$, then

$$|p'(x)| \leq n^2, \ x \in [-1,1].$$

2.4 Bandlimited functions on an interval

In many applications we are interested in bandlimited functions restricted to a finite interval. We study a representation of such functions using exponentials in this section and show how such representations provide local approximations of bandlimited functions in $\mathcal{B}_c$. Following [9] we define

Definition 7 (Bandlimited functions on an interval) *Define the space $\mathcal{E}_c$ of bandlimited functions of bandwidth c on an interval as*

$$\mathcal{E}_c = \left\{ u \in L^\infty([-1,1]) \mid u(x) = \sum_{k\in\mathbb{Z}} a_k e^{ib_k x} : \{a_k\}_k \in l^1,\ b_k \in [-c,c] \right\}.$$

We characterize the space $\mathcal{E}_c$ in the following theorem.

Theorem 8 *The space $\mathcal{E}_c$ of bandlimited functions on an interval satisfies the following properties:*

1. *$\mathcal{E}_c \subseteq C^\infty([-1,1])$*
2. *For every $\epsilon > 0$ and $u \in \mathcal{F}_c$ there exists a function $\tilde{u} \in \mathcal{E}_c$ such that $|u(x) - \tilde{u}(x)| < \epsilon$ almost everywhere on $[-1,1]$.*
3. *For every $\epsilon > 0$ and $u \in \mathcal{E}_c$ there exists a function $\tilde{u} \in \mathcal{B}_c$ such that $\|u - \tilde{u}\|_2 < \epsilon$.*
4. *For every $\epsilon > 0$ and $u \in \mathcal{B}_c$ there exists a function $\tilde{u} \in \mathcal{E}_c$ such that $\|u - \tilde{u}\|_2 < \epsilon$.*

In the proof of (2) below, we discretize an integral using the Riemann sum. This is usually not an efficient way for computing an integral, but we use it for the simplicity of the proof.

Proof. (1) Let $u \in \mathcal{E}_c$. Then the sequence $\{b_k\}$ is bounded, and it is easily shown that the family of functions $\{e^{ib_k x}\}_k$ is equicontinuous on $[-1,1]$. Since $\{a_k\}_k \in l^1$ it follows that

$$u(x) = \sum_{k\in\mathbb{Z}} a_k e^{ib_k x}$$

is continuous on $[-1,1]$ and hence $\mathcal{E}_c \subseteq C([-1,1])$. By the definition of $\mathcal{E}_c$ we have that the n:th derivative of u is given by

$$u^{(n)}(x) = \sum_{k\in\mathbb{Z}} (ib_k)^n a_k e^{ib_k x}.$$

Since $\{a_k\}_k \in l^1$ and $|(ib_k)^n| \leq c^n$ it follows that $\{(ib_k)^n a_k\}_k \in l^1$ and hence $u^{(n)}(x) \in \mathcal{E}_c$ for each $n = 0, 1, \ldots$. Since every function in $\mathcal{E}_c$ is continuous, it follows that $\mathcal{E}_c \subseteq C^\infty([-1,1])$.

(2) By Proposition 3 it follows that any function $u \in \mathcal{F}_c$ can be written on the form

$$u(x) = \int_{-c}^{c} \sigma(\omega) e^{i\omega x} \, d\omega$$

almost everywhere and since $u \in L^1(\mathbb{R})$, σ is continuous and bounded. Define $b_k = -c + \frac{2kc}{N}$ for $k = 1, \dots, N$. Then $|b_k| \leq c$ and since the integrand is continuous and bounded, we can approximate u with the Riemann sum

$$u(x) = \frac{2c}{N} \sum_{k=1}^{N} \sigma(b_k) e^{ib_k x} + E_N(x)$$

where $\lim_{N\to\infty} E_N(x) = 0$ for all $x \in [-1, 1]$. We choose N sufficiently large such that $\|E_N\|_\infty < \epsilon$. Define $a_k = 2c\sigma(b_k)/N$ for $k = 1, \dots, N$ and

$$\tilde{u}(x) = \sum_{k=1}^{N} a_k e^{ib_k x}$$

for $x \in [-1, 1]$. Then $|u(x) - \tilde{u}(x)| < \epsilon$ almost everywhere on$\in [-1, 1]$. Furthermore, $\{a_k\}_k \in l^1$ and therefore $\tilde{u}$ is bounded on $[-1, 1]$.

(3) Any function $u \in \mathcal{E}_c$ can be written as

$$u(x) = \sum_{k\in\mathbb{Z}} a_k e^{ib_k x}.$$

For $x \in \mathbb{R}$, define

$$\tilde{u}(x) = \sum_{j=0}^{\infty} u_j \psi_j(x)$$

where ψ_j is the j:th prolate spheroidal wave function, and

$$u_j = \lambda_j \left(\sum_{k\in\mathbb{Z}} a_k \psi_j(b_k/c) \right)$$

where λ_j is the eigenvalue corresponding to ψ_j according to Definition 4. Introduce

$$\tilde{u}_N(x) = \sum_{j=0}^{N} u_j \psi_j(x).$$

By using (2.8) it follows that $u(x) = \tilde{u}(x)$ for all $x \in [-1, 1]$. Since $\tilde{u} \in L^2([-1, 1])$ and $\{\psi_j\}_j$ forms a complete basis in this space according to Theorem 5, we can choose N sufficiently large so that $\|u - \tilde{u}_N\|_2 < \epsilon$. Furthermore, $\psi_j \in \mathcal{B}_c$ for $j = 0, 1, \dots, N$ and hence $\tilde{u}_N \in \mathcal{B}_c$.

(4) Let $u \in \mathcal{B}_c$. Then, since $\mathcal{F}_c$ is dense in $\mathcal{B}_c$, there exists a function $v \in \mathcal{F}_c$ such that

$$\|u - v\|_2 < \epsilon/2. \tag{2.9}$$

From the second part of the theorem we know that there exists $\tilde{u} \in \mathcal{E}_c$ such that $|v(x) - \tilde{u}(x)| < \frac{\epsilon}{2\sqrt{2}}$ almost everywhere on $[-1, 1]$. Then

$$\|v - \tilde{u}\|_2^2 = \int_{-1}^{1} |v(x) - \tilde{u}(x)|^2 \, dx \le \int_{-1}^{1} \frac{\epsilon^2}{8} \, dx < \epsilon^2/4$$

which combined with (2.9) gives us that

$$\|u - \tilde{u}\|_2 \le \|u - v\|_2 + \|v - \tilde{u}\|_2 < \epsilon.$$

□

2.5 Approximation of bandlimited functions on an interval

The goal of this section is to show that any bandlimited function on an interval of bandlimit c can be approximated by a linear combination of a finite number of exponentials in the form $e^{ic\theta_k x}$ where $|\theta_k| \le 1$. The phases θ_k are chosen as nodes for quadratures for bandlimited functions. We establish the existence of such quadratures for bandlimited functions in the following theorem.

Theorem 9 *Let σ be a real, non-negative, integrable weight function supported in $[-\nu, \nu]$, $0 \le \nu \le 1/2$, and let ϵ and γ be positive numbers with $\gamma < 1$. Then, for N sufficiently large, there exist real constants $\{v_1, \dots, v_N\}$ and $\{t_1, \dots, t_N\}$, with $w_j > 0$ and $|t_j| < \nu$, such that*

$$\left| \int_{-1}^{1} \sigma(t) e^{i\pi t y} \, dt - \sum_{j=1}^{N} v_j e^{i\pi t_j y} \right| < \epsilon, \text{ for } |y| \le \gamma N + 1.$$

Furthermore, there exists a positive integer m and positive constants α_m and d_m (independent of N), such that the error ϵ is bounded by

$$\epsilon < 2\|\sigma\|_1 \left(3\nu^{2m} + \frac{2}{2 + (2+\sqrt{3})^N + (2-\sqrt{3})^N} + \frac{2d_m e^{-\alpha_m(1-\gamma)N}}{1 - e^{-\alpha_m}} \right).$$

For a proof, see [9, Theorem 6.1].

The constants α_m and d_m in the error estimate, are related to properties of the exponential Euler splines, see [9, Theorem 6.1] and [50, pp. 29,30, and 35]. This theorem establishes the existence of a quadrature and provides an error estimate. Actual computations, however, are based on a slightly different algorithm where a large number of the weights are sufficiently close to zero to be disregarded (see [9] for more details). Therefore, the number of nodes and weights actually used is typically significantly less than the number of nodes and weights required by the error estimate.

The theorem above is more general than our needs. We next state a special case of this theorem which will be the foundation for our applications.

Corollary 10 *Let c and ϵ be positive numbers. Then, for N sufficiently large, there exist constants $\{w_1, \ldots, w_N\}$ and $\{\theta_1, \ldots, \theta_n\}$, such that for any $x \in [-1, 1]$*

$$\left| \int_{-1}^{1} e^{ictx} \, dt - \sum_{k=1}^{N} w_k e^{ic\theta_k x} \right| < \epsilon \tag{2.10}$$

where w_k are real and non-negative, and $|\theta_k| < 1$.

Let $\gamma \in (0, 1)$ and let m and N be positive integers such that $N \geq \frac{2\gamma c}{\pi}$. The error ϵ is bounded by

$$\epsilon \leq 4 \left(3 \left(\frac{c}{\gamma N \pi} \right)^{2m} + \frac{2}{2 + (2+\sqrt{3})^N + (2-\sqrt{3})^N} + \frac{2 d_m e^{-\alpha_m (1-\gamma) N}}{1 - e^{-\alpha_m}} \right)$$

where α_m and d_m are positive constants independent of N.

Proof. See Appendix 1. □

We note that error is of the form $O(N^{-2m}) + O(c_m^{-N})$, where c_m is a constant, for any integer m. Hence, the error decays faster than any polynomial decay. However, due to the third term in the estimate, the constant c_m depends on m and therefore this estimate is mostly of theoretical interest.

Definition 11 (Quadrature for bandlimited functions) *For $c > 0$ and $\epsilon > 0$, suppose N quadrature nodes $\theta_1, \theta_2, \ldots, \theta_N$ and weights $w_1, w_2, \ldots, w_N$ are such that (2.10) is satisfied. Let E denote the matrix with elements given by $E_{kl} = e^{ic\theta_k \theta_l}$.*

If $\det(E) \neq 0$ then we say the nodes and weights are a quadrature for bandlimited functions of bandwidth c and accuracy ϵ.

The condition $\det(E) \neq 0$ is not necessary if the nodes and weights are going to be used only for integrating bandlimited functions. In Section 2.6.2 below, we construct an approximation to the prolate spheroidal wave functions using the functions $e^{ic\theta_k x}$, and this construction depends on the invertibility of the matrix E. Although we do not have a proof that our construction guarantees that $\det(E) \neq 0$, we have not encountered any counterexamples in our experiments.

For the construction of the quadrature nodes in [9], we have the symmetry properties

$$\theta_k = -\theta_{N-k+1} \tag{2.11}$$

and

$$w_k = w_{N-k+1}. \tag{2.12}$$

We use the quadrature nodes and weights to construct a basis for $\mathcal{E}_c$ according to the following theorem.

Theorem 12 *Consider the bandwidth c and a function $u \in \mathcal{E}_c$ represented by*

$$u(x) = \sum_{k\in\mathbb{Z}} a_k e^{ib_k x}.$$

Let $\epsilon > 0$, and let $\{\theta_l\}_{l=1}^N$ and $\{w_l\}_{l=1}^N$ be a set of quadrature nodes and weights for bandwidth $2c$ and accuracy ϵ^2. Then there exist constants $\{u_l\}_{l=1}^N$ and A such that

$$\left\| u(x) - \sum_{l=1}^{N} u_l e^{ic\theta_l x} \right\|_\infty \leq A \left(\sum_{k\in\mathbb{Z}} |a_k| \right) \epsilon$$

and

$$\left\| u(x) - \sum_{l=1}^{N} u_l e^{ic\theta_l x} \right\|_2 \leq \sqrt{2} A \left(\sum_{k\in\mathbb{Z}} |a_k| \right) \epsilon.$$

Proof. By definition, $u(x) = \sum_{k\in\mathbb{Z}} a_k e^{ib_k x}$ for some set of constants a_k and b_k where $\{a_k\}_k \in l^1$ and $|b_k| \leq c$. We use [9, Theorem 8.1] which asserts that there exist constants α_{kl} for $k \in \mathbb{Z}$ and $l = 1, \ldots, N$, and a constant A such that

$$\left\| e^{ib_k x} - \sum_{l=1}^{N} \alpha_{kl} e^{ic\theta_l x} \right\|_\infty \leq A\epsilon.$$

According to the proof of Theorem 8.1 in [9], $\alpha_{kl} = w_l \sum_{j=0}^{N-1} \psi_j(b_k/c)\psi_j(\theta_l)$ where ψ_j denotes the j:th prolate spheroidal wave function. According to (2.7) these functions satisfy $||\psi_j||_\infty \leq K_c$ for $j = 0, 1, \ldots$. Since the quadrature weights w_l are bounded, the sequence $\{\alpha_{kl}\}$ is bounded. The sequence $\{a_k\}_k \in l^1$ by assumption and, therefore, $u_l = \sum_{k\in\mathbb{Z}} a_k \alpha_{kl}$ is well-defined for $l = 1, \dots, N$. It follows that

$$\left\| u(x) - \sum_{l=1}^{N} u_l e^{ic\theta_l x} \right\|_\infty = \left\| \sum_{k\in\mathbb{Z}} a_k e^{ib_k x} - \sum_{k\in\mathbb{Z}} a_k \left(\sum_{l=1}^{N} \alpha_{kl} e^{ic\theta_l x} \right) \right\|_\infty$$

$$= \left\| \sum_{k\in\mathbb{Z}} a_k \left(e^{ib_k x} - \sum_{l=1}^{N} \alpha_{kl} e^{ic\theta_l x} \right) \right\|_\infty \leq A \left(\sum_{k\in\mathbb{Z}} |a_k| \right) \epsilon.$$

For the $L^2([-1,1])$-norm we have that

$$\left\| u(x) - \sum_{l=1}^{N} u_l e^{ic\theta_l x} \right\|_2^2 = \int_{-1}^{1} \left| u(x) - \sum_{l=1}^{N} u_l e^{ic\theta_l x} \right|^2 dx$$

$$\leq \int_{-1}^{1} \left\| u(x) - \sum_{l=1}^{N} u_l e^{ic\theta_l x} \right\|_\infty^2 dx$$

$$\leq 2A^2 \left(\sum_{k\in\mathbb{Z}} |a_k| \right)^2 \epsilon^2$$

and hence

$$\left\| u(x) - \sum_{l=1}^{N} u_l e^{ic\theta_l x} \right\|_2 \leq \sqrt{2} A \left(\sum_{k\in\mathbb{Z}} |a_k| \right) \epsilon.$$

□

The error estimate in the previous theorem depends on the sum $\sum_{k\in\mathbb{Z}} |a_k|$ which is the upper bound for both the function on $[-1,1]$ and on $\mathbb{R}$. Nevertheless, numerical experiments indicate strongly that this representation gives sufficient accuracy which we illustrate in Section 2.7.

2.6 Bases for bandlimited functions on an interval

In the previous section we represented bandlimited functions on an interval using exponential functions. We also discussed how we can construct quadratures for bandlimited functions on an interval, and use the resulting nodes and weights to construct a finite dimensional subspace that approximates the space of bandlimited functions on an interval within a finite but arbitrary precision. Even though the resulting basis of exponential functions is convenient to study from an analytical view point, this basis is not well-conditioned and therefore not always suitable for direct numerical computations.

In this section we illustrate the basis of exponential functions and introduce two additional bases; an approximation of the prolate spheroidal wave functions and an interpolating basis. All three bases are derived in [9].

2.6.1 Exponentials

We define the basis of exponentials for bandlimited functions on an interval as follows.

Definition 13 (Basis of exponentials) *Given the bandwidth $c > 0$ and $\epsilon > 0$ construct N quadrature nodes $\{\theta_k\}_k$ according to Definition 11. The sequence of functions given by $\{e^{ic\theta_k x}\}_{k=1}^{N}$ is called a basis of exponentials for bandlimited functions on an interval of bandwidth c and accuracy ϵ. We define an N-dimensional space $\mathcal{E}_{c,\epsilon}$ of bandlimited functions on the interval $[-1, 1]$ as*

$$\mathcal{E}_{c,\epsilon} = \text{span}\left\{e^{ic\theta_l x}\right\}_{l=1}^{N}.$$

In Figure 2.1 we display the real part of two basis functions constructed for the bandwidth $c = 8.5\pi$ and the accuracy $\epsilon = 10^{-7}$ which results in 32 nodes. Note that these trigonometric functions are not periodic.

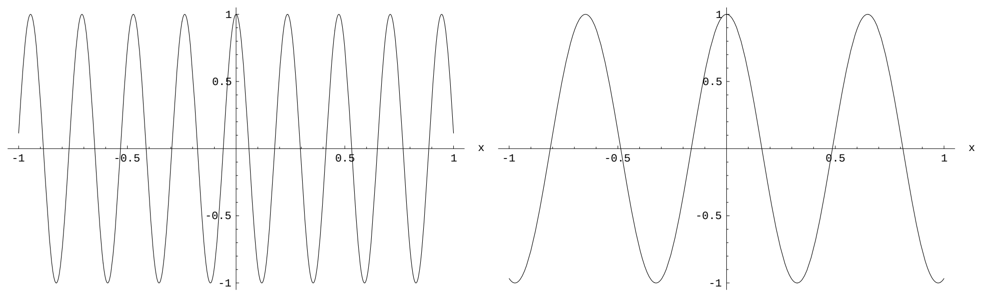


Figure 2.1: In the first figure we display the real part of the basis function $e^{ic\theta_1 x}$ where $c = 8.5\pi$ and $\theta_1 \simeq -0.9957$. In the second figure we display the real part of the function $e^{ic\theta_{12} x}$ where $c = 8.5\pi$ and $\theta_{12} \simeq -0.3626$.

The basis of exponentials has the advantage of being easy to visualize and plot since the basis functions have a closed form expression. Furthermore, differentiation and integration can be represented by diagonal matrices with respect to this basis. However, these functions have the disadvantage of forming an ill-conditioned basis which means that this basis is usually not suitable for numerical computations. We explore this issue further in Section 2.7.

2.6.2 Approximate prolate spheroidal wave functions

In the last section we pointed out that a disadvantage of the exponential basis was the ill-conditioning of the basis. We saw in Section 2.3 that the prolate spheroidal wave functions are bandlimited and form a complete orthogonal basis in $\mathcal{B}_c$ and in $L^2([-1,1])$. This motivates us to seek an approximation to these functions. Such approximations were described in [56] and [9]. In this thesis we use the approach in [9] which approximates prolate spheroidal wave functions by a linear combination of the exponential basis functions defined in the previous section. In this section, we first define a set of "approximate" prolate spheroidal wave functions . We then show how they can be constructed and that they form a set which is "almost" orthonormal. Finally, we show that these functions indeed form a basis of the space $\mathcal{E}_{c,\epsilon}$ introduced in the previous section.

By discretizing the integral operator (2.3) we define the approximate prolate spheroidal wave functions as follows.

Definition 14 (Approximate prolate spheroidal wave functions)
Given the bandwidth $c > 0$ and $\epsilon > 0$, construct N quadrature nodes $\{\theta_l\}_{l=1}^N$ and weights $\{w_l\}_{l=1}^N$ according to Definition 11. Consider the algebraic eigenvalue problem

$$\sum_{l=1}^{N} w_l e^{ic\theta_m\theta_l}\Psi_{\mathbf{j}}(\theta_l) = \eta_j \Psi_{\mathbf{j}}(\theta_m). \tag{2.13}$$

Define the approximate prolate spheroidal wave functions on $[-1,1]$ by

$$\Psi_j(x) = \frac{1}{\eta_j}\sum_{l=1}^{N} w_l e^{icx\theta_l}\Psi_{\mathbf{j}}(\theta_l) \tag{2.14}$$

where $\Psi_{\mathbf{j}}(\theta_l)$ are the eigenvectors defined by (2.13).

By Definition 11 we assume that the matrix with elements given by $e^{ic\theta_k\theta_l}$ is invertible. Since all the quadrature weights w_l are positive, it follows that the matrix with elements given by $w_l e^{ic\theta_k\theta_l}$ is invertible and hence $\eta_j \neq 0$.

The algebraic eigenvalue problem defining the approximate prolate spheroidal wave functions approximates the eigenvalue problem (2.3). The operator F_c defined in (2.3) has a countable spectrum. Therefore, we expect that the eigenvalues $\{\eta_j\}_j$ to the algebraic eigenvalue problem approximate the eigenvalues $\{\lambda_j\}_j$ of F_c. Experimentally, we have found that this is the case within high precision, with exceptions for the smallest eigenvalues, where the relative error may be large. We have also found that the approximate prolate spheroidal wave function do not necessarily match the "true" prolate spheroidal wave function as individual functions due to the initial $\sim 2\pi/c$ eigenvalues in Definition 14 being so close to each other that they are numerically indistinguishable. However, the subspace spanned by the initial approximate prolate spheroidal wave functions approximates well the corresponding subspace spanned by the initial true prolate spheroidal wave functions. In Figure 2.2 we plot four approximate prolate spheroidal wave functions constructed for the bandwidth $c = 8.5\pi$ and the accuracy $\epsilon = 10^{-7}$ resulting in 32 quadrature nodes. Note that the first approximate prolate spheroidal wave function (the upper left plot) happens to have one root while the first true prolate spheroidal wave function has no root. The first approximate prolate spheroidal wave function is in this case similar to the second true prolate spheroidal wave function. As the eigenvalues decay, they become numerically distinguishable and corresponding eigenfunctions match well until the eigenvalues become smaller than $\sim \sqrt{\epsilon}$. In this case the corresponding eigenfunctions appear similar, but numerically they are different.

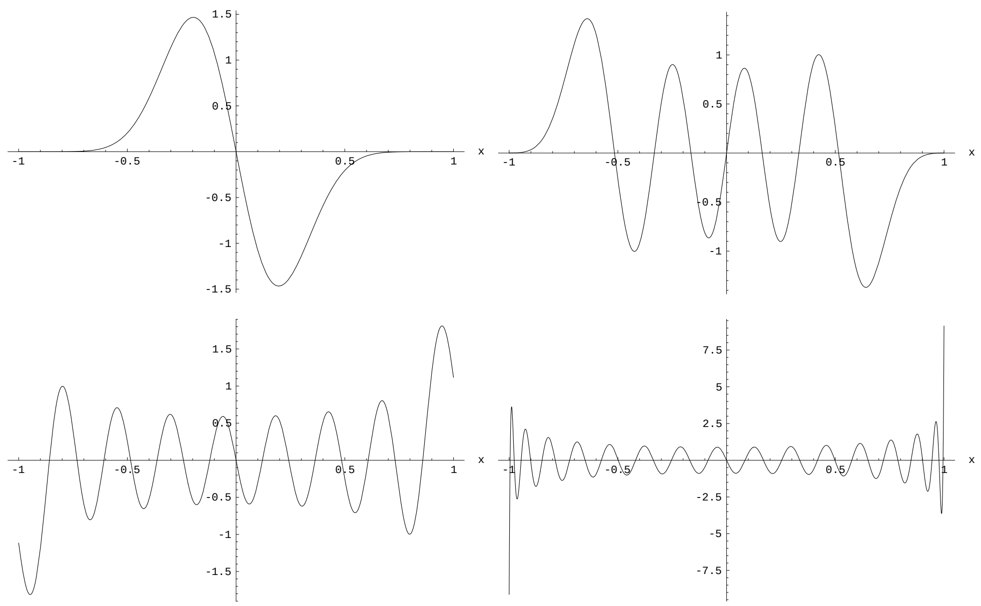


Figure 2.2: Four approximate prolate spheroidal wave functions constructed for the bandwidth $c = 8.5\pi$ with the accuracy $\epsilon = 10^{-7}$ resulting in 32 linearly independent functions. In the upper left figure we display the first approximate prolate spheroidal wave function, in the upper right the eighth, in the lower left the 16:th, and in the lower right the 32:nd.

Let us establish some properties for the eigenvectors and eigenvalues used to define the approximate prolate spheroidal wave functions. In particular, we establish that the approximate prolate spheroidal wave functions are either odd or even.

Proposition 15 *Let $\Psi_\mathbf{j}$ denote an eigenvector and let η_j denote the corresponding eigenvalue to the eigenvalue problem defined by (2.13). Define A as the matrix with elements given by*

$$A_{kl} = \sqrt{w_k} e^{ic\theta_k\theta_l} \sqrt{w_l}.$$

Define $\mathbf{q^j}$ as the vector with elements given by $\mathbf{q^j}_m = \sqrt{w_m}\Psi_{\mathbf{j}}(\theta_m)$. Then the following properties hold.

1. *A is normal and $A^* = \bar{A}$.*
2. *There exists a real and orthonormal set of eigenvectors of A.*
3. *The vector $\mathbf{q^j}$ is an eigenvector of A with the eigenvalue η_j.*
4. *The vector $\mathbf{q^j}$ is a left eigenvector of A with the eigenvalue η_j.*
5. *The eigenvalue η_j is either pure real or pure imaginary.*
6. *The vectors $\mathbf{q^j}$ and $\Psi_{\mathbf{j}}$ are either even or odd, that is, $\mathbf{q^j}_l = \pm\mathbf{q^j}_{N-l+1}$ and $\Psi^{\mathbf{j}}(\theta_l) = \pm\Psi^{\mathbf{j}}(\theta_{N-l+1})$.*

Proof. (1) This property follows immediately from the expression for the elements of A.

(2) See [9, Proposition 8.2].

(3) From the definition of A and $\mathbf{q^j}$ we have that

$$\begin{aligned}(A\mathbf{q^j})_k &= \sum_{m=1}^{N} \sqrt{w_k}e^{ic\theta_k\theta_m}\sqrt{w_m}\sqrt{w_m}\Psi_{\mathbf{j}}(\theta_m) = \sqrt{w_k}\sum_{m=1}^{N} w_m e^{ic\theta_k\theta_m}\Psi_{\mathbf{j}}(\theta_m)\\ &= \sqrt{w_k}\eta_j\Psi_{\mathbf{j}}(\theta_k) = \eta_j\mathbf{q^j}_k\end{aligned}$$

where we used that $\Psi_{\mathbf{j}}$ is an eigenvector to the eigenvalue problem (2.13).

(4) Using the first three properties of the proposition we have that

$$\mathbf{q^j}^* A = (A^*\mathbf{q^j})^* = (\overline{A}\mathbf{q^j})^* = (\overline{A\mathbf{q^j}})^* = \eta_j\mathbf{q^j}^*.$$

(5) Let $\mathbf{q}$ be a real eigenvector to the N-by-N-matrix A. Since $\{\theta_k\}_k$ are quadrature nodes it follows from Definition 11 that $\operatorname{rank}(A) = N$ and that the eigenvalue η of A is non-zero. Since $\mathbf{q}$ is real we have that $\mathbf{q}$ is an eigenvector to $\bar{A}$ with the eigenvalue $\bar{\eta}$. Let us write η as $\eta = \alpha + i\beta$ where α and β are real. Then

$$\begin{cases} A\mathbf{q} = (\alpha + i\beta)\mathbf{q} \\ \bar{A}\mathbf{q} = (\alpha - i\beta)\mathbf{q} \end{cases} \Rightarrow \begin{cases} A_r\mathbf{q} = \alpha\mathbf{q} \\ A_i\mathbf{q} = \beta\mathbf{q} \end{cases}$$

where $A_r = \operatorname{Re}(A)$ and $A_i = \operatorname{Im}(A)$. Denote the range and the null space of a matrix as $\mathcal{R}$ and $\mathcal{N}$, respectively. To prove property (5) of the theorem it suffices to show that $\mathcal{N}(A_r) = \mathcal{R}(A_i)$.

The elements of A_r are given by $(A_r)_{kl} = \sqrt{w_k}\cos(c\theta_k\theta_l)\sqrt{w_l}$. From the symmetry properties (2.11) and (2.12), it follows that

$$(A_r)_{N-k+1,l} = \sqrt{w_{N-k+1}}\cos(c\theta_{N-k+1}\theta_l)\sqrt{w_l} = \sqrt{w_k}\cos(-c\theta_k\theta_l)\sqrt{w_l} = (A_r)_{kl}.$$

Hence, if N is even, we have that $\mathrm{rank}(A_r) \leq N/2$ and by the same argument $\mathrm{rank}(A_i) \leq N/2$.

Assume that there exists an eigenvector $\mathbf{q}$ such that $\mathbf{q}\in\mathcal{N}(A_r)\cap\mathcal{N}(A_i)$. Then $\alpha = \beta = \eta = 0$ which contradicts that $\eta \neq 0$. Hence, if $\mathbf{q}\in\mathcal{N}(A_r)$, then $\mathbf{q}\in\mathcal{R}(A_i)$. Therefore,

$$\mathcal{N}(A_r)\subseteq\mathcal{R}(A_i). \tag{2.15}$$

Assume that $\mathrm{rank}(A_r) < N/2$. Then $\dim(\mathcal{N}(A_r)) \geq N/2+1$ which by (2.15) implies that $\mathrm{rank}(A_i) \geq N/2+1$ which is a contradiction. Hence, $\mathrm{rank}(A_r) = N/2$ and by the same argument it follows that $\mathrm{rank}(A_i) = N/2$. Since $\dim(\mathcal{N}(A_r)) = \mathrm{rank}(A_i) = N/2$ it follows from (2.15) that $\mathcal{N}(A_r) = \mathcal{R}(A_i)$. (The case for odd N is analogous).

(6) Recall that the quadrature weights are real and even in the sense that $w_m = w_{N-m+1}$, and that the quadrature nodes are real and odd such that $\theta_m = -\theta_{N-m+1}$. Using the symmetry of the nodes θ_m it follows that

$$A_{N-m+1,l} = \sqrt{w_{N-m+1}}e^{ic\theta_{N-m+1}\theta_l}\sqrt{w_l} = \sqrt{w_m}e^{-ic\theta_m\theta_l}\sqrt{w_l} = \overline{A_{ml}}. \tag{2.16}$$

Therefore, since $\mathbf{q^j}$ is an eigenvector of A with eigenvalue η,

$$\mathbf{q^j}_{N-m+1} = \frac{1}{\eta_j}\sum_{l=1}^{N} A_{N-m+1,l}\mathbf{q^j}_l = \frac{1}{\eta_j}\sum_{l=1}^{N}\overline{A_{ml}}\mathbf{q^j}_l. \tag{2.17}$$

Since $\bar{A} = A^*$ by the first part of the theorem and $\mathbf{q^j}$ can be chosen to be real by the second part of the theorem, it follows that $\mathbf{q^j}$ is an eigenvector to $\bar{A}$ with eigenvalue $\overline{\eta_j}$. Since the eigenvalues η_j are either pure real or pure imaginary by property (5), equation (2.17) implies that

$$\mathbf{q^j}_{N-m+1} = \frac{\overline{\eta_j}}{\eta_j}\mathbf{q^j}_m = \pm\mathbf{q^j}_m.$$

We now have that

$$\Psi_{\mathbf{j}}(\theta_{N-m+1}) = \frac{1}{\sqrt{w_{N-m+1}}}\mathbf{q^j}_{N-m+1} = \pm\frac{1}{\sqrt{w_m}}\mathbf{q^j}_m = \pm\Psi_{\mathbf{j}}(\theta_m). \tag{2.18}$$

$\square$

Corollary 16 *The eigenfunctions $\Psi_j(x)$ defined in Definition 14 are even or odd.*

Proof. Using (2.14) and property (6) in Proposition 15 we have that

$$\Psi_j(-x) = \frac{1}{\eta_j}\sum_{l=1}^{N} w_l e^{-icx\theta_l}\Psi_{\mathbf{j}}(\theta_l) = \pm\frac{1}{\eta_j}\sum_{l=1}^{N} w_{N-l+1}e^{icx\theta_{N-l+1}}\Psi_{\mathbf{j}}(\theta_{N-l+1})$$
$$= \pm\frac{1}{\eta_j}\sum_{l=1}^{N} w_l e^{icx\theta_l}\Psi_{\mathbf{j}}(\theta_l) = \pm\Psi_j(x) \tag{2.19}$$

□

Let us study the conditioning of this basis by considering the matrix S with elements defined by

$$S_{ij} = \int_{-1}^{1} \Psi_i(x)\Psi_j(x)\, dx \tag{2.20}$$

for $i, j = 1, \ldots, N$. Even though the prolate spheroidal wave functions are orthogonal, this is not true for the approximate prolate spheroidal wave functions. Using Corollary 16, we next show that the matrix S is close to the identity matrix.

Proposition 17 *The functions Ψ_m and Ψ_n are nearly orthogonal and the elements of S defined in (2.20) satisfy*

$$|S_{mn} - \delta_{mn}| \le \begin{cases} \frac{\epsilon^2 \sum_{k=1}^{N} w_k}{|\eta_m||\eta_n|} & \textit{if } \Psi_m(x) \textit{ and } \Psi_n(x) \textit{ are both even or both odd} \\ 0 & \textit{otherwise} \end{cases}.$$

Proof. By definition, $S_{mn} = \int_{-1}^{1} \Psi_m(x)\Psi_n(x)\, dx$ and by Corollary 16 we know that the functions $\Psi_i(x)$ are either even or odd. Therefore the integral vanishes when the functions have different parity. The error estimate for the other case is shown in [9, Proposition 8.1]. □

The magnitude of η_j decreases monotonically and $\eta_1 \sim \sqrt{2\pi/c}$. Since η_N is typically chosen to be close to ϵ, the accuracy we seek, the matrix S deviates significantly from the identity matrix only when both η_m and η_n are small and close to ϵ. We give a numerical example of the condition number of S in Table 1.1.

Let us introduce the matrices Q, W, H, and $\tilde{S}$ with elements given by

$$Q_{ij} = \sqrt{w_i}\Psi_j(\theta_i), \tag{2.21}$$

$$W_{ij} = \begin{cases} \sqrt{w_i} & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}, \tag{2.22}$$

$$H_{ij} = \begin{cases} \frac{1}{\eta_i} & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}, \tag{2.23}$$

and

$$\tilde{S}_{ij} = \begin{cases} 2 & \text{if } i = N - j + 1 \\ 2\frac{\sin(c(\theta_i+\theta_j))}{c(\theta_i+\theta_j)} & \text{if } i \neq N - j + 1 \end{cases}. \tag{2.24}$$

Using these matrices it follows from (2.14) that

$$\Psi_j(x) = H_{jj} \sum_{l=1}^{N} W_{ll} Q_{lj} e^{icx\theta_l}. \tag{2.25}$$

The inner product matrix S defined in (2.20) is given by

$$\begin{aligned} S_{mn} = (\Psi_n, \Psi_m) &= \int_{-1}^{1} \Psi_m(x)\Psi_n(x)\ dx \\ &= \int_{-1}^{1} H_{mm} \sum_{k,l=1}^{N} W_{kk} Q_{km} e^{icx\theta_k} H_{nn} W_{ll} Q_{ln} e^{icx\theta_l}\ dx \\ &= H_{mm} \left(\sum_{k,l=1}^{N} Q^T_{mk} W_{kk} \tilde{S}_{kl} W_{ll} Q_{ln} \right) H_{nn}. \end{aligned} \tag{2.26}$$

Hence, the matrix S can be computed by the matrix product

$$S = HQ^T W \tilde{S} W Q H. \tag{2.27}$$

The condition numbers of $\tilde{S}$ and H may be very large. Since η_N is typically chosen to be of order ϵ, the accuracy we seek, this means that the condition number of H grows as $\sim \frac{1}{\epsilon}$. In particular, we see that to compute S_{NN}, the summation in (2.26) is multiplied by a factor of the order $\frac{1}{\epsilon^2}$. Since the matrix H is badly conditioned, the matrices S and K should be computed using extended precision, and then truncated to the desired precision. This is computationally expensive, but in most applications this matrix can be pre-computed and tabulated and will not affect the speed of the applications. When constructing differentiation matrices in Chapter 3 we will see that it is necessary to invert the inner product matrix S. Although Proposition 17 shows that S is close to the identity matrix for all but the lower right corner of the matrix, it does not guarantee a small condition number. However, using Mathematica we have computed S using extended precision and found that it is well conditioned (see Table 2.1).

Table 2.1: Condition number for S using $\epsilon = 10^{-7}$

Bandwidth c	**Number of nodes N**	**Condition number**
4π	21	1.48
8π	31	2.11
12π	40	2.06
16π	49	2.34
20π	57	2.89

We now establish that the approximate prolate spheroidal wave functions can be used as a basis for the space of bandlimited functions $\mathcal{E}_{c,\epsilon}$ defined in Definition 13. By defining the matrix $B = HQ^TW$ we see that from (2.25) we have that

$$\Psi_j(x) = \sum_{l=1}^{N} B_{jl} e^{ic\theta_l x}.$$

From Proposition 15 it follows that Q is orthogonal, and therefore

$$\det(B) = \det(H)\det(W) = \prod_{l=1}^{N} \frac{\sqrt{w_l}}{\eta_l}.$$

Since η_l and w_l are non-zero, it follows that B is invertible (although it may be ill-conditioned), and therefore the set of approximate prolate spheroidal wave functions spans the space $\mathcal{E}_{c,\epsilon}$ defined in Definition 13.

2.6.3 Interpolating functions

In many applications, it is convenient to work with function values of a function rather than with expansion coefficients with respect to a set of basis functions. This motivates the introduction of the interpolating basis. When a function is expanded into this basis, the expansion coefficients are the function values at some set of nodes, in our case quadrature nodes for bandlimited functions. Such bases are also useful in some multiwavelet applications when solving non-linear PDEs, see [1]. We define the interpolating basis functions for bandlimited functions on an interval as follows.

Definition 18 (Basis of interpolating functions) *Define the matrices Q, W, and H according to (2.21)-(2.23), and the matrix $R = WQHQ^TW$. The sequence of functions*

$$r_k(x) = \sum_{l=1}^{N} R_{kl} e^{ic\theta_l x} \tag{2.28}$$

for $k = 1, \dots, N$ is called a basis of interpolating functions for bandlimited functions on an interval.

Note that we can express the elements of the matrix R as

$$R_{kl} = \sum_{j=1}^{N} w_k \Psi_{\mathbf{j}}(\theta_k) \frac{1}{\eta_j} \Psi_{\mathbf{j}}(\theta_l) w_l.$$

Using this expression, equation (2.13), and the fact that Q is orthogonal, we conclude that the basis function $r_k(x)$ has the interpolating property $r_k(\theta_l) = \delta_{kl}$. In Figure 2.3 we plot two interpolating functions constructed for the bandwidth $c = 8.5\pi$ and accuracy $\epsilon = 10^{-7}$ resulting in 32 quadrature nodes.

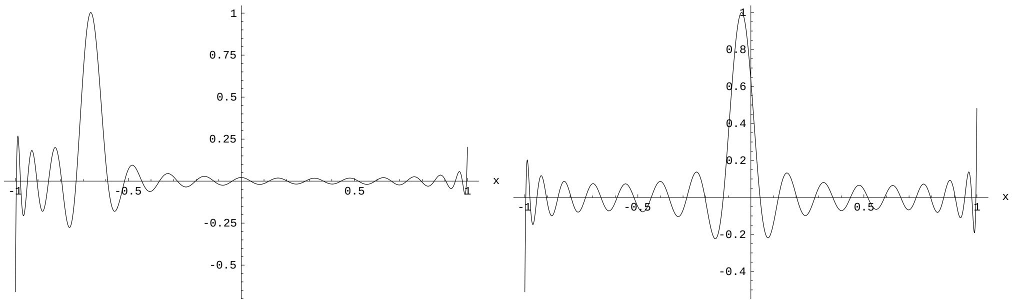


Figure 2.3: Two interpolating basis functions constructed for the bandwidth $c = 8.5\pi$ with the accuracy $\epsilon = 10^{-7}$ resulting in 32 linearly independent functions. In left figure we display the eighth interpolating function and in the right figure the 16:th.

Since Q is orthogonal we have that

$$\det(R) = \det(H)\det(W^2) = \prod_{l=1}^{N} \frac{w_l}{\eta_l}$$

and since η_l and w_l are non-zero it follows that R is invertible (although it may be ill-conditioned). Therefore the basis of interpolating functions spans the space $\mathcal{E}_{c,\epsilon}$ defined in Definition 13.

2.6.4 Transformation matrices

In the previous three sections we presented three examples of bases for bandlimited functions on an interval. All three of them are useful for different situations, and it is important to be able to transform between the different bases. In this section we list the transformation matrices between the different bases and give examples of the condition numbers for the different transforms. We will see that some of the transforms are ill-conditioned, but to transform between approximate prolate spheroidal wave functions and interpolating functions is a well-conditioned operation.

From Section 2.6.2 we know that the transformation matrix between expansion coefficients with respect to the exponential basis to expansion coefficients with respect to the basis of approximate prolate spheroidal wave functions is given by

$$B_c = (B^{-1})^T = H^{-1}Q^TW^{-1} \tag{2.29}$$

where the matrices H, Q, and W are defined in (2.21)-(2.23). From Section 2.6.3 we know that the transformation matrix between expansion coefficients with respect to the exponential basis to expansion coefficients with respect to the basis of interpolating functions is given by

$$R_c = (R^{-1})^T = W^{-1}QH^{-1}Q^TW^{-1}. \tag{2.30}$$

Using the fact that Q is orthogonal we see that the transformation matrix between the approximate prolate spheroidal wave functions and interpolating functions is given by

$$P_c = R_cB_c^{-1} = W^{-1}Q. \tag{2.31}$$

We summarize all transformation matrices in Figure 2.4 below. Note that by using (2.28) we have that the elements of R_c are given by

$$(R_c)_{kl} = e^{ic\theta_k\theta_l}.$$

In Tables 2.2 and 2.3 we have computed the condition numbers for the different transformation matrices for different bandwidths, using two different accuracies. We see that in all cases the condition number for transforming between approximate prolate spheroidal wave functions and interpolating functions is small while the other two cases yield large condition numbers.

Let us compare the bases of bandlimited functions to bases of polynomials. The space of polynomials of degree N is spanned by, e.g., the monomials $\{1, x, x^2, \ldots, x^N\}$ and the Legendre polynomials of degree $\leq N$. The basis of monomials is not close to being orthogonal and ill-conditioned. The Legendre polynomials, on the other hand, are orthonormal on $[-1, 1]$ with the unit weight function, and well-conditioned. We can therefore compare the basis of exponentials to the basis of monomials, while the basis of approximate prolate spheroidal wave functions (which are nearly orthonormal) plays a similar role as the Legendre polynomials.

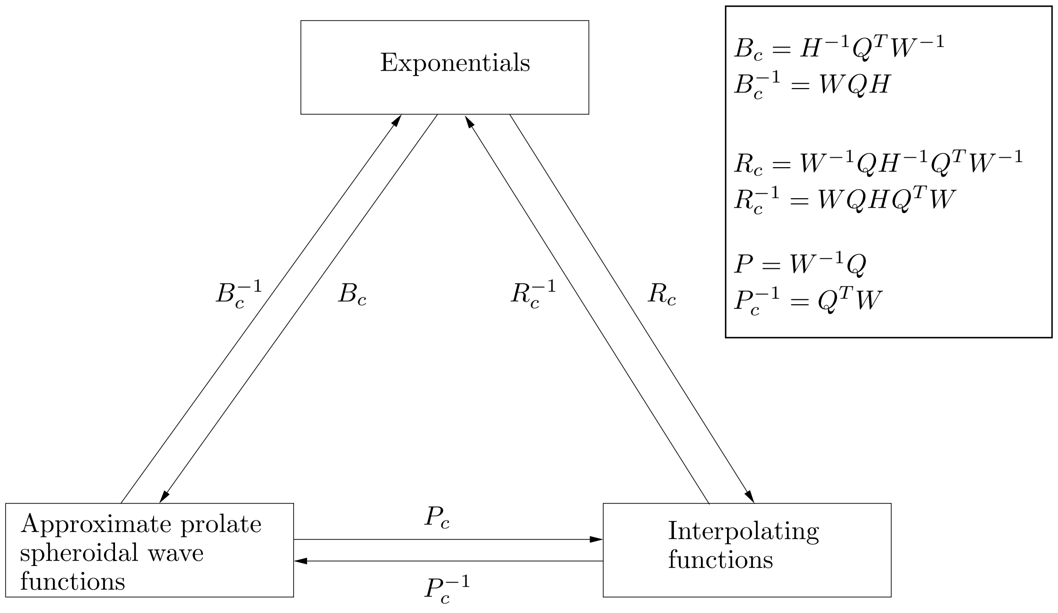


Figure 2.4: Transformation matrices between three different bases for bandlimited functions on an interval. The matrices H, Q, and W are defined in (2.21)-(2.23). Note that the matrices H and W are diagonal.

Table 2.2: Condition number for transformation matrices for the bandwidth $c = 8.5\pi$. The accuracy $\epsilon = 10^{-7}$ requires 32 nodes and the accuracy $\epsilon = 10^{-14}$ requires 41 nodes.

Transformation matrix	$\epsilon = 10^{-7}$	$\epsilon = 10^{-14}$
P_c	2.7	3.5
B_c	1.1×10^8	2.5×10^{14}
R_c	1.2×10^8	3.1×10^{14}

Table 2.3: Condition number for transformation matrices for the bandwidth $c = 17\pi$. The accuracy $\epsilon = 10^{-7}$ requires 51 nodes and the accuracy $\epsilon = 10^{-14}$ requires 62 nodes.

Transformation matrix	$\epsilon = 10^{-7}$	$\epsilon = 10^{-14}$
P_c	2.8	3.8
B_c	1.2×10^8	3.4×10^{14}
R_c	1.3×10^8	4.0×10^{14}

In some situations, it is useful to evaluate a function at an equally spaced grid (including the endpoints) given the function values at the quadrature nodes. Let us introduce the equally spaced grid

$$x_k = -1 + 2\frac{k-1}{N-1}$$

for $k = 1, \dots, N$, and the matrix V with elements defined by

$$V_{kl} = e^{icx_k\theta_l}.$$

Then the matrix $C = VR_c^{-1}$ maps $\{f(\theta_k)\}_{k=1}^N$ to $\{f(x_k)\}_{k=1}^N$.

2.7 Numerical results

In this section we provide numerical examples. We approximate trigonometric functions, Chebyshev polynomials, and Gaussian bell functions on an interval using bandlimited functions. We saw in Section 2.5 that we can approximate bandlimited functions on an interval using a finite number of exponentials with phases given by quadrature nodes for bandlimited functions. Theorem 12 provides us with an error estimate for such approximations. However, this error estimate is not useful in practice due to the factor $\sum_{k\in\mathbb{Z}} |a_k|$ in the error estimate. This factor is bounded, but depends on the function to be approximated. The factor is related to an upper bound of the norm of the bandlimited function to be approximated, but it depends on its norm over $\mathbb{R}$, and not on the norm over $[-1, 1]$ which is the relevant norm for many applications. Nevertheless, in this section we present strong numerical evidence that the approximation indeed is accurate within the precision ϵ selected for the quadratures.

We saw in Section 2.6 how we can use different bases to represent bandlimited functions on an interval. In this section we will illustrate the importance of choosing the right basis for numerical computations. It turns out that the approximate prolate spheroidal wave functions give superior accuracy compared to using exponentials as the basis. Later in this section we provide a heuristic explanation for this behavior.

For our experiments, we sample the function to be approximated at quadrature nodes. We then find the expansion coefficients α_k with respect to the basis of exponentials, and the expansion coefficients β_k with respect to the basis of approximate prolate spheroidal wave functions, using the inverse of the transformation matrices in (2.30) and (2.31), respectively. We finally evaluate the functions at equally spaced points (including the endpoints) on the interval $[-1, 1]$ using the expansions

$$f(x) \simeq \sum_{k=1}^{N} \alpha_k e^{ic\theta_k x} \tag{2.32}$$

and

$$f(x) \simeq \sum_{k=1}^{N} \beta_k \Psi_k(x) \tag{2.33}$$

where $\Psi_k(x)$ is defined in (2.14). Although both function approximations can be shown to be identical, numerical ill-conditioning makes the first expansion considerable less accurate as we will observe in the experiments. A heuristic explanation for this is given in Section 2.7.1.

2.7.1 Approximations of trigonometric functions by bandlimited functions on an interval

For the first example, we construct 32 quadrature nodes and weights for the four bandwidths, $c = 5.5\pi$, 7π, 8.5π, and 10.5π. In order to obtain these bandwidths using 32 nodes, we set the accuracy ϵ to 10^{-13}, 10^{-10}, 10^{-7}, and 10^{-4}, respectively. We approximate the function e^{ibx} for $|b| \le c$ by using the expansion (2.32) in Figure 2.5, and using the expansion (2.33) in Figure 2.6. Note how the approximation in the first case (Figure 2.5) is better for higher bandwidths than for lower bandwidths. In the second case (Figure 2.6) where we use

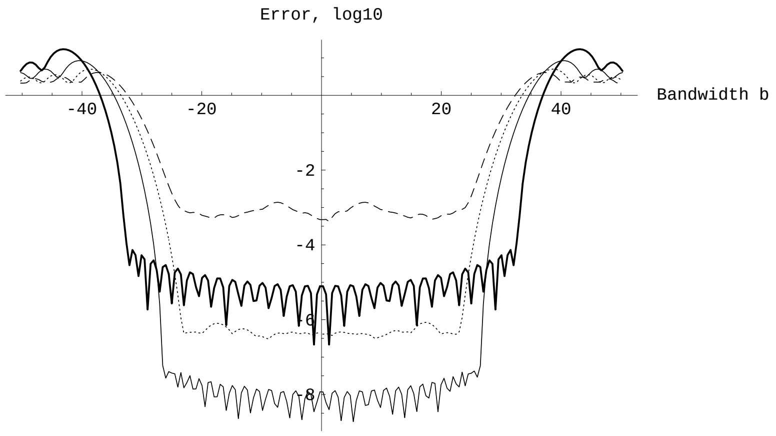


Figure 2.5: Absolute error ($\log_{10}$) for approximating the function e^{ibx} in the interval $[-1, 1]$ with $|b| \le 16\pi$ using exponentials as basis. The approximating functions are constructed using 32 basis functions with maximum bandwidth $c = 10.5\pi$ (thick solid curve), $c = 8.5\pi$ (thin solid curve), $c = 7\pi$ (dotted curve), and $c = 5.5\pi$ (dashed curve). The error of each approximation is measured at 32 equally spaced points (including the end points) on the interval $[-1, 1]$.

transformation matrix P_c^{-1}, the lower bandwidth gives the higher accuracy as expected.

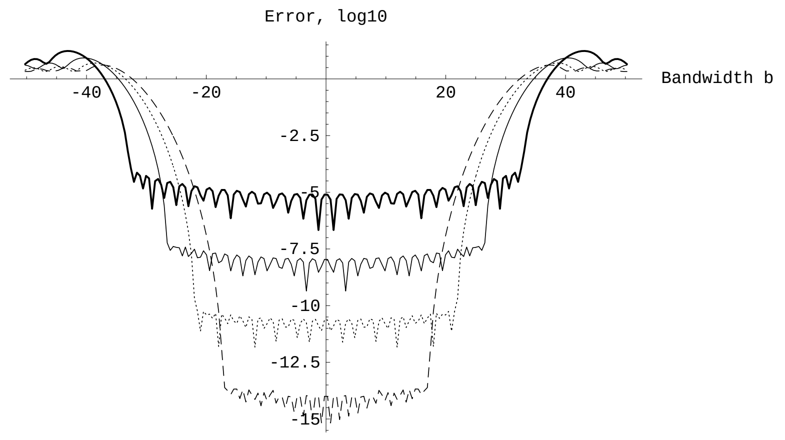


Figure 2.6: Absolute error ($\log_{10}$) for approximating the function e^{ibx} in the interval $[-1, 1]$ with $|b| \leq 16\pi$ using approximate prolate spheroidal wave function as basis. The approximating functions are constructed using 32 basis functions with maximum bandwidth $c = 10.5\pi$ (thick solid curve), $c = 8.5\pi$ (thin solid curve), $c = 7\pi$ (dotted curve), and $c = 5.5\pi$ (dashed curve). The error of each approximation is measured at 32 equally spaced points (including the end points) on the interval $[-1, 1]$.

We now give a heuristic argument why the expansion (2.33) is better suited for numerical computations than (2.32). Let us first study the expansion (2.32). We have that

$$\sum_{k=1}^{N} \alpha_k e^{ic\theta_k x} = \sum_{k=1}^{N} \left(\sum_{l=1}^{N} (R_c^{-1})_{kl} f(\theta_l) \right) e^{ic\theta_k x} \tag{2.34}$$

where

$$(R_c^{-1})_{kl} = (WQHQ^T W)_{kl} = \sqrt{w_k} \left(\sum_{m=1}^{N} Q_{km} \frac{1}{\eta_m} Q_{lm} \right) \sqrt{w_l}. \tag{2.35}$$

As pointed out in Section 2.6.2, $\eta_1 \sim \sqrt{\frac{2\pi}{c}}$ and $\eta_N \sim \epsilon$. We see that the terms in the summation in (2.35) have strongly varying magnitudes due to the factor $1/\eta_m$. This summation is therefore ill-conditioned for small ϵ. Hence, the computation of expansion coefficients α_k is unstable, potentially introducing large errors in (2.34).

Let us now see what happens when using the expansion (2.33). We have that

$$f(x) = \sum_{k=1}^{N} \beta_k \Psi_k(x) = \sum_{k=1}^{N} \left(\sum_{l=1}^{N} (P_c^{-1})_{kl} f(\theta_l) \right) \left(\sum_{n=1}^{N} (HQ^T W)_{kn} e^{ic\theta_n x} \right)$$

where

$$(P_c^{-1})_{kl} = (Q^T W)_{kl}.$$

This transformation matrix does not contain the greatly varying factor $\frac{1}{\eta_j}$ that caused problems when computing expansion coefficients with respect to exponentials. However, this factor does appear in the expansion of $\Psi_k(x)$ as a linear combination of exponentials, namely

$$\Psi_k(x) = \sum_{n=1}^{N} (HQ^T W)_{kn} e^{ic\theta_n x}. \tag{2.36}$$

Since

$$(HQ^T W)_{kn} = \frac{1}{\eta_k} Q_{nk} \sqrt{w_n}$$

we see that the summation in (2.36) is multiplied by a factor $\frac{1}{\eta_k}$. For large k, the denominator $\eta_k \sim \epsilon$, and hence the computation of Ψ_k for large k may contain large errors. On the other hand, the computation of the expansion coefficients β_k is stable and for functions of the type e^{ibx}, equation (2.8) shows that the expansion coefficients $\beta_k \sim \eta_k$. Therefore the expansion coefficients with respect to Ψ_k for large k are small. In other words, from (2.8) we have small expansion coefficients with respect to the numerically unstable basis functions.

In Figure 2.7 we expand the function e^{ibx} for a range of bandwidths using 64 approximate prolate spheroidal wave functions as basis functions. We construct the 64 nodes using the four bandwidths $c = 18.5\pi$, 20.5π, 23π, and 26π. In order to obtain these bandwidths using 64 nodes, we set the accuracy ϵ to 10^{-13}, 10^{-10}, 10^{-7}, and 10^{-4}, respectively.

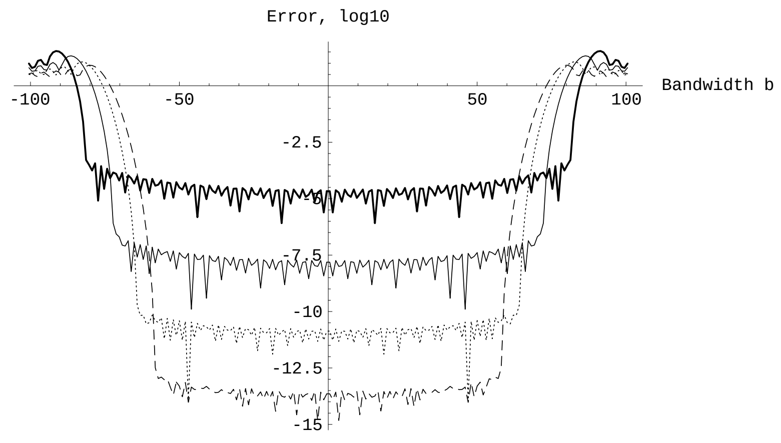


Figure 2.7: Absolute error ($\log_{10}$) for approximating the function e^{ibx} in the interval $[-1, 1]$ with $|b| \leq 32\pi$ using approximate prolate spheroidal wave function as basis. The approximating functions are constructed using 64 basis functions with maximum bandwidth $c = 26\pi$ (thick solid curve), $c = 23\pi$ (thin solid curve), $c = 20.5\pi$ (dotted curve), and $c = 18.5\pi$ (dashed curve). The error of each approximation is measured at 64 equally spaced points (including the end points) on the interval $[-1, 1]$.

2.7.2 Approximations of polynomials by bandlimited functions on an interval

In this section we study approximations of polynomials using bandlimited functions . We consider the Chebyshev polynomials on the interval $[-1, 1]$ and approximate them using 32 and 64 approximate prolate spheroidal wave functions as basis functions (Figure 2.8 and Figure 2.9). The approximate prolate spheroidal wave functions are constructed using the same bandwidths and accuracies as in the previous section.

We note that for a fixed number of nodes, the results improves when decreasing the bandwidth c and increasing the accuracy ϵ used for constructing the quadrature nodes. For the case with $c = 8.5\pi$ and $\epsilon = 10^{-7}$ (32 nodes) we get approximately 6 digits of accuracy for Chebyshev polynomials of degree 8 and lower. For the case with $c = 5.5\pi$ and $\epsilon = 10^{-13}$ (32 nodes) we get approximately 6 digits of accuracy for Chebyshev polynomials of degree 15 and lower.

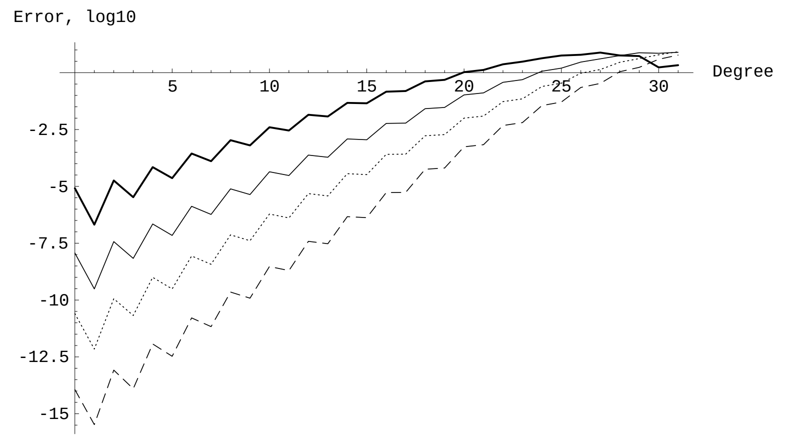


Figure 2.8: Absolute error ($\log_{10}$) for approximating the Chebyshev polynomial $T_k(x)$ in the interval $[-1, 1]$ with $k = 0, \ldots, 31$ using approximate prolate spheroidal wave function as basis. The approximating functions are constructed using 32 basis functions with maximum bandwidth $c = 10.5\pi$ (thick solid curve), $c = 8.5\pi$ (thin solid curve), $c = 7\pi$ (dotted curve), and $c = 5.5\pi$ (dashed curve). The error of each approximation is measured at 32 equally spaced points (including the end points) on the interval $[-1, 1]$.

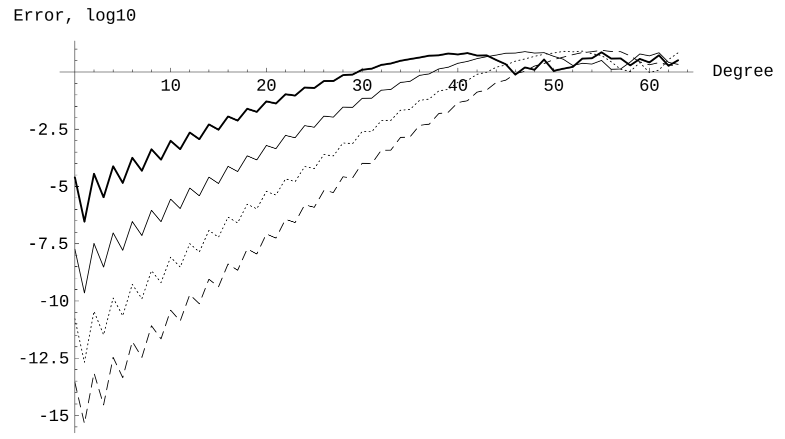


Figure 2.9: Absolute error ($\log_{10}$) for approximating the Chebyshev polynomial $T_k(x)$ in the interval $[-1, 1]$ with $k = 0, \ldots, 63$ using approximate prolate spheroidal wave function as basis. The approximating functions are constructed using 64 basis functions with maximum bandwidth $c = 26\pi$ (thick solid curve), $c = 23\pi$ (thin solid curve), $c = 20.5\pi$ (dotted curve), and $c = 18.5\pi$ (dashed curve). The error of each approximation is measured at 64 equally spaced points (including the end points) on the interval $[-1, 1]$.

2.7.3 Approximations of Gaussians by bandlimited functions on an interval

We conclude this chapter by approximating Gaussian bell functions with bandlimited functions on an interval. This is an example where the function is "almost" bandlimited. The support of the Fourier transform of these functions is not compact, but decays exponentially fast to zero.

Let us consider functions of type

$$f(x) = e^{-\frac{x^2}{\sigma^2}}$$

on the interval $[-1, 1]$ for variances $\sigma^2 \in [0.0001, 10]$ (see Figure 2.10). In Figure 2.11 and Fig-

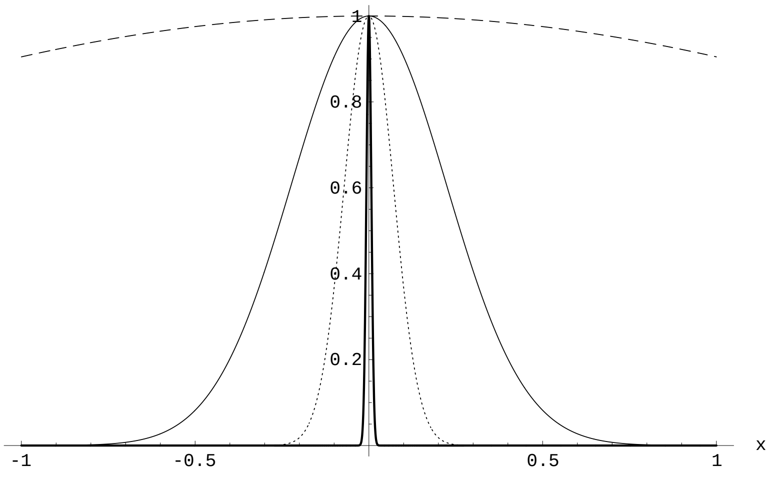


Figure 2.10: Plot of the Gaussian bell functions $f(x) = e^{-\frac{x^2}{\sigma^2}}$ for the variances $\sigma^2 = 0.0001$ (thick solid line), $\sigma^2 = 0.01$ (dotted line), $\sigma^2 = 0.1$ (thin solid line), and $\sigma^2 = 10$ (dashed line).

ure 2.12 we show the error using 32 and 64 approximate prolate spheroidal wave functions as basis functions, respectively, to represent the Gaussian bells. For each case, we demonstrate the accuracy using four different bandwidths. The approximate prolate spheroidal wave functions are constructed using the same bandwidths and accuracies as in Section 2.7.1.

We note that when using 32 nodes and using quadrature nodes constructed for $c = 5.5\pi$ and $\epsilon = 10^{-13}$, we get approximately 6 digits of accuracy for $\sigma^2 \geq 0.05$. For 64 nodes using quadrature nodes constructed for $c = 18.5\pi$ and $\epsilon = 10^{-13}$, we get approximately 6 digits of accuracy for $\sigma^2 \geq 0.01$.

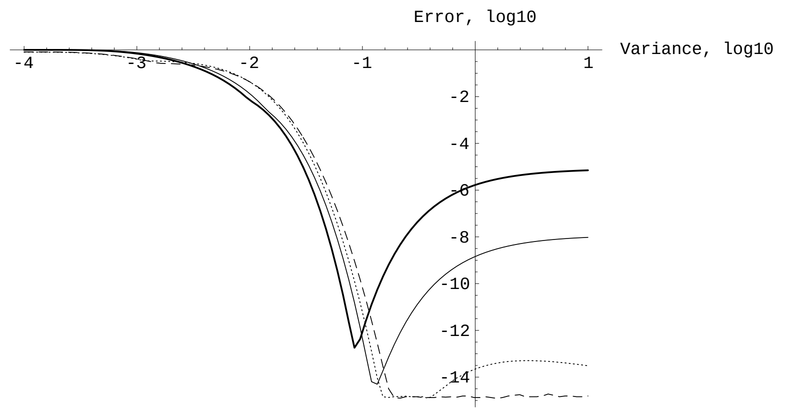


Figure 2.11: Absolute error ($\log_{10}$) for approximating the function $e^{-\frac{x^2}{\sigma^2}}$ in the interval $[-1,1]$ with variance $\sigma^2 \in [0.0001, 10]$ using approximate prolate spheroidal wave function as basis. The approximating functions are constructed using 32 basis functions with maximum bandwidth $c = 10.5\pi$ (thick solid curve), $c = 8.5\pi$ (thin solid curve), $c = 7\pi$ (dotted curve), and $c = 5.5\pi$ (dashed curve). The error of each approximation is measured at 33 equally spaced points (including the end points and $x = 0$) on the interval $[-1,1]$.

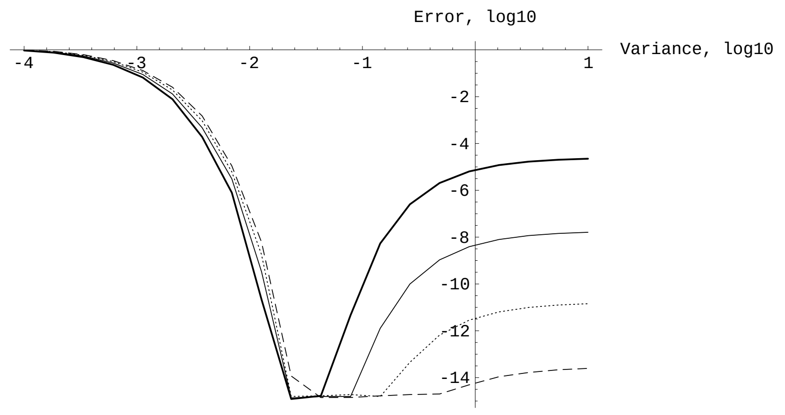


Figure 2.12: Absolute error ($\log_{10}$) for approximating the function $e^{-\frac{x^2}{\sigma^2}}$ in the interval $[-1,1]$ with variance $\sigma^2 \in [0.0001, 10]$ using approximate prolate spheroidal wave function as basis. The approximating functions are constructed using 64 basis functions with maximum bandwidth $c = 26\pi$ (thick solid curve), $c = 23\pi$ (thin solid curve), $c = 20.5\pi$ (dotted curve), and $c = 18.5\pi$ (dashed curve). The error of each approximation is measured at 65 equally spaced points (including the end points and $x = 0$) on the interval $[-1,1]$.

Chapter 3

Derivative matrices with boundary and interface conditions

When solving ordinary and partial differential equations using spectral methods, we expand the solution into a set of basis functions which we can differentiate exactly. The trigonometric functions or Chebyshev polynomials are two common choices of such basis functions. In this thesis, we use the interpolating basis for bandlimited functions which also can be differentiated exactly. We solve time-dependent PDEs by discretizing the spatial operator and then compute the exponential of the resulting matrix. This approach requires the boundary conditions to be incorporated into the spatial operator.

This thesis will also study numerical solutions to wave propagation problems over domains with piecewise smooth coefficients. We decompose the domain into a collection of subdomains such that the coefficients are smooth on each subdomain. Differentiation will be well-defined within such subdomains, but across the interfaces the solution is usually not differentiable. The solution satisfies some interface condition, e.g., continuity. In using the exponential of the spatial operator, we need a way to maintain the interface condition by enforcing the condition in constructing the spatial operator.

In this chapter we show how to incorporate the boundary and interface conditions into the spatial operator for a general set of (smooth) basis functions. In the next chapter we provide an example of using this approach for a basis of bandlimited functions introduced in Chapter 2. We incorporate the boundary and interface conditions by using integration by parts as it was done in Alpert et al. [1].

We begin by outlining the general technique for deriving derivative matrices on an interval with respect to a general (finite) set of smooth basis functions. We then extend the technique to piecewise smooth functions defined on a collection of subintervals. Numerical examples are provided in Chapter 4. Let us begin with the following

Definition 19 *Let* $\{\phi_i(x)\}_{i=1}^{N} \subseteq C^{\infty}([-1,1])$ *be a set of* N *linearly independent functions.*

Define $\mathcal{G}$ as the linear span of $\{\phi_i(x)\}_{i=1}^N$ equipped with the inner product

$$(u,v) = \int_{-1}^{1} u(x)\overline{v(x)}\ dx.$$

Define S, K, E, F, and G as the N-by-N matrices with elements given by

$$S_{kl} = (\phi_l(x), \phi_k(x)),$$

$$K_{kl} = \left(\phi_l(x), \frac{d\phi_k(x)}{dx}\right),$$

$$E_{kl} = \overline{\phi_k(-1)}\phi_l(-1),$$

$$F_{kl} = \overline{\phi_k(1)}\phi_l(1),$$

and

$$G_{kl} = \overline{\phi_k(1)}\phi_l(-1).$$

We note that the matrices E, F, and G are of rank one. Furthermore,

$$K_{kl} = \int_{-1}^{1} \phi_l(x)\overline{\frac{d}{dx}\phi_k(x)}\ dx = \phi_l(1)\overline{\phi_k(1)} - \phi_l(-1)\overline{\phi_k(-1)} - \int_{-1}^{1}\overline{\phi_k(x)}\frac{d}{dx}\phi_l(x)\ dx$$

$$= F_{kl} - E_{kl} - \overline{K_{lk}}$$

and hence

$$K = F - E - K^*. \tag{3.1}$$

We note that if $F = E$, then the matrix K is anti-symmetric.

3.1 Derivative matrices on an interval

In this section we derive derivative matrices defined on the interval $[-1,1]$. In the following section we generalize this technique to subdivided intervals. Let $u(x)$ be a test function such that $u, u_x \in \mathcal{G}$. Then

$$u(x) = \sum_{l=1}^{N} s_l\phi_l(x)$$

for some set of coefficients s_l. We seek coefficients $\tilde{s}_l$ such that

$$\frac{du}{dx} = \sum_{l=1}^{N} \tilde{s}_l \phi_l(x). \tag{3.2}$$

Computing the inner product with ϕ_k of both sides of (3.2) yields

$$\begin{aligned} \int_{-1}^{1} \frac{du}{dx}\overline{\phi_k(x)}\, dx &= \sum_{l=1}^{N} \tilde{s}_l \left(\int_{-1}^{1} \phi_l(x)\overline{\phi_k(x)}\, dx \right) \\ &= \sum_{l=1}^{N} S_{kl}\tilde{s}_l. \end{aligned} \tag{3.3}$$

Integrating the left hand side of (3.3) by parts, we have

$$\int_{-1}^{1} \frac{du}{dx}\overline{\phi_k(x)}\, dx = \Big[u(x)\overline{\phi_k(x)} \Big]_{-1}^{1} - \sum_{l=1}^{N} K_{kl}s_l. \tag{3.4}$$

Combining (3.3) and (3.4), we obtain

$$\sum_{l=1}^{N} S_{kl}\tilde{s}_l = u(1)\overline{\phi_k(1)} - u(-1)\overline{\phi_k(-1)} - \sum_{l=1}^{N} K_{kl}s_l. \tag{3.5}$$

Our next step is to express $u(\pm 1)$ via the coefficients s_l. Let us consider the case where we do not impose any boundary condition. Then using $u(\pm 1) = \sum_{l=1}^{N} s_l \phi_l(\pm 1)$, and inserting it into (3.5) gives us

$$\begin{aligned} \sum_{l=1}^{N} S_{kl}\tilde{s}_l &= \sum_{l=1}^{N} s_l\phi_l(1)\overline{\phi_k(1)} - \sum_{l=1}^{N} s_l\phi_l(-1)\overline{\phi_k(-1)} - \sum_{l=1}^{N} K_{kl}s_l \\ &= \sum_{l=1}^{N} F_{kl}s_l - \sum_{l=1}^{N} E_{kl}s_l - \sum_{l=1}^{N} K_{kl}s_l. \end{aligned} \tag{3.6}$$

By introducing the notation $\mathbf{s} = [s_1, s_2, \dots, s_N]^T$ and $\tilde{\mathbf{s}} = [\tilde{s}_1, \tilde{s}_2, \dots, \tilde{s}_N]^T$, we can write the equation for the derivative matrix as

$$S\tilde{\mathbf{s}} = (F - E - K)\mathbf{s}.$$

Since the basis functions form a linearly independent set, the matrix S is invertible and the derivative matrix D is given by

$$D = S^{-1}(F - E - K). \tag{3.7}$$

Using (3.1) we can write D as

$$D = S^{-1}K^{*}. \tag{3.8}$$

So far we have not made any assumption about the boundary values of the function $u(x)$ and the derivative matrix is applicable for arbitrary boundary values. Since the function $u(x)$ is assumed to be differentiable, results using (3.8) will coincide with the classical derivative except for numerical errors introduced in computing the matrices S^{-1}, F, E, and K.

We next consider the case where we construct the differentiation matrix for functions satisfying boundary conditions. If $u(-1) = 0$ or $u(1) = 0$, then the matrices E or F in (3.7) can be set to zero. Let us define the derivative matrices D_0^+, D_0^-, and D_0 as

$$D_0^+ = S^{-1}(F - K), \tag{3.9}$$

$$D_0^- = S^{-1}(-E - K), \tag{3.10}$$

and

$$D_0 = -S^{-1}K. \tag{3.11}$$

These matrices correspond to the boundary condition $u(-1) = 0$, $u(1) = 0$, and $u(\pm 1) = 0$, respectively.

Finally, we consider the case with the periodic boundary conditions. If $u(-1) = u(1)$, we can get two possible derivative matrices from (3.5),

$$D_1 = S^{-1}(G - E - K)$$

and

$$D_2 = S^{-1}(F - G^{*} - K).$$

Let us define D_p as

$$D_{per} = \frac{D_1 + D_2}{2}.$$

Using (3.1) we find that

$$D_p = \frac{S^{-1}(G - G^{*} - K + K^{*})}{2}. \tag{3.12}$$

We note that the factor $G - G^* - K + K^*$ is anti-symmetric and has a pure imaginary spectrum. Therefore, the derivative matrix D_p is a better choice for a derivative matrix than the matrices D_1 and D_2.

Using (3.8)-(3.12), we summarize our results as

Definition 20 *Let E, F, G, S, and K be the matrices in Definition 19. We define the derivative matrices D, D_0^+, D_0^-, D_0, and D_{per} as*

$$D = S^{-1}K^*,$$

$$D_0^+ = S^{-1}(F - K),$$

$$D_0^- = S^{-1}(-E - K),$$

$$D_0 = -S^{-1}K,$$

and

$$D_{per} = \frac{S^{-1}(G - G^* - K + K^*)}{2},$$

respectively.

3.1.1 Example

Let $\mathcal{B} = \text{span}\{\frac{1}{\sqrt{2}}e^{i\pi(k-1)x}\}_{k=1}^{N}$. Then

$$S_{kl} = \delta_{kl},$$

$$K_{kl} = -i\pi\frac{k-1}{2}\delta_{kl},$$

$$E_{kl} = \frac{\overline{e^{-i\pi(k-1)}}e^{-i\pi(l-1)}}{2} = \frac{e^{i\pi(k-l)}}{2},$$

and

$$F_{kl} = \frac{\overline{e^{i\pi(k-1)}}e^{i\pi(l-1)}}{2} = \frac{e^{i\pi(l-k)}}{2} = \overline{E_{kl}}.$$

Therefore,

$$(F - E)_{kl} = \overline{E_{kl}} - E_{kl} = \frac{e^{-i\pi(k-l)} - e^{i\pi(k-l)}}{2} = -i\sin\pi(k-l) = 0\ .$$

In this special case it makes no difference if we restrict our operator to functions vanishing at the endpoints due to the special structure of the basis functions. This is not true in general.

3.1.2 Properties

Let us investigate the properties of the differentiation matrices derived above. Analytically, we expect the eigenfunctions of $\frac{d}{dx}$ with no boundary conditions to be multiples of $v(x) = e^{\lambda x}$, where λ is a complex number. The location of λ in the complex plane determines the nature of our eigenfunctions. For example, by considering eigenfunctions with pure imaginary eigenvalues we get oscillatory eigenfunctions (trigonometric functions). We also notice that the eigenfunctions are smooth.

The case where the differentiation operator is restricted to functions vanishing at the boundaries is more subtle. Clearly, there are no nontrivial solutions of the eigenvalue problem

$$\begin{aligned} &\frac{dv}{dx} = \lambda v(x) \\ &v(-1) = v(1) = 0, \end{aligned} \tag{3.13}$$

and thus no eigenfunctions satisfying the boundary conditions. Nevertheless, the differentiation matrix $-S^{-1}K$ derived above is well-defined (if S is invertible) and has a set of eigenvectors. Yet it is impossible to relate its behavior to the eigenvalue problem (3.13) for the first derivative. To connect the derivative matrix $-S^{-1}K$ to the analytic differentiation, we need to consider the singular value problem.

Let us show that the singular vectors are well-defined for both the analytical and discretized differentiation operator with zero boundary conditions. Consider the set of equations

$$\begin{cases} \frac{du}{dx} = \lambda v \\ \frac{d^*}{dx}(v) = \lambda u \\ v(-1) = v(1) = 0 \end{cases}, \tag{3.14}$$

where due to the boundary conditions, the adjoint differentiation operator is given by $\frac{d^*}{dx} = -\frac{d}{dx}$. Normalized solutions u and v of the first two equations in (3.14) are the left and right singular vectors, of the differentiation operator $\frac{d}{dx}$. The corresponding singular values σ are given by $\sigma = |\lambda|$. It is easy to verify that the functions

$$v(x) = -\sin(\lambda(x + a))$$

and

$$u(x) = \cos(\lambda(x + a))$$

satisfy the first two equations in (3.14). The boundary conditions imply that

$$\begin{cases} \lambda(a - 1) = k\pi \\ \lambda(a + 1) = l\pi \end{cases}, \quad k, l \in \mathbb{Z}.$$

If $a = \pm 1$, then we have that $\lambda_k = k\pi/2$ for $k \in \mathbb{Z}$. If $a \neq \pm 1$ and $k, l \neq 0$, then a straightforward calculation shows that

$$a = \frac{k+l}{k-l}$$

and

$$\lambda = \frac{k(k-l)\pi}{2l}.$$

If $a \neq \pm 1$ and $k = 0$ or $l = 0$ then $a \in \mathbb{C}$ and $\lambda = 0$.

3.2 Derivative matrix over a subdivided interval

We extend our approach from the previous section to the construction of derivative matrices defined over multiple intervals joined together. The construction is similar to the one used for multiwavelets in [1]. We represent functions on such domains by using N smooth basis functions on each interval. These subinterval representations are independent of each other and in our construction the derivative operator couples them together.

We begin by introducing the following notation. Let M be a positive integer and let I denote the interval $[-1, -1+2M]$. Define the subintervals

$$I_l = [\bar{x}_l, \bar{x}_{l+1}] = [-3+2l, -1+2l],$$

where $l = 1, \ldots, M$, and $I^o = I \setminus \{x_l\}_{l=2}^{M}$. Let $\{\phi_k(x)\}_{k=1}^{N}$ be a basis of $\mathcal{G}$ defined in Definition 19 and define $\{\phi_{kl}(x)\}_{k,l}$ for $k = 1, \ldots, N$ and $l = 1, \ldots, M$ by

$$\phi_{kl}(x) = \left\{ \begin{array}{ll} \phi_k(x - 2l + 2) & x \in I_l \\ 0 & x \notin I_l \end{array} \right. .$$

Let $\mathcal{G}_M$ denote the linear span of $\{\phi_{kl}(x)\}_{k,l}$ equipped with the inner product

$$(u, v) = \int_{-1}^{-1+2M} u(x)\overline{v(x)}\, dx.$$

We will refer to $\{x_l\}_{l=2}^{M}$ as the interfaces. Note that $\phi_{kl}(\bar{x}_l) = \phi_k(-1)$ and $\phi_{kl}(\bar{x}_{l+1}) = \phi_k(1)$. It follows that

$$\int_{\bar{x}_l}^{\bar{x}_{l+1}} \phi_{jl}(x)\overline{\phi_{il}(x)}\, dx = S_{ij}$$

and

$$\int_{\bar{x}_l}^{\bar{x}_{l+1}} \phi_{jl}(x)\overline{\frac{d\phi_{il}(x)}{dx}}\, dx = K_{ij}$$

where S_{ij} and K_{ij} are given in Definition 19.

Let the function $f(x)\in\mathcal{G}_M$ be written as

$$f(x) = \sum_{l=1}^{M}\sum_{i=1}^{N} s_{il}\phi_{il}(x) \tag{3.15}$$

for some set of coefficients s_{il}. We note that for each interior interface $\bar{x}_l$, there are two possible expansions available, namely from the left

$$f(\bar{x}_l) = \sum_{i=1}^{N} s_{i,l-1}\phi_i(1) \tag{3.16}$$

and from the right

$$f(\bar{x}_l) = \sum_{i=1}^{N} s_{il}\phi_i(-1)\ . \tag{3.17}$$

These two expansions correspond to taking the limit from the respective direction. For the first node, $\bar{x}_1$, only (3.17) applies and for the last node, $\bar{x}_{M+1}$, only (3.16) applies. Unless we impose continuity of f on the entire interval I, the representation (3.15) does not by itself uniquely define f at the interfaces.

Let $f\in\mathcal{G}_M$ and define the derivative $\frac{df}{dx}\in\mathcal{G}_M$ formally as a function on the form

$$\frac{df}{dx} = \sum_{l=1}^{M}\sum_{i=1}^{N} \tilde{s}_{il}\phi_{il}(x) \tag{3.18}$$

such that the coefficients $\tilde{s}_{il}$ are determined by

$$\int_{\bar{x}_l}^{\bar{x}_{l+1}} \frac{df}{dx}\overline{\phi_{il}(x)}\, dx = f(\bar{x}_{l+1})\overline{\phi_{il}(\bar{x}_{l+1})} - f(\bar{x}_l)\overline{\phi_{il}(\bar{x}_l)} - \int_{\bar{x}_l}^{\bar{x}_{l+1}} f(x)\overline{\frac{d\phi_{il}}{dx}}\, dx \tag{3.19}$$

for each interval $l = 1,\dots,M$. In view of (3.16) and (3.17), the function f is not uniquely determined at the interfaces unless we impose continuity of f. Hence, the derivative defined by (3.18) and (3.19) is not uniquely defined without imposing additional conditions.

From (3.19) and (3.15) we get

$$\begin{aligned}\int_{\bar{x}_l}^{\bar{x}_{l+1}} \frac{df}{dx}\overline{\phi_{il}(x)}\,dx &= f(\bar{x}_{l+1})\overline{\phi_i(1)} - f(\bar{x}_l)\overline{\phi_i(-1)} \\ &\quad - \sum_{l'=0}^{M}\sum_{j=1}^{N} s_{jl'}\int_{\bar{x}_l}^{\bar{x}_{l+1}} \phi_{jl'}\frac{d\overline{\phi_{il}}}{dx}\,dx \\ &= f(\bar{x}_{l+1})\overline{\phi_i(1)} - f(\bar{x}_l)\overline{\phi_i(-1)} \\ &\quad - \sum_{j=1}^{N} s_{jl}\int_{\bar{x}_l}^{\bar{x}_{l+1}} \phi_{jl}\frac{d\overline{\phi_{il}}}{dx}\,dx \\ &= f(\bar{x}_{l+1})\overline{\phi_i(1)} - f(\bar{x}_l)\overline{\phi_i(-1)} - \sum_{j=1}^{N} K_{ij}s_{jl}\end{aligned} \tag{3.20}$$

for each interval $l = 1, \ldots, M$. Since $\frac{df}{dx}$ is of the form in (3.18) we have that

$$\begin{aligned}\int_{\bar{x}_l}^{\bar{x}_{l+1}} \frac{df}{dx}\overline{\phi_{il}(x)}\,dx &= \sum_{j=1}^{N} \tilde{s}_{jl}\int_{\bar{x}_l}^{\bar{x}_{l+1}} \phi_{jl}(x)\overline{\phi_{il}(x)}\,dx \\ &= \sum_{j=1}^{N} S_{ij}\tilde{s}_{jl}.\end{aligned} \tag{3.21}$$

Combining (3.20) and (3.21) gives us

$$\sum_{j=1}^{N} S_{ij}\tilde{s}_{jl} = f(\bar{x}_{l+1})\overline{\phi_i(1)} - f(\bar{x}_l)\overline{\phi_i(-1)} - \sum_{j=1}^{N} K_{ij}s_{jl}. \tag{3.22}$$

Since the basis functions are linearly independent, S is invertible. Therefore, if f is continuous, the expression (3.22) uniquely determines the expansion coefficients $\tilde{s}_{jl}$. There are two possible expansions for each interface,

$$\frac{df(\bar{x}_l)}{dx} = \sum_{i=1}^{N} \tilde{s}_{i,l-1}\phi_i(1) \tag{3.23}$$

and

$$\frac{df(\bar{x}_l)}{dx} = \sum_{i=1}^{N} \tilde{s}_{il}\phi_i(-1). \tag{3.24}$$

Unless f is differentiable, the derivative $\frac{df}{dx}$ is not well-defined at the interface points. If we impose that $f \in C^1(I) \cap \mathcal{G}_M$, then $\frac{df}{dx}$ is well defined and coincides with the classical derivative.

In the next three sections we impose boundary and interface conditions. It is then convenient to introduce the following notation. For each interval let us define $\mathbf{s_l} = [s_{1l}, s_{2l}, \dots, s_{Nl}]^T$ and $\tilde{\mathbf{s}}_\mathbf{l} = [\tilde{s}_{1l}, \tilde{s}_{2l}, \dots, \tilde{s}_{Nl}]^T$ for $l = 1, \dots, M$ where the coefficients s_{il} and $\tilde{s}_{il}$ are the expansion coefficients in (3.15) and (3.18) respectively.

3.2.1 Conditions for the end intervals

In this section we consider the end intervals where boundary conditions may be imposed. Let us first consider arbitrary boundary conditions at the left end point. We have that

$$f(-1) = \sum_{i=1}^{N} s_{i0}\phi_i(-1).$$

When describing $f(1)$, the right end point of the first interval, we can choose to take the limit via (3.17) or (3.16). We consider a weighted contribution from these two expansions by introducing the parameter $a \in [0, 1]$,

$$f(1) = \sum_{i=1}^{N} (1-a)s_{i1}\phi_i(1) + as_{i2}\phi_i(-1).$$

Inserting the expansions for $f(\pm 1)$ into (3.22), we obtain

$$\sum_{j=1}^{N} S_{ij}\tilde{s}_{j1} = \left(\sum_{j=1}^{N}(1-a)s_{j1}\phi_j(1) + as_{j2}\phi_j(-1)\right)\overline{\phi_i(1)} - \left(\sum_{j=1}^{N} s_{j1}\phi_j(-1)\right)\overline{\phi_i(-1)}$$
$$- \sum_{j=1}^{N} K_{ij}s_{j1}$$

or, equivalently using matrix-vector notation,

$$S\tilde{\mathbf{s}}_\mathbf{1} = \left((1-a)F - E - K\right)\mathbf{s}_1 + aG\mathbf{s}_2 \tag{3.25}$$

where we used the matrices defined in Definition 19. If $f(-1) = 0$, then (3.25) reduces to

$$S\tilde{\mathbf{s}}_\mathbf{1} = \left((1-a)F - K\right)\mathbf{s}_1 + aG\mathbf{s}_2. \tag{3.26}$$

Let us repeat the same argument for the last interval, I_M. We first consider arbitrary boundary conditions at the right end point where

$$f(\bar{x}_{M+1}) = \sum_{i=1}^{N} s_{iM}\phi_i(1).$$

In describing $f(\bar{x}_M)$, the left end point of the last interval, we can choose between (3.17) and (3.16). We will consider a weighted contribution from these two expansions by introducing the parameter $b \in [0,1]$, by

$$f(\bar{x}_M) = \sum_{i=1}^{N}(1-b)s_{iM}\phi_i(-1) + bs_{i,M-1}\phi_i(1).$$

Inserting the expansions for $f(\bar{x}_{M+1})$ and $f(\bar{x}_M)$ into (3.22) we have

$$\begin{aligned}\sum_{i=1}^{N} S_{ij}\tilde{s}_{jM} = &\left(\sum_{j=1}^{N} s_{jM}\phi_j(1)\right)\overline{\phi_i(1)}\\ &-\left(\sum_{j=1}^{N}(1-b)s_{jM}\phi_j(-1) + bs_{j,M-1}\phi_j(1)\right)\overline{\phi_i(-1)}\\ &-\sum_{j=1}^{N} K_{ij}s_{jM}\end{aligned}$$

or, equivalently, in matrix-vector notation,

$$S\tilde{\mathbf{s}}_{\mathbf{M}} = -bG^{*}\mathbf{s}_{\mathbf{M-1}} + (F + (b-1)E - K)\,\mathbf{s}_{\mathbf{M}}. \tag{3.27}$$

If $f(\bar{x}_{M+1}) = 0$ then the expression in (3.27) reduces to

$$S\tilde{\mathbf{s}}_{\mathbf{M}} = -bG^{*}\mathbf{s}_{\mathbf{M-1}} + ((b-1)E - K)\,\mathbf{s}_{\mathbf{M}}. \tag{3.28}$$

3.2.2 Interior intervals

In this section we consider the interior intervals. We construct the differentiation matrix for the interior intervals using a weighted contribution of the left and right limit at both the left and right boundary of the interval. At the left and right interface we use the parameters b and a respectively:

$$\begin{aligned}f(\bar{x}_l) &= \sum_{j=1}^{N}(1-b)s_{j,l}\phi_j(-1) + bs_{j,l-1}\phi_j(1)\\ f(\bar{x}_{l+1}) &= \sum_{j=1}^{N}(1-a)s_{j,l}\phi_j(1) + as_{j,l+1}\phi_j(-1).\end{aligned}$$

Using this expression in (3.22) yields

$$\begin{aligned}\sum_{j=1}^{N} S_{ij}\tilde{s}_{jl} = &\sum_{j=1}^{N}(1-a)s_{j,l}\phi_j(1)\overline{\phi_i(1)} + as_{j,l+1}\phi_j(-1)\overline{\phi_i(1)} \\ &- \left(\sum_{j=1}^{N}(1-b)s_{j,l}\phi_j(-1)\overline{\phi_i(-1)} + bs_{j,l-1}\phi_j(1)\overline{\phi_i(-1)}\right), \\ &- \sum_{j=1}^{N} K_{ij}s_{jl}\end{aligned}$$

or, in matrix-vector notation,

$$S\tilde{\mathbf{s}}_{\mathbf{l}} = -bG^*\mathbf{s}_{\mathbf{l-1}} + ((1-a)F - (1-b)E - K)\,\mathbf{s}_{\mathbf{l}} + aG\mathbf{s}_{\mathbf{l+1}}, \tag{3.29}$$

for $l = 2, \ldots, M-1$.

3.2.3 Periodic boundary conditions

In order to construct derivative matrices for periodic boundary conditions, we use (3.29) for the interior intervals. For the first and the last intervals we use (3.29) with $l = 1$ and $l = M$, respectively, by identifying $\mathbf{s_0} = \mathbf{s_M}$ and $\mathbf{s_{M+1}} = \mathbf{s_1}$. Namely, we have

$$S\tilde{\mathbf{s}}_{\mathbf{1}} = -bG^*\mathbf{s}_{\mathbf{M}} + ((1-a)F - (1-b)E - K)\,\mathbf{s}_{\mathbf{1}} + aG\mathbf{s}_{\mathbf{2}}, \tag{3.30}$$

and

$$S\tilde{\mathbf{s}}_{\mathbf{M}} = -bG^*\mathbf{s}_{\mathbf{M-1}} + ((1-a)F - (1-b)E - K)\,\mathbf{s}_{\mathbf{M}} + aG\mathbf{s}_{\mathbf{1}}. \tag{3.31}$$

3.2.4 Construction of the derivative matrix

We now have all the components to construct derivative matrices using different coupling across the intervals (by varying the parameters a and b), and for different types of boundary conditions. The derivative matrices with respect to N quadrature nodes over M intervals can be represented by an M-by-M block tri-diagonal matrix. The structure of the matrix is given by

$$D = \begin{pmatrix} r_0^l & r_{-1}^l & & & & & & r_1^l \\ r_1 & r_0 & r_{-1} & & & & & \\ & r_1 & r_0 & r_{-1} & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & r_1 & r_0 & r_{-1} & \\ & & & & & r_1 & r_0 & r_{-1} \\ r_{-1}^r & & & & & & r_1^r & r_0^r \end{pmatrix} \tag{3.32}$$

where each block is an $N \times N$ matrix. The elements of the blocks are given by (3.25)-(3.31) and for some blocks, we can use (3.1) to simplify the expression. We summarize the structure of the blocks in Tables 3.1-3.3.

Table 3.1: Stencil for the derivative matrix at the left boundary.

Characterization	a	b	**Boundary conditions**	r^l_0	r^l_{-1}	r^l_1
No coupling	0	0	Arbitrary, periodic	$S^{-1}K^*$	0	0
No coupling	0	0	$f(-1) = 0$	$S^{-1}(F - K)$	0	0
Forward	1	0	Arbitrary, periodic	$S^{-1}(-E - K)$	$S^{-1}G$	0
Forward	1	0	$f(-1) = 0$	$-S^{-1}K$	$S^{-1}G$	0
Backward	0	1	Periodic	$S^{-1}(F - K)$	0	$-S^{-1}G^*$
Central	1/2	0	Arbitrary	$S^{-1}(\frac{F}{2} - E - K)$	$\frac{S^{-1}G}{2}$	0
Central	1/2	0	$f(-1) = 0$	$S^{-1}(\frac{F}{2} - K)$	$\frac{S^{-1}G}{2}$	0
Central	1/2	1/2	Periodic	$S^{-1}(\frac{F-E}{2} - K)$	$\frac{S^{-1}G}{2}$	$-\frac{S^{-1}G^*}{2}$

Table 3.2: Stencil for the derivative matrix at the right boundary.

Characterization	a	b	**Boundary conditions**	r^r_{-1}	r^r_1	r^r_0
No coupling	0	0	Arbitrary, periodic	0	0	$S^{-1}K^*$
No coupling	0	0	$f(\bar{x}_{M+1}) = 0$	0	0	$S^{-1}(-E-K)$
Forward	1	0	Periodic	$S^{-1}G$	0	$S^{-1}(-E-K)$
Backward	0	1	Arbitrary, periodic	0	$-S^{-1}G^*$	$S^{-1}(F-K)$
Backward	0	1	$f(\bar{x}_{M+1}) = 0$	0	$-S^{-1}G^*$	$-S^{-1}K$
Central	0	1/2	Arbitrary	0	$-\frac{S^{-1}G^*}{2}$	$S^{-1}(F-\frac{E}{2}-K)$
Central	0	1/2	$f(\bar{x}_{M+1}) = 0$	0	$-\frac{S^{-1}G^*}{2}$	$S^{-1}(-\frac{E}{2}-K)$
Central	1/2	1/2	Periodic	$\frac{S^{-1}G}{2}$	$-\frac{S^{-1}G^*}{2}$	$S^{-1}(\frac{F-E}{2}-K)$

Table 3.3: Stencil for the derivative matrix for the interior intervals.

Characterization	a	b	r_1	r_0	r_{-1}
No coupling	0	0	0	$S^{-1}K^*$	0
Forward	1	0	0	$S^{-1}(-E-K)$	$S^{-1}G$
Backward	0	1	$-S^{-1}G^*$	$S^{-1}(F-K)$	0
Central	1/2	1/2	$-\frac{S^{-1}G^*}{2}$	$S^{-1}(\frac{F-E}{2}-K)$	$\frac{S^{-1}G}{2}$

3.2.5 Properties of the derivative matrix

In this section we analyze properties of the derivative matrices for multiple intervals. For simplicity we consider only two intervals. If we uncouple the intervals, we can form

$$D = \begin{pmatrix} \frac{d}{dx} & 0 \\ 0 & \frac{d}{dx} \end{pmatrix}$$

and consider eigenfunctions of this operator. Since an eigenfunction of $\frac{d}{dx}$ is of the form $u(x) = Ce^{\lambda x}$ the eigenvectors of D are of the form

$$u(x) = \begin{pmatrix} u^{(1)}(x) \\ u^{(2)}(x) \end{pmatrix} = \begin{pmatrix} C_1 e^{\lambda x} \\ C_2 e^{\lambda x} \end{pmatrix} \tag{3.33}$$

where the superscripts 1 and 2 refer to the left and right interval respectively. The eigenfunctions are not necessarily continuous at the interface. However, if we impose continuity across the interface, these eigenfunctions become infinitely differentiable over the entire interval.

Let us give a heuristic argument showing that choosing the coupling parameters $a = b = 1/2$ leads to $C_1 = C_2$ in (3.33), that is, smooth eigenfunctions. Without loss of generality we assume an orthonormal basis. According to (3.32) and Tables 3.1 and 3.2 the derivative matrix with $a = b = 1/2$ takes the form

$$D = \begin{pmatrix} \frac{F}{2} - E - K & \frac{G}{2} \\ -\frac{G^*}{2} & F - \frac{E}{2} - K \end{pmatrix} = \begin{pmatrix} -\frac{F}{2} + K^* & \frac{G}{2} \\ -\frac{G^*}{2} & \frac{E}{2} + K^* \end{pmatrix} \tag{3.34}$$

where we used (3.1) for the diagonal blocks. From Section 3.2.1 we see that the operators F and E evaluates a function at the right and left endpoint of the subinterval, respectively. The operator G evaluates a function at the left endpoint on the adjacent subinterval to the right, and G^* evaluates a function at the right endpoint on the adjacent subinterval to the left. From Definition 20 we have that K^* differentiates a function at a subinterval. Assuming eigenfunctions on the form (3.33), and D on the form (3.34), we have that

$$Du = \lambda u + \begin{pmatrix} \frac{e^\lambda}{2}(C_2 - C_1) \\ \frac{e^\lambda}{2}(C_2 - C_1) \end{pmatrix}.$$

In order for u to be an eigenfunction of D, we must have that $C_1 = C_2$ and, therefore, u is smooth.

Chapter 4

Algorithms and numerical results for differentiation and integration of bandlimited functions

In order to use bandlimited functions to solve PDEs numerically, we need to construct derivative operators for bandlimited functions. In this chapter we use the results from the previous chapters to construct such derivative matrices. We provide algorithms and numerical results including a comparison with finite differences and pseudo-spectral methods. Furthermore, we demonstrate that using spectral projectors to remove spurious eigenvalues of derivative matrices with boundary conditions improves the accuracy.

We consider derivative operators for one and multiple intervals in the first two sections. In Section 4.3 we construct the second derivative for Dirichlet boundary conditions and demonstrate the advantage of using spectral projectors. We conclude the chapter with a section on how to construct an integration matrix with respect to bandlimited functions.

4.1 The derivative operator

In this section we construct derivative matrices via integration by parts as described in Chapter 3 using bandlimited functions. As mentioned in Section 2.6 there are three bases available for bandlimited functions on an interval. We choose to construct the derivative matrix with respect to the approximate prolate spheroidal wave functions introduced in Section 2.6.2. This basis is nearly orthogonal and, therefore, allows us to solve (3.5) and (3.22) in a stable manner. However, in many applications it is more convenient to represent functions with respect to function values rather than with its expansion coefficients. Therefore we use the transformation matrix P_c defined by (2.31) to transform the derivative matrix to the interpolating basis defined in Section 2.6.3.

4.1.1 Construction of derivative matrices

The matrix K representing the inner products of basis functions with the first derivative of basis functions can be obtained by using (2.21)-(2.25). Let $\{\theta_k\}_k$ denote a set of quadrature nodes. By introducing the diagonal matrix Θ with the diagonal elements $\Theta_{kk} = \theta_k$ it follows that

$$K = icHQ^TW\Theta\tilde{S}WQH. \tag{4.1}$$

where H, Q, $\tilde{S}$, and W are defined in (2.21)-(2.24). Let us summarize the steps to construct derivative matrices in following algorithm.

Algorithm: Construction of derivative matrices with respect to bandlimited functions

1. Construct N quadrature nodes and weights according to Definition 11.
2. Construct the matrix S according to (2.27), the matrix K according to (4.1), and the matrices E, F, and G in Definition 19 by using (2.21)-(2.25). Construct the matrices P_c and P_c^{-1} according to (2.31).
3. Construct the derivative matrix D for single intervals by using Definition 20, and for multiple intervals by using (3.32) and Tables 3.1-3.3.
4. Compute $\tilde{D} = P_cDP_c^{-1}$ to represent the derivative matrix with respect to function values.

4.1.2 Accuracy of the derivative matrix for varying bandwidth

Using the algorithm above, we present two graphs that illustrate the accuracy of the derivative matrix. We fix the number of nodes and change the accuracy ϵ and observe the change in the bandwidth c.

In the first experiment we use 32 nodes on the interval $[-1, 1]$. Note that the Nyquist frequency for 32 nodes corresponds to the bandwidth $c = 16\pi$ (for periodic functions). We construct four derivative matrices with the bandwidth c set to 5.5π, 7π, 8.5π, and 10.5π, respectively. In order to obtain these bandwidths using 32 nodes, we set the accuracy ϵ to 10^{-13}, 10^{-10}, 10^{-7}, and 10^{-4}, respectively. We differentiate the function $f(x) = e^{ibx}$ for 200 values of b ranging between $\pm 16\pi$. Note that these functions are not necessarily periodic. For each b we differentiate the test function at the 32 quadrature nodes. We then use the matrix C introduced in Section 2.6.4 to interpolate the result at 32 equally spaced points (including the end points) over the interval $[-1, 1]$. The result is shown in Figure 4.1.

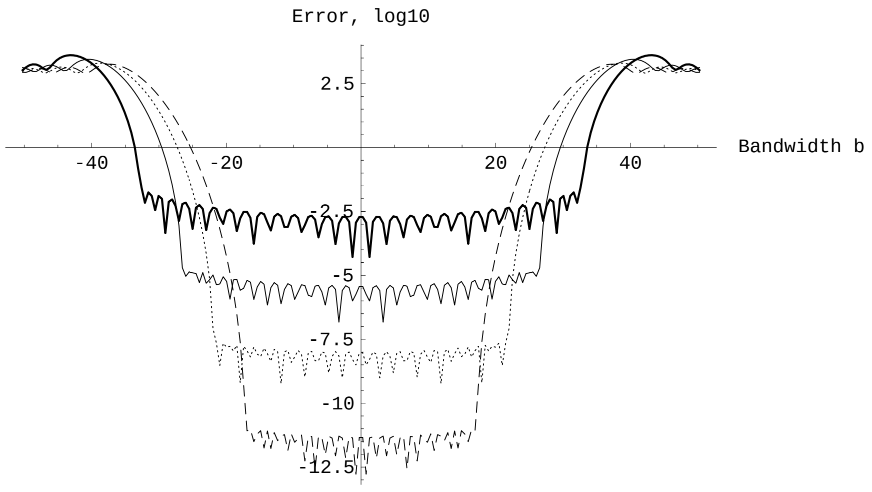


Figure 4.1: Absolute error ($\log_{10}$) for the first derivative of the function e^{ibx} in the interval $[-1, 1]$ with $|b| \leq 16\pi$. The derivative matrices are constructed with respect to 32 basis functions. We use a basis of approximate prolate spheroidal wave functions with maximum bandwidth $c = 10.5\pi$ (thick solid curve), $c = 8.5\pi$ (thin solid curve), $c = 7\pi$ (dotted curve), and $c = 5.5\pi$ (dashed curve). The error of each differentiation is measured at 32 equally spaced points (including the end points) on the interval $[-1, 1]$.

We note from Figure 4.1 that the error profile is almost uniform for $|b| \leq c$. It is also clear that we trade accuracy for bandwidth. A derivative matrix constructed for low accuracy gives a good approximation within a larger bandwidth than a derivative matrix constructed for a higher accuracy.

In the second experiment we use 64 nodes corresponding to the Nyquist frequency 32π (for periodic functions). We construct four derivative matrices with the bandwidth c set to 18.5π, 20.5π, 23π, and 26π, respectively. In order to obtain these bandwidths using 64 nodes, we set the accuracy ϵ to 10^{-13}, 10^{-10}, 10^{-7}, and 10^{-4}, respectively. We differentiate the function $f(x) = e^{ibx}$ for 200 values of b ranging between $\pm 32\pi$. For each b we differentiate the test function on the 64 quadrature nodes. We then use the transformation matrix C introduced in Section 2.6.4 to interpolate the result at 64 equally spaced points (including the end points) over the interval $[-1, 1]$. The result is shown in Figure 4.2. In this experiment we double the

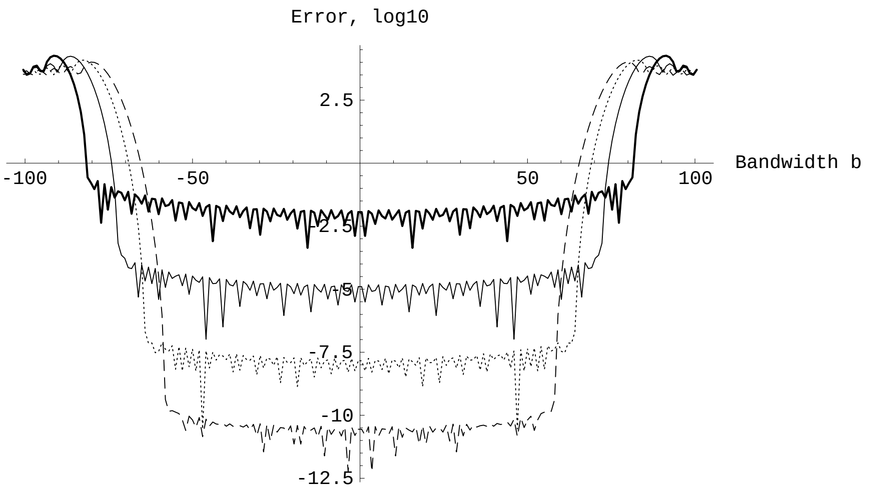


Figure 4.2: Absolute error ($\log_{10}$) for the first derivative of the function e^{ibx} in the interval $[-1, 1]$ with $|b| \leq 32\pi$. The derivative matrices are constructed with respect to 64 basis functions. We use a basis of approximate prolate spheroidal wave functions with maximum bandwidth $c = 26\pi$ (thick solid curve), $c = 23\pi$ (thin solid curve), $c = 20.5\pi$ (dotted curve), and $c = 18.5\pi$ (dashed curve). The error of each differentiation was measured at 64 equally spaced points (including the end points) on the interval $[-1, 1]$.

number of nodes compared to the previous experiment, and observe the same uniform error profile. This demonstrate that we can choose any number of nodes without observing any reduced accuracy. This is not the case with spectral derivative matrices based on polynomials where the accuracy of the derivative matrix deteriorates with increasing number of nodes. The ill-conditioning of derivative matrices based on Chebyshev polynomials has been studied by Greengard [30].

Let us compare the accuracy of differentiation illustrated in Figure 4.1 to the accuracy of interpolation illustrated in Figure 2.5. The profile of the error graphs in the two experiments is similar. The error is nearly uniform within the bandwidth c. However, the absolute error is 2-3 digits greater for the differentiation experiment (Figure 4.1). We give a heuristic explanation to this phenomenon by using Proposition 3. Extend the function u to be differentiated onto the real line such that most of its energy is concentrated within the interval $[-1, 1]$. Then $\alpha \simeq 1$ in Proposition 3 and consequently the ratio $\|Du\|_2/\|u\|_2$ is approximately bounded by c. In particular, this "norm amplification" holds for the error made when approximating u, and hence the absolute error may be amplified by a factor of the order c.

In many situation, the relative error is more relevant. We note that the absolute error for the functions used in Figure 2.5 coincides with the relative error for the functions since the max norm in this case equals one. In contrast, the absolute error displayed in Figure 4.1 is greater than the relative error when $|b| < 1$, since the max norm of $\frac{de^{ibx}}{dx}$ equals b. Thus dividing the absolute error displayed in Figure 4.1 by b, gives a relative error one to two digits smaller than the absolute error for all but the lowest frequencies (see Figure 4.3).

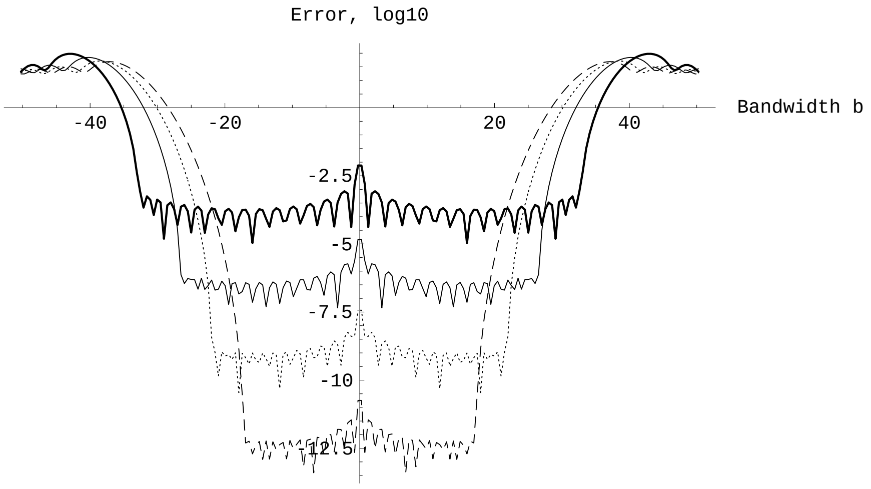


Figure 4.3: Relative error ($\log_{10}$) for the first derivative of the function e^{ibx} in the interval $[-1, 1]$ with $|b| \leq 16\pi$. The derivative matrices are constructed with respect to 32 basis functions. We use a basis of approximate prolate spheroidal wave functions with maximum bandwidth $c = 10.5\pi$ (thick solid curve), $c = 8.5\pi$ (thin solid curve), $c = 7\pi$ (dotted curve), and $c = 5.5\pi$ (dashed curve). The error of each differentiation is measured at 32 equally spaced points (including the end points) on the interval $[-1, 1]$.

4.1.3 Comparison with pseudo-spectral methods and finite differences

Let us now compare the derivative matrix constructed using approximate prolate spheroidal wave functions to a second order finite difference derivative matrix and to a spectral derivative matrix with respect to Chebyshev polynomials. First, we construct such derivative matrices using 32 nodes corresponding to a Nyquist frequency of 16π. We construct two derivative matrices with respect to approximate prolate spheroidal wave functions. We construct one derivative matrix with the accuracy $\epsilon = 10^{-7}$ and bandwidth $c = 8.5\pi$, and another with the accuracy $\epsilon = 10^{-13}$ and bandwidth $c = 5.5\pi$. For comparison, we construct a second order central finite difference derivative matrix using a second order boundary stencil for the first and the last row of the matrix. Finally, we construct a spectral derivative matrix with respect to the first 32 Chebyshev polynomials using the algorithm in [25, Appendix C]. We differentiate the function $f(x) = \sin(bx)$ for 200 values of b ranging between $\pm 16\pi$ and evaluate the result at 32 equally spaced grid points (including the endpoints) on the interval $[-1, 1]$. The result is shown in Figure 4.4.

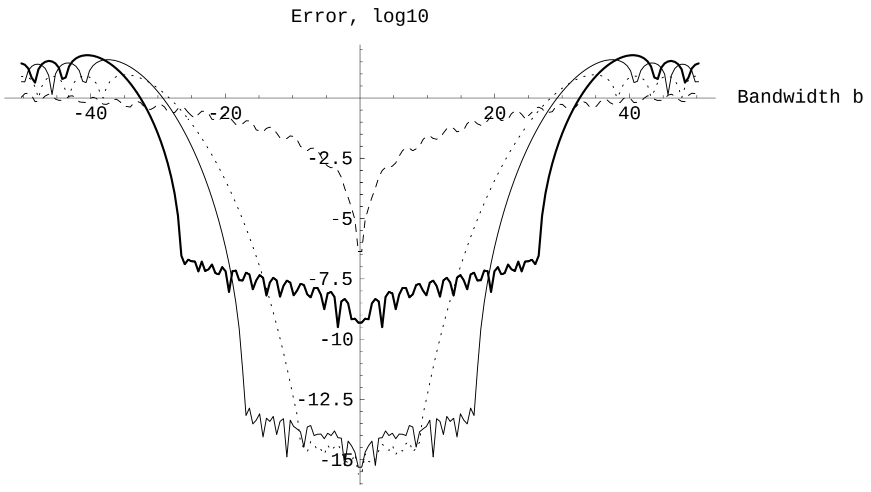


Figure 4.4: Absolute error ($\log_{10}$) for the first derivative of the function $\sin(bx)$ in the interval $[-1, 1]$ with $|b| \leq 16\pi$. The derivative matrices are constructed with respect to 32 nodes or basis functions using a basis of approximate prolate spheroidal wave functions with maximum bandwidth $c = 8.5\pi$ (thick solid curve) and $c = 5.5\pi$ (thin solid curve), the second order central finite difference stencil (dashed curve), and the 32 first Chebyshev polynomials (dotted curve). The error is normalized by dividing by the Nyquist frequency 16π. The error of each differentiation operation was measured at 32 equally spaced points (including the end points) on the interval $[-1, 1]$.

We next consider an experiment using 64 nodes or basis functions corresponding to a Nyquist frequency of 32π. We construct two derivative matrices constructed for approximate prolate spheroidal wave functions with the accuracy $\epsilon = 10^{-7}$ and bandwidth $c = 23\pi$, and another with $\epsilon = 10^{-13}$ and bandwidth $c = 18.5\pi$. For comparison, we construct a second order central finite difference derivative matrix using a second order boundary stencil for the first and the last row of the matrix.

Finally, we construct a block diagonal spectral derivative matrix where each diagonal block is a derivative matrix with respect to the first 16 Chebyshev polynomials. Each block is applied to one of the four subintervals $[-1, -1/2]$, $[-1/2, 0]$, $[0, 1/2]$, and $[1/2, 1]$. The reason for this sub division is that derivative matrices based on Chebyshev polynomials tend to become ill-conditioned for high degree polynomials and are therefore not suited for solving PDEs. The ill-conditioning of derivative matrices based on Chebyshev polynomials is discussed in [30].

We differentiate the function $f(x) = \sin(bx)$ for 200 values of b ranging between $\pm 32\pi$ and evaluate the result at 64 equally spaced grid points (including the endpoints) on the interval $[-1, 1]$. The result is shown in Figure 4.5.

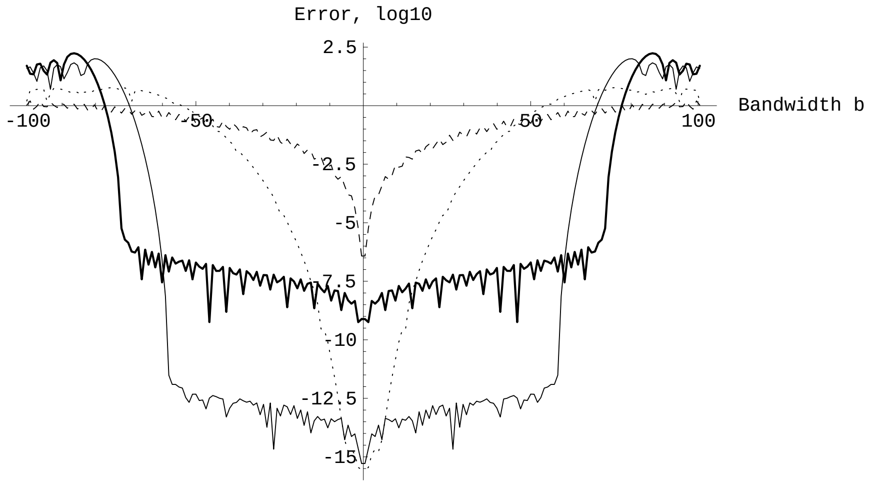


Figure 4.5: Absolute error ($\log_{10}$) for the first derivative of the function $\sin(bx)$ in the interval $[-1, 1]$ with $|b| \leq 32\pi$. The derivative matrices are constructed with respect to 64 nodes or basis functions using a basis of approximate prolate spheroidal wave functions with maximum bandwidth $c = 23\pi$ (thick solid curve) and $c = 18.5\pi$ (thin solid curve), the second order central finite difference stencil (dashed curve), and the 16 first Chebyshev polynomials applied to four subintervals (dotted curve). The error is normalized by dividing by the Nyquist frequency 32π. The error of each differentiation operation was measured at 64 equally spaced points (including the end points) on the interval $[-1, 1]$.

4.1.4 Differentiation of piecewise differentiable functions

Let us now differentiate a continuous piecewise differentiable function. We illustrate the differentiation by defining a continuous function that exhibits different types of behavior on the different subintervals. Consider the function

$$f(x) = \begin{cases} \frac{x+1}{2} & x \in [-1, 1) \\ 1 + \sin(\frac{15\pi(x-1)}{2}) & x \in [1, 3) \\ -P_9(x-4) & x \in [3, 5) \\ -\sin(\frac{3\pi(x-7)}{4}) & x \in [5, 7] \end{cases} \tag{4.2}$$

(see Figure 4.6). Here $P_9(x)$ refers to the Legendre polynomial of degree 9. We construct

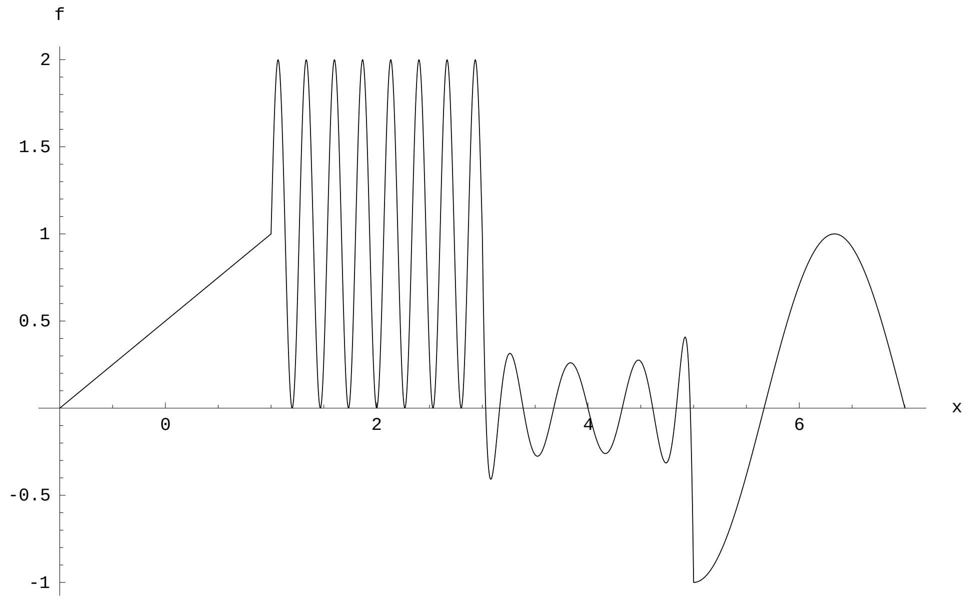


Figure 4.6: Plot of the function in (4.2).

a derivative matrix using (3.32) and Tables 3.1-3.3 with respect to a basis of approximate prolate spheroidal wave functions. We use 32 basis functions constructed for the accuracy $\epsilon = 10^{-7}$ and the bandwidth $c = 8.5\pi$. We couple the intervals using coupling parameters $a = b = 1/2$ according to Section 3.2. The reason for coupling the intervals is that later on when we study wave propagation across interfaces, it will be necessary to couple solutions on neighboring intervals. We will see that this can be accomplished by introducing coupling in the derivative matrix. We sample the function at quadrature nodes on each subinterval, differentiate using the derivative matrix, and evaluate the result at 125 equally spaced nodes (including the endpoints) in the interval $[-1, 7]$. However, we disregard the result at the interface points $x = 1,\ 3$, and 5 where $f'(x)$ is not defined. We normalize the error with $\max_{x\in[-1,7]} |f'(x)| \simeq 28$ and display the relative error in Figure 4.7.

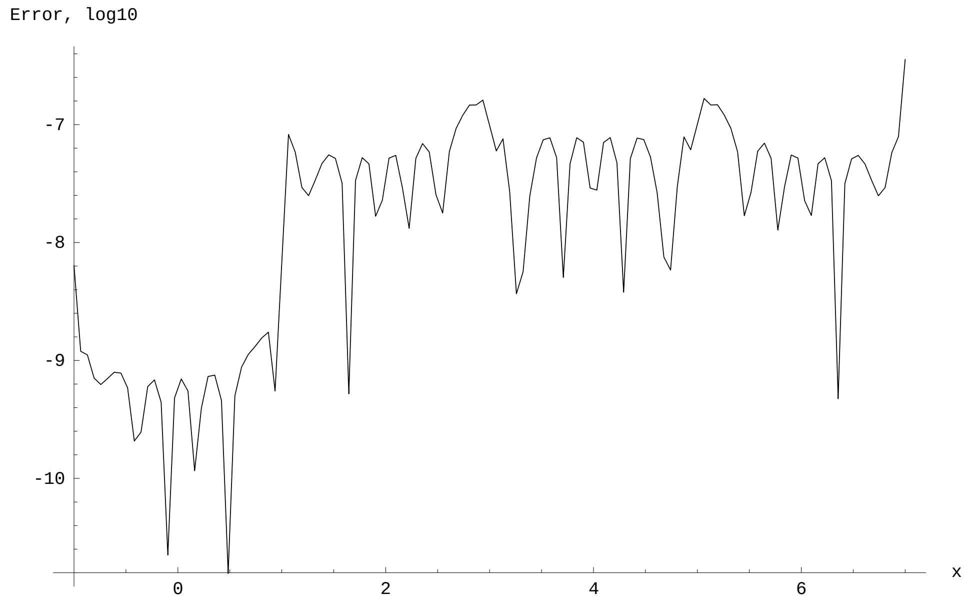


Figure 4.7: The relative error ($\log_{10}$) of $f'(x)$ where $f(x)$ is given in (4.2) and displayed in Figure 4.6.

4.2 Using spectral projectors to improve properties of the derivative matrix with periodic boundary conditions

The algorithm in Section 4.1 can be used to construct derivative operators with periodic boundary conditions. However, the accuracy of the derivative matrix can be increased by projecting the operator. In this section, we modify the algorithm in Section 4.1 for the case of periodic boundary conditions. We begin with showing why it is appropriate to use the technique of projections. A similar technique will be used in Section 4.3 for the second derivative operator with zero boundary conditions.

4.2.1 Properties of the derivative matrix

Let us derive the eigenvalues and eigenfunctions that are required for the derivative operator with periodic boundary conditions. We will see that when constructing the derivative matrix using approximate prolate spheroidal wave functions, it makes sense to apply a projection operator to the derivative matrix to remove spurious eigenvalues.

Consider the eigenvalue problem

$$\begin{cases} Du = \lambda u \\ u(-1) = u(1) \end{cases} . \tag{4.3}$$

It is easily seen that

$$u_k(x) = e^{ik\pi x}$$

for $k = 0, 1, \ldots$ are eigenfunctions to (4.3) with the corresponding eigenvalues

$$\lambda_k = ik\pi.$$

Let us consider a discretization of D obtained by the methods in Section 3.1 using approximate prolate spheroidal wave functions of bandwidth c as the basis. The eigenfunctions of the discretized problem will mimic the eigenfunctions $u_k(x)$. Our choice of basis functions will give good approximations of $u_k(x)$ for all $k = 0, 1, \ldots$ such that $k\pi \leq c$. For k such that $\frac{c}{\pi} < k \leq N$ the eigenvectors will still "attempt" to describe the corresponding eigenfunctions $u_k(x)$, but the accuracy of these approximations will rapidly decrease with increasing k. Therefore, the eigenvectors corresponding to eigenfunctions $u_k(x)$ for $k > \frac{c}{\pi}$ are not useful to us, since we seek approximations only to bandlimited functions within the bandwidth c. Hence, the eigenvectors corresponding to higher frequencies can be disregarded. More formally, let $\mathcal{P}$ denote the projector onto the space spanned by all eigenfunctions $u_k(x)$ such that $k \leq \frac{c}{\pi}$. Our goal is then to find a derivative matrix that approximates the operator $\mathcal{P}D\mathcal{P}$. The details of the construction are given below.

4.2.2 Construction of the projected derivative matrix

To project the derivative matrix D, we diagonalize D as

$$D = E\Lambda E^{-1}$$

where E is a matrix with eigenvectors of D as columns and Λ is a diagonal matrix containing the eigenvalues. The diagonal elements in Λ corresponding to the unwanted eigenspaces are put to zero giving a new diagonal matrix we denote as $\tilde{\Lambda}$. The projected operator denoted as $\tilde{D}$ is then given by $\tilde{D} = E\tilde{\Lambda}E^{-1}$.

Formally, this procedure can be described as follows. Let $\mathbf{e}_k$ and $\mathbf{f}_k$ denote the left and the right eigenvector of D, respectively, with the eigenvalue λ_k. Scale $\mathbf{e}_k$ (or $\mathbf{f}_k$) such that $\mathbf{f}_k^T\mathbf{e}_k = 1$. Define $P_k = \mathbf{e}_k\mathbf{f}_k^T$. If D is an N-by-N diagonalizable matrix then

$$D = \sum_{k=1}^{N} \lambda_k P_k.$$

Then the projected matrix $\tilde{D}$ is given by

$$\tilde{D} = \sum_{|\lambda_k| \leq c}^{N} \lambda_k P_k.$$

As an alternative, one can also use the sign iteration method described in Section 7.2.

A derivative matrix with periodic boundary conditions based on approximate prolate spheroidal wave functions is given as follows.

Algorithm: Derivative matrix with respect to approximate prolate spheroidal wave functions with periodic boundary conditions

1. Construct N quadrature nodes and weights according to Definition 11.
2. Construct the matrix S according to (2.27), the matrix K according to (4.1), and the matrices E, F, and G defined in Definition 19 by using (2.21)-(2.25). Construct the matrix P_c and P_c^{-1} according to (2.31).
3. Construct the derivative matrix D with periodic boundary conditions for single intervals by using Definition 20, and for multiple intervals by using (3.32) and Tables 3.1-3.3.
4. Project D to obtain

   $$D_{proj} = \sum_{|\lambda_k| \leq c} \lambda_k \mathbf{e}_k \mathbf{f}_k^T$$

   where $\mathbf{e}_k$ and $\mathbf{f}_k$ are the left and the right eigenvector of D, respectively, scaled such that $\mathbf{f}_k^T \mathbf{e}_k = 1$.
5. Compute $\tilde{D} = P_c D_{proj} P_c^{-1}$ to represent the derivative matrix with respect to the interpolating basis.

4.3 The second derivative operator with zero boundary conditions

When constructing the second derivative with zero boundary conditions, we can increase the accuracy of the derivative matrix by applying an appropriate projector. The reason for this is similar to the argument that was given for the periodic derivative in the previous section. We begin with modifying the reasoning from Section 4.2.1 to derive an appropriate projector for the second derivative with zero boundary conditions.

4.3.1 Properties of the derivative matrix

Let us denote the derivative operator with arbitrary boundary condition as D, and the derivative operator with zero boundary condition as D_0. We define the linear operator L_0 as $L_0 = DD_0$ and consider the eigenvalue problem

$$\begin{cases} L_0 u = \lambda u \\ u(-1) = u(1) = 0 \end{cases} . \tag{4.4}$$

It is easily seen that

$$u_k(x) = \sin\left(\frac{k\pi}{2}(x+1)\right)$$

for $k = 1, 2, \dots$ are eigenfunctions to (4.4) with the corresponding eigenvalues

$$\lambda_k = -\frac{k^2\pi^2}{4}.$$

Let us consider a discretization of L_0 obtained by the methods in Section 3.1 using approximate prolate spheroidal wave functions of bandwidth c as basis functions. The eigenfunctions of the discretized problem will mimic the eigenfunctions $u_k(x)$. Our choice of basis functions will give approximations of $u_k(x)$ for all $k = 1, 2, \dots$ such that $\frac{k\pi}{2} \leq c$. For k such that $\frac{2c}{\pi} < k \leq N$ the eigenvectors will still "attempt" to describe the corresponding eigenfunctions $u_k(x)$, but the accuracy of these approximations will rapidly decrease with increasing k. Therefore, the eigenvectors corresponding to eigenfunctions $u_k(x)$ for $k > \frac{2c}{\pi}$ are not useful to us, since we seek approximations only to bandlimited functions within the bandwidth c. Hence, the eigenvectors corresponding to higher frequencies can be disregarded. More formally, let $\mathcal{P}$ denote the projector onto the space spanned by all eigenfunctions $u_k(x)$ such that $k \leq \frac{2c}{\pi}$. Our goal is then to find a derivative matrix that approximates the operator $\mathcal{P}L_0\mathcal{P}$. The details of the construction are given in the next section and we give some numerical results in Section 4.3.3 below.

4.3.2 Construction of the second derivative matrix

A second derivative matrix with zero boundary conditions based on approximate prolate spheroidal wave functions is given as as follows.

Algorithm: Second derivative matrix with respect to approximate prolate spheroidal wave functions with zero boundary conditions

1. Construct N quadrature nodes and weights according to Definition 11.

2. Construct the matrix S according to (2.27), the matrix K according to (4.1), and the matrices E, F, and G defined in Definition 19 by using (2.21)-(2.25). Construct the matrix P_c and P_c^{-1} according to (2.31).

3. Construct the derivative matrix D with arbitrary boundary conditions for single intervals by using Definition 20, and for multiple intervals by using (3.32) and Tables 3.1-3.3.

4. Construct the derivative matrix D_0 with zero boundary conditions for single intervals by using Definition 20, and for multiple intervals by using (3.32) and Tables 3.1-3.3.

5. Compute $L_0 = DD_0$.

6. Project L_0 to obtain

$$L_{proj} = \sum_{|\lambda_k| \le c^2} \lambda_k \mathbf{e}_k \mathbf{f}_k^T$$

   where $\mathbf{e}_k$ and $\mathbf{f}_k$ are the left and the right eigenvector of L_0, respectively, scaled such that $\mathbf{f}_k^T \mathbf{e}_k = 1$.

7. Compute $\tilde{L}_0 = P_c L_{proj} P_c^{-1}$ to represent the derivative matrix with respect to the interpolating basis.

4.3.3 Numerical results

Using the algorithm above, we present two experiments that illustrate the accuracy of the second derivative matrix with zero boundary conditions. In the first experiment we use 32 nodes. Note that the Nyquist frequency for 32 nodes corresponds to the bandwidth $c = 16\pi$ (for periodic functions). We construct four derivative matrices with the bandwidth c set to 5.5π, 7π, 8.5π, and 10.5π, respectively. In order to obtain these bandwidths using 32 nodes, we set the accuracy ϵ to 10^{-13}, 10^{-10}, 10^{-7}, and 10^{-4}, respectively. We differentiate the function $f(x) = \sin b_k x$ where $b_k = k\pi/2$ for $k = 1, \ldots, 32$. For each b_k we differentiate the test function at the 32 quadrature nodes. We then use the transformation matrix C introduced in Section 2.6.4 to interpolate the result at 32 equally spaced points (including the end points) over the interval $[-1, 1]$. The result is shown in Figure 4.8.

In the second experiment we use 64 nodes corresponding to the Nyquist frequency 32π (for periodic functions). We construct four derivative matrices with the bandwidth c set to 18.5π, 20.5π, 23π, and 26π, respectively. In order to obtain these bandwidths using 64 nodes, we set the accuracy ϵ to 10^{-13}, 10^{-10}, 10^{-7}, and 10^{-4}, respectively. We differentiate the function $f(x) = \sin b_k x$ where $b_k = k\pi/2$ for $k = 1, \ldots, 64$. For each b_k we differentiate the test function at the 64 quadrature nodes. We then use the transformation matrix C

introduced in Section 2.6.4 to interpolate the result at 64 equally spaced points (including the end points) over the interval $[-1, 1]$. The result is shown in Figure 4.9.

We now demonstrate the importance of projecting the operator as described in Section 4.3.1. Let us construct two second derivative matrices with zero boundary conditions using 32 nodes. One of the matrices is projected according to the algorithm in Section 4.3.2, while the other derivative matrix is constructed without any projection. We choose the bandwidth $c = 8.5\pi$. In order to obtain this bandwidths using 32 nodes, we set the accuracy ϵ to 10^{-7}. For both the derivative matrices, we differentiate the function $f(x) = \sin b_k x$ where $b_k = k\pi/2$ for $k = 1, \ldots, 32$. For each b_k we differentiate the test function at the 32 quadrature nodes. We then use the transformation matrix C introduced in Section 2.6.4 to interpolate the result at 32 equally spaced points (including the end points) over the interval $[-1, 1]$. The result is shown in Figure 4.10.

Note that the error is larger when using the non-projected derivative matrix for all bandwidths $|b_k|$ within the maximum bandwidth c. An explanation to this behavior is given by comparing the norms of the two derivative matrices. For the projected derivative matrix, the Frobenius norm of the matrix was approximately 1431, while the Frobenius norm for the non-projected derivative matrix was approximately 72520. Hence, the norm for the non-projected matrix is approximately 50 times larger than for the projected matrix, which causes amplifications of any error of the function to be differentiated. The reason for the larger norm of the non-projected matrix, is that it tries to approximate highly oscillatory modes. However, the basis functions used in the construction do not have the bandwidth to properly approximate these modes. Therefore, such modes may be disregarded without affecting the class of functions we aim at differentiating. Removing these oscillatory modes decreases the norm of the matrix.

4.4 The integration operator

In solving integral equations it is often useful to map a sequence of function values $\{f(\theta_k)\}_{k=1}^{N}$ to the sequence of integrals $\{\int_{-1}^{\theta_k} f(x)\ dx\}_{k=1}^{N}$. We refer to such a mapping as an integration matrix. If the function can be expanded into basis functions that can be integrated exactly we can achieve high accuracy of the integration.

In this section we construct an integration matrix for the bandlimited functions on an interval. The approach is to represent the function to be integrated with respect to the interpolating basis and then transform to the basis of approximate prolate spheroidal wave functions. We choose the approximate prolate spheroidal wave functions as a basis rather than the exponentials to avoid the numerical instabilities discussed in Section 2.7.1.

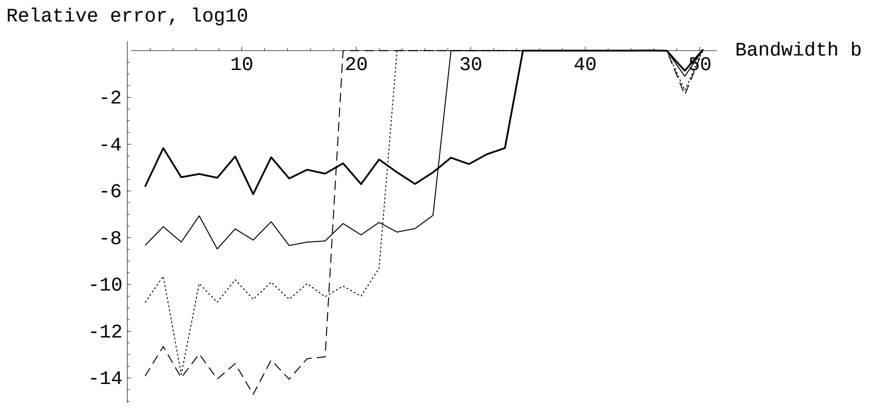


Figure 4.8: Relative error ($\log_{10}$) for the second derivative of the function $\sin b_k x$ where $b_k = k\pi/2$ for $k = 1, \ldots, 32$. The derivative matrices are constructed with respect to 32 basis functions. We use a basis of approximate prolate spheroidal wave functions with maximum bandwidth $c = 10.5\pi$ (thick solid curve), $c = 8.5\pi$ (thin solid curve), $c = 7\pi$ (dotted curve), and $c = 5.5\pi$ (dashed curve). The error of each differentiation is measured at 32 equally spaced points (including the end points) on the interval $[-1, 1]$.

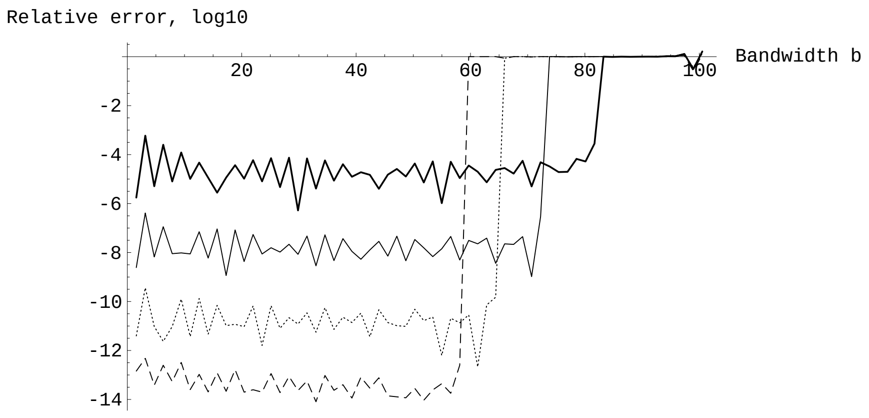


Figure 4.9: Relative error ($\log_{10}$) for the second derivative of the function $\sin b_k x$ where $b_k = k\pi/2$ for $k = 1, \ldots, 64$. The derivative matrices are constructed with respect to 32 basis functions. We use a basis of approximate prolate spheroidal wave functions with maximum bandwidth $c = 26\pi$ (thick solid curve), $c = 23\pi$ (thin solid curve), $c = 20.5\pi$ (dotted curve), and $c = 18.5\pi$ (dashed curve). The error of each differentiation is measured at 64 equally spaced points (including the end points) on the interval $[-1, 1]$.

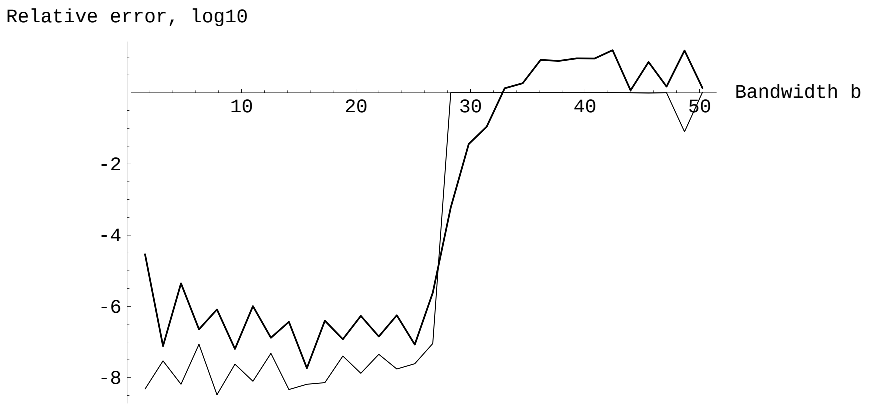


Figure 4.10: Relative error ($\log_{10}$) for the second derivative of the function $\sin b_k x$ where $b_k = k\pi/2$ for $k = 1, \ldots, 32$ using two different derivative matrices. Both derivative matrices are constructed with respect to 32 basis functions using a basis of approximate prolate spheroidal wave functions with maximum bandwidth $c = 8.5\pi$. The first derivative matrix is constructed using projections according to the algorithm in Section 4.3.2 (thin line), while the other derivative matrix was constructed without using any projection (thick line). The error of each differentiation is measured at 32 equally spaced points (including the end points) on the interval $[-1, 1]$.

4.4.1 Construction of the integration matrix

Let $\{\theta_k\}_{k=1}^N$ denote a set of quadrature nodes defined in Definition 11. If N is even, the symmetry of the nodes guarantees that all nodes are non-zero. If N is odd one quadrature node is zero. Define the matrix $\tilde{T}$ with elements defined by

$$\tilde{T}_{kl} = \int_{-1}^{\theta_k} e^{ic\theta_l x}\, dx = \begin{cases} \frac{e^{ic\theta_k\theta_l} - e^{-ic\theta_l}}{ic\theta_l}, & k \neq \frac{N+1}{2} \\ \theta_k + 1, & k = \frac{N+1}{2} \end{cases}. \tag{4.5}$$

for $k, l = 1, \ldots, N$. Note that if N is even, only the first case applies. Using (2.14) we find that

$$\int_{-1}^{\theta_k} \Psi_l(x)\, dx = H_{kk} \sum_{m=1}^{N} W_{mm} Q_{ml} \tilde{T}_{km}$$

The integration matrix based on approximate prolate spheroidal wave functions is given as as follows.

Algorithm: Integration matrix with respect to approximate prolate spheroidal wave functions

1. Construct N quadrature nodes and weights according to Definition 11.
2. Construct the matrices Q, W, and H according to (2.21)-(2.23).
3. Construct $\tilde{T}$ according to (4.5).
4. Construct P_c^{-1} according to Section 2.6.4.
5. Compute the integration matrix T by $T = \tilde{T}WQHP_c^{-1}$.

4.4.2 Numerical results

Using the algorithm above, we present two graphs that illustrate the accuracy of the integration matrix. In the first experiment we use 32 nodes. Note that the Nyquist frequency for 32 nodes corresponds to the bandwidth $c = 16\pi$ (for periodic functions). We construct four integration matrices with the bandwidth c set to 5.5π, 7π, 8.5π, and 10.5π, respectively. In order to obtain these bandwidths using 32 nodes, we set the accuracy ϵ to 10^{-13}, 10^{-10}, 10^{-7}, and 10^{-4}, respectively. Let $\{\theta_k\}_{k=1}^{32}$ be a set of quadrature nodes for bandlimited functions. We compute the integrals $\int_{-1}^{\theta_k} e^{ibx}\, dx$ for 200 values of b ranging between $\pm 16\pi$. Note that it includes integration of non-periodic functions. The result is shown in Figure 4.11.

In the second experiment we use 64 nodes corresponding to the Nyquist frequency for $c = 32\pi$ (for periodic functions). We construct four integration matrices with the bandwidth c set to 18.5π, 20.5π, 23π, and 26π, respectively. In order to obtain these bandwidths using 64 nodes, we set the accuracy ϵ to 10^{-13}, 10^{-10}, 10^{-7}, and 10^{-4}, respectively. Let $\{\theta_k\}_{k=1}^{64}$ be a set of quadrature nodes for bandlimited functions. We compute the integrals $\int_{-1}^{\theta_k} e^{ibx}\,dx$ for 200 values of b ranging between $\pm 32\pi$. Note that this includes non-periodic functions. The result is shown in Figure 4.12.

We note that the error behaves in the same way as for the derivative matrices in Section 4.1. However, the accuracy is in general higher for integration than for differentiation.

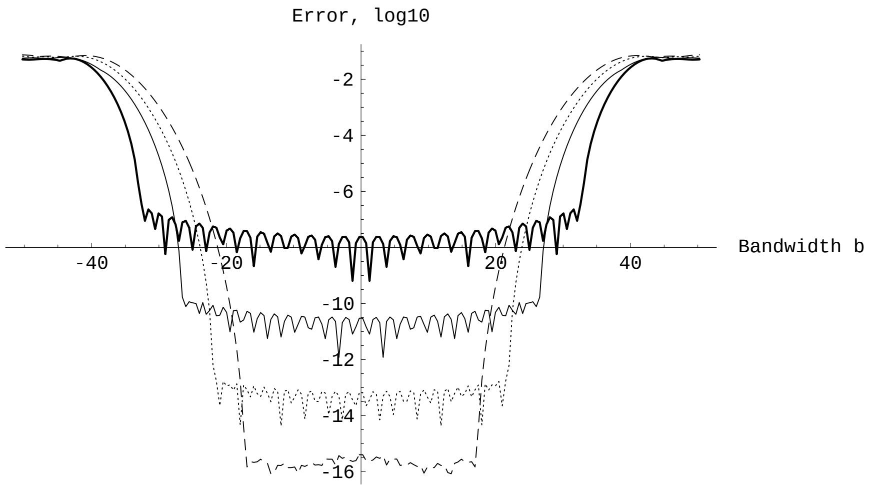


Figure 4.11: Absolute error ($\log_{10}$) for integrals $\int_{-1}^{\theta_k} e^{ibx}\, dx$ in with $|b| \le 16\pi$ where $\{\theta_k\}_{k=1}^{32}$ are quadrature nodes. The integration matrices are constructed with respect to 32 basis functions. We use a basis of approximate prolate spheroidal wave functions with maximum bandwidth $c = 10.5\pi$ (thick solid curve), $c = 8.5\pi$ (thin solid curve), $c = 7\pi$ (dotted curve), and $c = 5.5\pi$ (dashed curve).

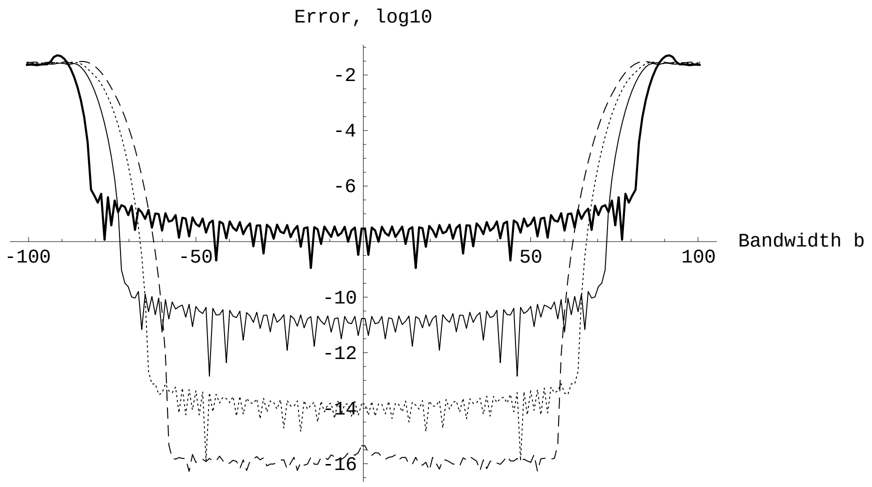


Figure 4.12: Absolute error ($\log_{10}$) for integrals $\int_{-1}^{\theta_k} e^{ibx}\, dx$ in with $|b| \le 32\pi$ where $\{\theta_k\}_{k=1}^{64}$ are quadrature nodes. The integration matrices are constructed with respect to 64 basis functions. We use a basis of approximate prolate spheroidal wave functions with maximum bandwidth $c = 26\pi$ (thick solid curve), $c = 23\pi$ (thin solid curve), $c = 20.5\pi$ (dotted curve), and $c = 18.5\pi$ (dashed curve).

Chapter 5

Low rank representations of operators

In this thesis we propose a scheme to solve the acoustic equation using a semi-group approach. If the spatial operator L in the acoustic equation is time independent, then we solve the equation by computing the matrix exponential e^{tL} using the scaling and squaring method (see Golub and Van Loan [29]). This technique shifts the difficulty into the question of representing the operators so that we can control the complexity of the computations. We also need to maintain an efficient operator representation for relatively large time steps t. If the spatial operator is time dependent, we write the differential equation as an integral equation, and use an iterative scheme to solve such equation. Using the matrix exponential allows large time steps and assures high accuracy. However, the matrix operations are computationally expensive since we need to perform a large number of matrix-matrix multiplications to compute e^{tL}. Computing the matrix exponential in two dimensions directly becomes prohibitively slow even for moderate sizes. The computational cost for a matrix-matrix multiplication in d dimensions grows as $O(N^{3d})$.

In order to overcome the prohibitive computational costs for solving PDEs using the semi-group approach in two or higher dimensions, we need an efficient operator representation. Such multidimensional operator calculus has been introduced by Beylkin and Mohlenkamp [7], and in this chapter we review their work for two dimensional problems. If L is an operator in two dimensions, then the separated representation decomposes the operator L as

$$L = \sum_{k=1}^{r} s_k A_k \otimes B_k \tag{5.1}$$

where $s_k > 0$ are scalars, and A_k and B_k are matrices in one dimension. If the separation rank in (5.1), r, is small, then the separated representation decomposes the operator L to a short sum of operators in one dimension, thus, greatly reducing the computational cost. In Chapter 6 we demonstrate that we can solve the acoustic equation efficiently in two dimensions using this representation.

For a large domain, the operators A_k and B_k are represented by large matrices. The matrices we use are dense, oscillatory, non-Toeplitz, and usually of high rank. This prevents us from using the Fourier or wavelet techniques to compress these matrices. In this thesis we use the so-called Partitioned Low Rank (PLR) representation which is a simplification of the Partitioned Singular Value Decomposition (PSVD) previously studied by Rokhlin et al. ([34], [31], and [48]), and by Beylkin et al. [10]. In this new PLR representation, we partition the matrix such that the diagonal blocks are represented as dense matrices, and the off-diagonal blocks are decomposed into sums of the type

$$B = \sum_{k=1}^{r} s_k e_k f_k^* \tag{5.2}$$

where e_k and f_k are vectors and s_k are scalars. Unlike in PSVD, we do not require the vectors e_k and f_k to be orthonormal. This significantly simplifies the algorithm for reducing the rank in (5.2) after performing linear algebra operations. It turns out that these vectors are close to being orthogonal. In Chapter 6 we demonstrate that the separated representation combined with the PLR representation, efficiently represent operators necessary for solving the acoustic equation. In addition to compute the matrix exponential, we compute spectral projectors in Chapter 7 using this representation for solving inverse problems.

In this chapter we consider algorithms for matrix operations on these representations including the computation of linear combinations and products. The separated rank representation is reviewed in Section 5.1, and we introduce the PLR representation in Section 5.2. When applying these algorithms, we will see that the rank r may grow and it is essential for r to remain small to control the computational cost of the algorithms. We provide an algorithm to reduce the rank in Appendix B.1.

5.1 The separated representation

To solve the acoustic equation in two dimensions we propagate the solution in time by computing the exponential e^{tL}, where L is the spatial operator. This scheme allows us to propagate the solution using large time steps with high accuracy and avoid the time-step restriction that the Courant-Friedrich-Levy (CFL) condition implies on differential formulations on wave propagation problems. The computation of the exponential of a dense matrix is computationally expensive and prohibitively slow for operators in two dimensions. Once e^{tL} has been computed, propagating the solution by computing the matrix vector product $e^{tL}u$ repeatedly is slow in two dimensions. We note that representing a function on an N-by-N mesh requires N^2 samples. We adopt the naming convention from [7] and refer to such a vector as a vector in two dimensions, and refer to an operator acting on a vector in two dimensions as an operator in two dimensions. A straightforward discretization of an operator in two dimensions requires N^4 elements. Consequently, a matrix-vector multiplication costs

$O(N^4)$ operations and a matrix-matrix multiplication costs $O(N^6)$ operations. If the matrices acting in the two directions are banded such that all entries outside the distance $b/2$ from the diagonal are zero, then the computational cost for matrix products is $O(b^4N^2)$ which is still too expensive for solving PDEs. Also, the bandedness of a matrix is not preserved under matrix-matrix multiplications.

The approach in [7] generalizes the usual technique of separation of variables. For example, consider the Laplacian operator in two dimensions which we write as

$$\Delta = \mathcal{D}_{xx} \otimes I + I \otimes D_{yy}$$

where D_{xx} and D_{yy} denote second derivatives, and I denotes the identity operator. The left and the right factors in the tensor products act in the x- and y-directions, respectively. Even if we represent the second derivatives and the identity operator as dense matrices, this representation still requires only $4N^2$ elements to be stored. Generalizations of such representations in higher dimensions have been studied in [7]. In this thesis, we will consider the two dimensional case, but our methods can be generalized to higher dimensions using the algorithms in [7].

Let $L : \mathbb{C}^{N^2} \to \mathbb{C}^{N^2}$ be a linear operator. Let $\{A_k\}_{k=1}^r$ and $\{B_k\}_{k=1}^r$ be N-by-N matrices, and $\{s_k\}_{k=1}^r$ be scalars, such that

$$L = \sum_{k=1}^{r} s_k A_k \otimes B_k.$$

We refer to this sum as a separated representation of L of rank r. Note that we can always find a representation such that $r \leq N^2$. The number of elements in a separated representation is given by $2rN^2 + r$.

This definition is specific for two dimensions, and has no analogue in higher dimensions. In many applications, it suffices to find an approximation to L in a separated form. Therefore we will use

Definition 21 (Separated representation) *For the accuracy $\epsilon > 0$, we define the separated rank representation of an operator L as*

$$\tilde{L} = \sum_{k=1}^{\tilde{r}} s_k A_k \otimes B_k$$

such that

$$\|L - \tilde{L}\| < \epsilon.$$

Usually, we construct the representation such that $s_k > 0$, $\|A_k\| = 1$, and $\|B_k\| = 1$. This definition is sufficient for numerical purposes, and generalizes in higher dimensions [7]. In many cases $\tilde{r} \ll r$.

In one and two dimensions, these two definitions are connected by the SVD, namely, by dropping terms in the SVD expansion we can match the two definitions within a precision determined by the singular values. The SVD in one dimension, $L = \sum_k \sigma_k e_k f_k^*$, has the property that the left factors e_k span the column space (range) of A, and the right factors f_k span the row space of A. Hence, in a sense the SVD separates the input from the output. This is not the case for the separated representation in higher dimensions where the left and the right factors separates the action of L into different directions, but does not separates the input from the output.

As suggested by Beylkin and Mohlenkamp [8], we replace the orthogonality with the weaker requirement that the separation condition number of the operator L defined as

$$\kappa \equiv \frac{\sum_{k=1}^{r} s_k}{\|L\|},$$

is low. (Here we assume that $s_k > 0$ and that A_k and B_k are normalized.)

5.1.1 Operator calculus using the separated rank representation

If $u \in \mathbb{C}^{N^2}$ is a vector in two dimensions, stored as a two dimensional array, (or equivalently, as a matrix in one dimension), then the matrix-vector product in two dimensions can be computed by

$$Lu = \sum_{k=1}^{r} s_k A_k u B_k^T. \tag{5.3}$$

In other words, the matrix A_k acts upon the columns of u, and B_k acts upon the rows of u. The computational cost for a matrix-vector multiplication in two dimensions is given by $O(rN^3)$.

Let us consider examples of linear algebra operations for this representation. Let $L_1 = \sum_{k=1}^{r_1} s_k^{(1)} A_k^{(1)} \otimes B_k^{(1)}$ and $L_2 = \sum_{k=1}^{r_2} s_k^{(2)} A_k^{(2)} \otimes B_k^{(2)}$. We compute linear combinations by

$$\alpha L_1 + \beta L_2 = \sum_{k=1}^{r_1+r_2} \tilde{s}_k \tilde{A}_k \otimes \tilde{B}_k, \tag{5.4}$$

where

$$\{\tilde{s}_k\}_{k=1}^{r_1+r_2} = \{\alpha s_1^{(1)}, \dots, \alpha s_{r_1}^{(1)}, \beta s_1^{(2)}, \dots, \beta s_{r_2}^{(2)}\},$$

$$\{\tilde{A}_k\}_{k=1}^{r_1+r_2} = \{A_1^{(1)}, \dots, A_{r_1}^{(1)}, A_1^{(2)}, \dots, A_{r_2}^{(2)}\},$$

and

$$\{\tilde{B}_k\}_{k=1}^{r_1+r_2} = \{B_1^{(1)}, \dots, B_{r_1}^{(1)}, B_1^{(2)}, \dots, B_{r_2}^{(2)}\}.$$

The rank is reduced using the algorithm in Appendix B.1. Typically, the approximate rank $\tilde{r}$ is significantly less than $r_1 + r_2$.

Similarly, the matrix product of the two separated representations L_1 and L_2 is computed as

$$L_1 L_2 = \sum_{k=1}^{r_1} \sum_{l=1}^{r_2} s_k^{(1)} s_l^{(2)} \left(A_k^{(1)} A_l^{(2)}\right) \otimes \left(B_k^{(1)} B_l^{(2)}\right). \tag{5.5}$$

We note that its rank is $r_1 r_2$. Again, we use the algorithm in Appendix B.1 to reduce the rank of the matrix product after each addition in (5.5). Typically, the approximate rank $\tilde{r}$ is significantly less than $r_1 r_2$.

5.1.2 Separated representation of operators for point wise multiplication

Operators representing pointwise multiplication are needed when considering the acoustic equation $u_{tt} = Lu$ with variable coefficients, where

$$L = \frac{1}{\kappa(x,y)} \frac{\partial}{\partial x} \left(\sigma(x,y) \frac{\partial}{\partial x}\right) + \frac{1}{\kappa(x,y)} \frac{\partial}{\partial y} \left(\sigma(x,y) \frac{\partial}{\partial y}\right).$$

Consider a multiplication operator F represented by the function $f(x,y)$ such that if u is function of two variables, then

$$Fu(x,y) = f(x,y) u(x,y).$$

If we discretize such an operator on an N-by-N mesh we can represent F by the following proposition.

Proposition 22 *A point wise multiplication operator F representing a function f on an N-by-N mesh, can be represented as*

$$F = \sum_{k=1}^{r} \sigma_k U_k \otimes V_k.$$

where U_k and V_k are diagonal matrices.

Proof. Represent f by its SVD,

$$f = \sum_{k=1}^{r} \sigma_k u_k v_k^*$$

where u_k and v_k are the left and right singular vectors, respectively. The proposition follows by constructing the diagonal operators $U_k = diag(u_k)$ and $V_k = diag(v_k)$. □

5.2 The Partitioned Low Rank (PLR) representation

In this section we introduce the Partitioned Low Rank (PLR) representation. This representation turns out to be efficient for many differential operators and functions of such operators. The idea of the PLR representation has been used by Rokhlin et al. ([34], [31], and [48]), for the fast multipole method, and for spectral projectors by Beylkin et al. [10]. The exponential of a matrix with pure imaginary spectrum and the bandlimited derivative matrix constructed in Chapter 4 are of high rank, dense, non-Toeplitz, and highly oscillatory. For exponentials of operators with pure imaginary spectrum there is no decay of modes as time increases. Unlike operators with real, negative spectrum, exponentials of such operators are not necessarily compressible via the wavelet transform while the PLR representation is efficient for functions of differential operators even when wavelet or multiwavelet transforms are dense. In Chapter 6 we apply the technique to exponentials of operators with pure imaginary spectrum, and in Chapter 7 we apply the representation to spectral projectors.

The idea of the PLR representation can be described heuristically as follows. A dense matrix acting on a vector couples all elements of the vector it acts upon. Interaction between elements of the vector set apart roughly by size are of low rank, while interaction between nearby elements are of high rank.

Definition 23 (PLR representation) *Let A be an $\tilde{N}$-by-$\tilde{N}$ matrix where $\tilde{N}$ is even. Partition A into four $\frac{\tilde{N}}{2}$-by-$\frac{\tilde{N}}{2}$ blocks. Decompose the off-diagonal blocks into the separated representation*

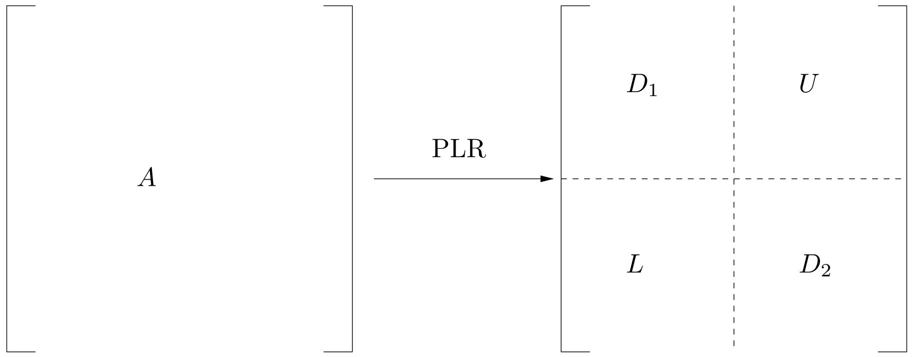
$$B = \sum_{k=1}^{r} s_k e_k f_k^*.$$

We refer to this partition as a level one PLR representation of A.

Let L be an N-by-N matrix for $N = K2^M$ where K is odd and M is a positive integer. Let $m \leq M$. A level m PLR representation of L is defined by the following algorithm.

```
**For** $k = 1 : m$
    **For** $l = 1 : 2^{k-1}$
        *Apply a level one PLR representation to the diagonal block of L given by the elements* $L_{ij},\ i,j = (l-1)\frac{N}{2^k}+1,\dots,\frac{lN}{2^k}$
    **end**
**end**
```

We illustrate the level one PLR representation in Figure 5.1.



$$(D_1)_{ij} = A_{ij},\ i,j = 1,\dots,N/2$$

$$(D_2)_{ij} = A_{ij},\ i,j = N/2+1,\dots,N^2$$

$$U = \sum_{k=1}^{r_U} s_k^{(U)} e_k^{(U)} f_k^{(U)^*}$$

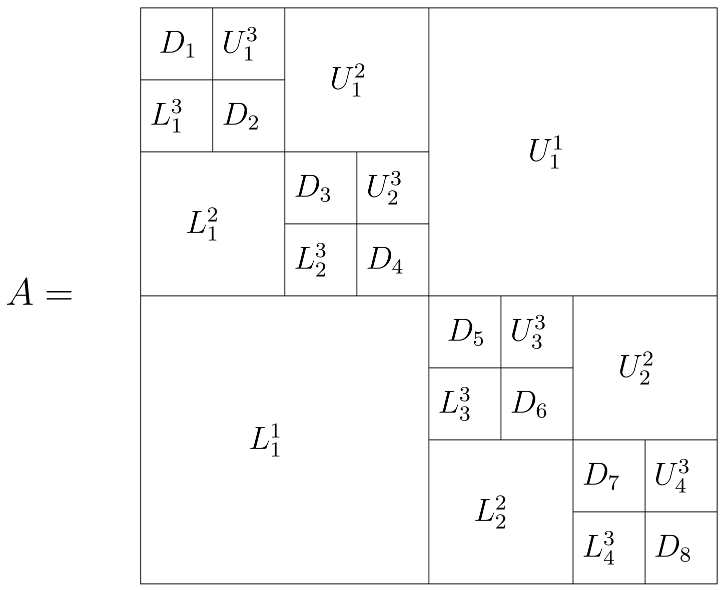
$$L = \sum_{k=1}^{r_L} s_k^{(L)} e_k^{(L)} f_k^{(L)^*}$$

Figure 5.1: Illustration of a level one PLR representation. The diagonal blocks are dense and the off-diagonal blocks are given as separated representations.

The general PLR representation is defined recursively by level one PLR representations and is illustrated in Figure 5.2. In Figure 5.2 we use the notation D_l, U_l^k, and L_l^k to denote the blocks of the partitioned matrix at different levels. This notation is convenient when describing linear algebra operations in the PLR representation.

5.2.1 Compression of matrices using the PLR representation

In all our computations we seek an approximation $\tilde{A}$ of an operator A such that $\|A-\tilde{A}\| < \epsilon$, where $\|\cdot\|$ is an operator norm, for some accuracy $\epsilon > 0$. For many operators represented in



D_l are dense matrices
$U_l^k = \sum_{i=1}^{r_{kl}^{(U)}} (\alpha_{kl})_i (e_{kl})_i (f_{kl})_i^*$
$L_l^k = \sum_{i=1}^{r_{kl}^{(L)}} (\beta_{kl})_i (g_{kl})_i (h_{kl})_i^*$

Figure 5.2: Illustration of a level three PLR representation. The diagonal blocks are dense and the off-diagonal blocks are given as separated representations where α_{kl} and β_{kl} are scalars, and e_{kl}, f_{kl}, g_{kl}, and h_{kl} are vectors.

the PLR representation, the coefficients s_k in the separated representation of the off-diagonal blocks decays rapidly in magnitude. In such cases, the sum in the separated representation can be truncated. In the following theorem, we estimate the threshold value we need to use at each level in a PLR representation of an operator A such that $\|A - \tilde{A}\|_2 < \epsilon$ where $\tilde{A}$ is the truncated operator and $\epsilon > 0$ is some accuracy.

Theorem 24 *Consider an N-by-N matrix A given by a level m PLR representation. Let $\tilde{A}$ denote an approximation of A where each off-diagonal block has been truncated. If we approximate each off-diagonal block $B = \sum_i \sigma_i u_i v_i^*$ given by its SVD, by the operator*

$$\tilde{B} = \sum_{\sigma_i \geq \epsilon_k} \sigma_i u_i v_i^*,$$

where $\epsilon_k = \frac{\epsilon}{2^k m}$ for level k, then

$$\|A - \tilde{A}\|_2 \leq \epsilon.$$

In practice, the separated representation may differ from the SVD, but by performing sufficiently many orthogonalization sweeps in the algorithm in Appendix B.1, we can approximate the SVD with arbitrary accuracy.

Proof. We note that

$$\|A\|_2 \leq \sum_{k=1}^{m} \left(\sum_{l=1}^{2^k} \|D_l^k\|_2 + \sum_{l=1}^{2^{k-1}} \left(\|U_l^k\|_2 + \|L_l^k\|_2 \right) \right). \tag{5.6}$$

An off-diagonal block B at level k is represented by

$$B = \sum_{i=1}^{r} \sigma_i u_i v_i^*.$$

If we truncate this sum using the threshold ϵ_k such that the truncated block $\tilde{B}$ is given by

$$\tilde{B} = \sum_{\sigma_i \geq \epsilon_k} \sigma_i u_i v_i^*,$$

then $\|B - \tilde{B}\|_2 < \epsilon_k$. There are 2^k off-diagonal blocks at each level. Hence, if we select the truncation error $\epsilon_k = \frac{\epsilon}{2^k m}$ and do not truncate the diagonal blocks, the truncated matrix $\tilde{A}$ satisfies

$$\|A - \tilde{A}\|_2 \leq \sum_{k=1}^{m} 2^k \frac{\epsilon}{2^k m} = \epsilon.$$

$\square$

To further justify the use of the PLR representation we review an example from [7]. According to the Christoffer-Darboux formula, the separated sum

$$\sum_{k=0}^{n} p_k(x)p_k(y)$$

where $\{p_k(x)\}_k$ is a set of orthonormal polynomials, can be written as the operator product D_1AD_2 where D_1 and D_2 are diagonal operators and $A = \frac{1}{x-y}$. The matrix A can be efficiently represented by the PLR representation, and multiplying the matrix with diagonal matrices does not change this fact. In contrast, if D_1 and D_2 are diagonal matrices with random numbers along the diagonal, wavelet techniques perform badly when compressing D_1AD_2 even if A is compressible in the wavelet representation.

As an example of the efficiency of the PLR representation, consider the matrix L with elements given by

$$L_{ij} = \begin{cases} \frac{(-1)^{i-j}}{i-j}, & i \neq j \\ 0, & i = j \end{cases} . \tag{5.7}$$

This matrix is of the same form as a derivative matrix on an equally spaced grid of infinite order (see [25, Section 3.2]). The example therefore gives an idea of how derivative matrices in general can be compressed using the PLR representation. In Table 5.1 we show the rank for the off-diagonal blocks along with the compression ratio of the truncated matrix for different sizes. The blocks are truncated such that the relative error in the Frobenius norm is less that 10^{-7}. In the next section we square this matrix using the PLR representation. More examples with the PLR representation are given in Chapter 6 and Chapter 7.

5.2.2 Operator calculus using the PLR representation

In this section we provide algorithms for computing matrix-vector multiplications, linear combinations, and products of matrices in the PLR representation. We use the notation for the blocks introduced in Figure 5.2 and introduce the notation $u(i : j)$ to denote the vector $[u(i), u(i+1), \ldots , u(j)]$ and define N_k as $N_k = \frac{N}{2^k}$.

Let us construct an algorithm to compute the matrix-vector product $v = Au$ where $u \in \mathbb{C}^N$, and A is an N-by-N matrix given as a level m PLR representation. From Figure 5.2 it is clear that there are two types of matrix-vector multiplications we need to compute. First, we need to compute $D_k u((k-1)N_m + 1 : N_m)$ for $k = 1, \ldots , 2^m$ which is a dense matrix-vector multiplication with an N_m-by-N_m matrix. Secondly, we need to compute matrix-vector products of the type $L_l^k \tilde{u}$ and $U_l^k \tilde{u}$ where $\tilde{u} \in \mathbb{C}^{N_k}$, and L_l^k and U_l^k are N_k-by-N_k matrices given as separated representations on the form $\sum_i \sigma_i e_i f_i^*$. If $L_l^k = \sum_{i=1}^r s_i e_i f_i^*$ then the matrix-vector product $L_l^k u$ is computed by the formula

$$L_l^k u = \sum_{i=1}^{r} s_i < u, f_i > e_i. \tag{5.8}$$

Since the computational cost of the inner product is $O(N_k)$, the total cost of such a matrix-vector multiplication scales as $O(rN_k)$. The full algorithm for the matrix-vector multiplication is given in Appendix B.2.

In order to compute linear combinations and products of two PLR representations, we need the following results. Let $L_1 = \sum_{k=1}^{r_1} s_k^{(1)} e_k^{(1)} f_k^{(1)^*}$ and $L_2 = \sum_{k=1}^{r_2} s_k^{(2)} e_k^{(2)} f_k^{(2)^*}$. We compute a linear combination by

$$\alpha L_1 + \beta L_2 = \sum_{k=1}^{r_1+r_2} \tilde{s}_k \tilde{e}_k \tilde{f}_k^* \tag{5.9}$$

where

$$\{\tilde{s}_k\}_{k=1}^{r_1+r_2} = \{\alpha s_1^{(1)}, \dots, \alpha s_{r_1}^{(1)}, \beta s_1^{(2)}, \dots, \beta s_{r_2}^{(2)}\},$$

$$\{\tilde{e}_k\}_{k=1}^{r_1+r_2} = \{e_1^{(1)}, \dots, e_{r_1}^{(1)}, e_1^{(2)}, \dots, e_{r_2}^{(2)}\},$$

and

$$\{\tilde{f}_k\}_{k=1}^{r_1+r_2} = \{f_1^{(1)}, \dots, f_{r_1}^{(1)}, f_1^{(2)}, \dots, f_{r_2}^{(2)}\}.$$

The rank is reduced using the algorithm in Appendix B.1. Typically, the approximate rank $\tilde{r}$ is significantly less than $r_1 + r_2$.

Similarly, the matrix product of the two separated representations L_1 and L_2 is given by

$$L_1 L_2 = \sum_{k=1}^{r_1} \sum_{l=1}^{r_2} s_k^{(1)} s_l^{(2)} < e_l^{(2)}, f_k^{(1)} > e_k^{(1)} f_l^{(2)^*}. \tag{5.10}$$

We note that its rank is $r_1 r_2$. Again, we use the algorithm in Appendix B.1 to reduce the rank of the matrix product after each addition in (5.5). Typically, the approximate rank $\tilde{r}$ is significantly less than $r_1 r_2$. In both (5.9) and (5.10) we note that the structure of the separated representation is preserved under linear combinations and multiplications, respectively. That is, if the input is given as separated representations, then the output is also given as a separated representation.

We next describe an algorithm for computing the product of two matrices given in the PLR representation. In many applications, it is essential that the matrix product preserves the structure of the PLR representation. We present an outline to an algorithm that accomplishes this goal. A detailed algorithm depends on what data structure that is used for storing the data of a PLR representation.

Consider the matrix product $C = AB$ where $A,\ B$, and C are N-by-N matrices given as level m PLR representations. We write A as the block matrix

$$A = \left[\begin{array}{cc} A_{11} & A_{12} \\ A_{21} & A_{22} \end{array} \right]$$

where the off-diagonal blocks are given as separated representations on the form $\sum_i \sigma_i e_i f_i^*$ and the diagonal blocks are level $m-1$ PLR representations. We partition B in the same way. The product A and B involves two types of block multiplications. The first type is a product between two separated representations, and such multiplications can be computed using (5.10) which preserves the separated representation. The second type of block multiplication is a block given as a level $m-1$ PLR representation multiplied with block given in a separated representation. This gives the algorithm a recursive structure.

The product of two level one PLR representations consists of three different types of multiplications; block multiplications between dense (small) matrices, multiplications of a dense block and a separated representation, and products of separated representations. Products of a dense block and a separated representation is computed by repeated use of the matrix-vector algorithm in Appendix B.2, and the product of two separated representations is given by (5.10). A more detailed algorithm is given in Appendix B.3.

We conclude this chapter with a numerical example. We use the operator L used for the experiment in Section 5.2.1 and square this operator using the algorithm in Appendix B.3 combined with the rank reduction algorithm in Appendix B.1. We select the truncation such that the relative error in the Frobenius norm of the product is less than 10^{-7}. We record the time for a standard (dense) matrix-matrix multiplication algorithm (using the subroutine `dgemmm` in the BLAS package), and divide the time with the CPU time for multiplying the matrices using the PLR representation. The time ratio is displayed in Figure 5.3.

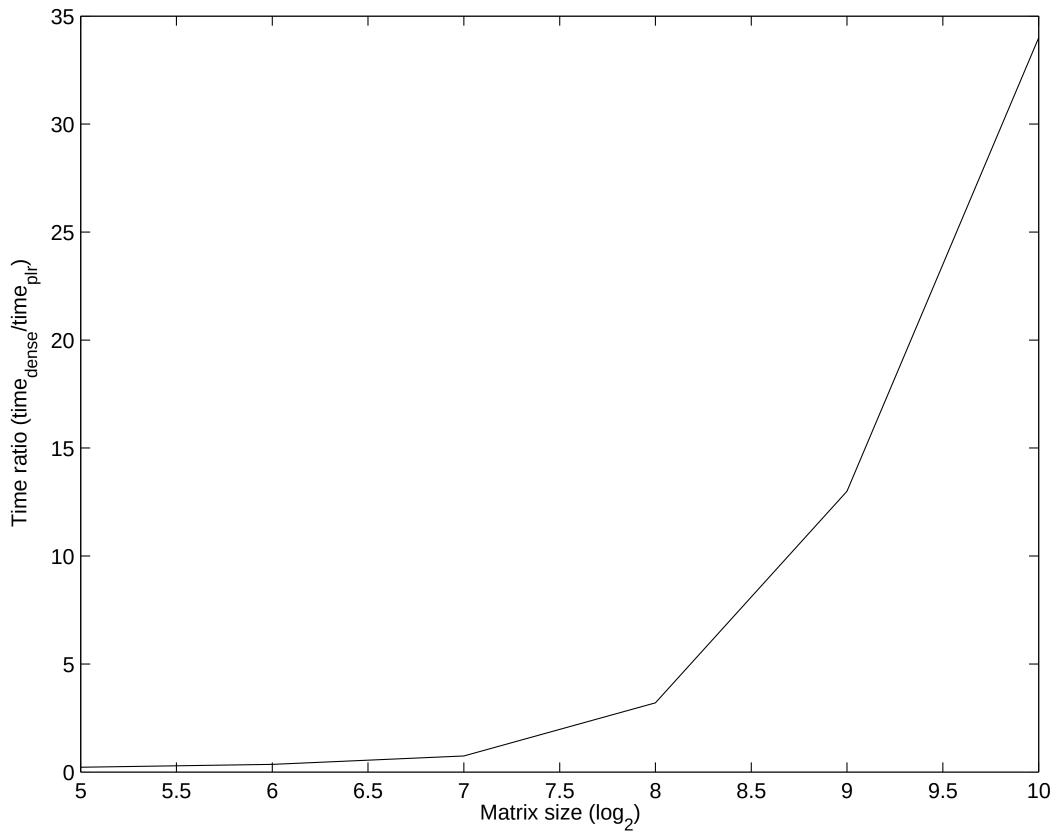


Figure 5.3: Time ratio for computing the square of the matrix in (5.7) using dense multiplication compared with PLR multiplication.

Table 5.1: Compression using the PLR representation. The maximum (full) rank is given within parenthesis. The compression ratio is measured as the number of stored elements for the full dense representation divided by the number of stored elements for the full PLR representation. The off-diagonal block at the fifth level for $N = 512$ has rank 8, and is therefore not worth compressing.

	$N = 32$	$N = 64$	$N = 128$	$N = 256$	$N = 512$	$N = 1024$
Rank of U_1^1	7 (16)	8 (32)	9 (64)	10 (128)	11 (256)	12 (512)
Rank of U_l^2, $l = 1, 2$	N.A.	7 (16)	8 (32)	9 (64)	10 (128)	12 (256)
Rank of U_l^3, $l = 1, \ldots, 4$	N.A.	N.A.	7 (16)	8 (32)	10 (64)	11 (128)
Rank of U_l^4, $l = 1, \ldots, 8$	N.A.	N.A.	N.A.	7 (16)	9 (32)	10 (64)
Rank of U_l^5, $l = 1, \ldots, 16$	N.A.	N.A.	N.A.	N.A.	(dense)	9 (32)
Compression ratio	1.1	1.4	2.0	3.0	4.5	7.3

Chapter 6

Numerical solutions to the acoustic equation in two dimensions

In this chapter we use the tools from this thesis to construct a numerical scheme for solving the acoustic equation in two dimensions. Let us consider

$$\begin{aligned}
&\kappa(x,y)u_{tt} = (\sigma(x,y)u_x)_x + (\sigma(x,y)u_y)_y + F(x,y,t), \quad (x,y)\in\mathcal{D},\ t\in[0,\infty)\\
&u(x,y,0) = f(x,y)\\
&u_t(x,y,0) = g(x,y)\\
&u|_{\partial\mathcal{D}} = h(x,y)
\end{aligned} \tag{6.1}$$

where $\mathcal{D}$ is a rectangle. The function $\kappa(x,y)$ is the compressibility of the medium, and the function $\sigma(x,y)$ denotes the specific volume (the inverse of density).

Iserles [33] gives an overview of finite difference methods for solving hyperbolic problems, and Fornberg [25] discusses the use of pseudo-spectral methods. Alpert et al. use a method related to the spherical means representation to construct a numerical scheme for the wave equation in [2].

Let us first write the acoustic equation (6.1) as a first order system in time. This allows us to use the semigroup approach to solve the equation by constructing a propagator $P(t)$ such that $P(t)u(x,y,0)$ solves the equation at any time t. Since our medium coefficients κ and σ are time independent, the propagator for homogeneous problems ($F=0$) is given by the exponential of a matrix.

In this thesis we introduce a method for solving (6.1) by computing the matrix exponential. To control the computational cost, we represent the spatial operator by the separated representation described in Section 5.1 which decomposes the operator into a short sum of matrices acting in one dimension. These matrices are compressed using the PLR representation introduced in Section 5.2. To minimize the sampling rate, we use the bandlimited functions introduced in Chapter 2 as a basis. We incorporate the boundary and interface conditions according to Chapter 3.

In the first two sections we review the first order formulation of the acoustic equation and the semi-group approach. In the final section we present our algorithm and provide a number of numerical results with constant and variable coefficients. Our scheme is compared to a fourth order scheme using the fourth order finite difference stencil in space and the explicit Runge-Kutta scheme of order 4 (RK4) in time.

6.1 The acoustic equation in two dimensions as a first order system

Our first step is to convert (6.1) to a first order system in time. We follow the derivation by Bazer and Burridge [3] for hyperbolic equations. Once the equation is given in the form $u_t = Lu$, we can use either a traditional time stepping scheme for first order ODEs, such as the RK4, or we can solve the equation by computing the exponential e^{tL}.

By introducing functions v and w, we write the acoustic equation (6.1) as

$$\begin{bmatrix} v \\ w \\ u \end{bmatrix}_t = \begin{bmatrix} 0 & 0 & \sigma(x,y)[\frac{\partial^b}{\partial x} \otimes I_y] \\ 0 & 0 & \sigma(x,y)[I_x \otimes \frac{\partial^b}{\partial y}] \\ \frac{1}{\kappa(x,y)}[\frac{\partial}{\partial x} \otimes I_y] & \frac{1}{\kappa(x,y)}[I_x \otimes \frac{\partial}{\partial y}] & 0 \end{bmatrix} \begin{bmatrix} v \\ w \\ u \end{bmatrix} \tag{6.2}$$

$$+ \begin{bmatrix} 0 \\ 0 \\ \int_0^t F(x,y,\tau)\, d\tau \end{bmatrix} \equiv L\mathbf{u} + \mathbf{F},$$

where the operators $\frac{\partial}{\partial x} \otimes I_y$ and $I_x \otimes \frac{\partial}{\partial y}$ are defined by

$$[\frac{\partial}{\partial x} \otimes I_y]u(x,y) \equiv u_x(x,y) \quad \text{and} \quad [I_x \otimes \frac{\partial}{\partial y}]u(x,y) \equiv u_y(x,y) \ ,$$

and the left and right factors are operators in one dimension acting upon x and y variables, respectively. Here $\frac{\partial^b}{\partial x}$ and $\frac{\partial^b}{\partial y}$ denote differentiation operators with boundary conditions imposed in the x and y direction, respectively. We note that $\frac{\partial}{\partial x}$ and $\frac{\partial^b}{\partial x}$, and $\frac{\partial}{\partial y}$ and $\frac{\partial^b}{\partial y}$, do not generally commute. In Appendix C.1 we look closer at the meaning of $\frac{\partial}{\partial x} \otimes I_y$ when the domain is composed of subdomains where each subdomain has its set of basis functions.

We will use (6.2) when solving the acoustic equation by using the bandlimited functions.

As an alternative, we can write (6.1) as

$$\begin{aligned}\left[\begin{array}{c} v \\ u \end{array}\right]_t &= \left[\begin{array}{cc} 0 & \frac{1}{\kappa}\left(\frac{\partial}{\partial x}\left(\sigma\frac{\partial^b}{\partial x}\right)\otimes I_y + I_x \otimes \frac{\partial}{\partial y}\left(\sigma\frac{\partial^b}{\partial y}\right)\right) \\ I_x \otimes I_y & 0 \end{array}\right]\left[\begin{array}{c} v \\ u \end{array}\right] + \left[\begin{array}{c} F \\ 0 \end{array}\right] \\ &\equiv L\mathbf{u} + \mathbf{F}\end{aligned} \tag{6.3}$$

which we use to construct the fourth order scheme.

6.2 The semigroup approach

In this section we use (6.2) to construct a numerical scheme based on the semigroup approach. The semigroup approach for PDEs is described by, e.g., Yoshida [57] and Evans [23]. Numerical schemes for the semigroup approach for parabolic PDEs and the advection-diffusion equation have been developed by Beylkin and Keiser [5], and by Alpert et al. [1].

We can write (6.2) and (6.3) as

$$\begin{aligned}\mathbf{u}_t &= L\mathbf{u} + \mathbf{F} \\ \mathbf{u}(0) &= \mathbf{u_0}\end{aligned} \tag{6.4}$$

where $\mathbf{u} = [v\ w\ u]^T$ (for (6.2)), $\mathbf{u} = [v\ u]^T$ (for (6.3)), and L is the linear operator on the right-hand side of equations (6.2) and (6.3). From now on we assume that (6.4) is discretized in space resulting in a finite dimensional system of ODEs.

The equation (6.4) is solved by

$$\mathbf{u}(t) = P(t)\mathbf{u_0} + P(t)\int_0^t P(\tau)^{-1}\mathbf{F}(\tau)\ d\tau$$

where the propagator $P(t)$ is an invertible matrix of the same dimensions as L solving the integral equation

$$P(t) = I + \int_0^t L(\tau)P(\tau)\ d\tau.$$

If L is time independent, then $P(t) = e^{tL}$. The propagator $P(t) = e^{tL}$ is continuous with respect to time and has the properties $P(0) = I$ and $P(t+s) = P(t)P(s) = P(s)P(t)$ which gives the propagator the semigroup property. The main difficulty in using this approach as a numerical scheme is to control the complexity of the computation of the matrix exponential.

6.3 Numerical results

In this section we provide examples of wave propagation in two dimensions. We will combine the techniques from previous chapters. We construct L using derivative operators for bandlimited functions with boundary and interface conditions incorporated according to Chapter 3 and propagate the solution by applying the matrix exponential e^{tL}. We refer to this scheme as the bandlimited semigroup method and give the details of the algorithm below.

We compare the resulting scheme to a fourth order finite difference (FD4) scheme where we propagate the solution in time by using the RK4 scheme. For both schemes we display the accuracy for the case with constant coefficients where we compare the result with the exact solution. In addition, we provide examples with variable coefficients. For variable coefficients we cannot compare the solution to the exact solution, but we provide image sequences that illustrate that the solution using the bandlimited semigroup method behaves in the expected manner. In contrast, we demonstrate how the fourth order scheme generates artifacts associated with numerical dispersion.

6.3.1 The bandlimited semigroup method

Let us describe a numerical scheme for solving the homogeneous acoustic equation (6.1) in two dimensions with time independent coefficients.

Algorithm: The bandlimited semigroup method

1. Write the acoustic equation as a first order system in time according to (6.2).

2. Construct the derivative matrices representing $\frac{\partial}{\partial x}$ and $\frac{\partial^b}{\partial x}$ using the algorithm in Appendix C.2.

3. Construct the separated representation of the multiplication operators A and B representing $1/\kappa(x,y)$ and $\sigma(x,y)$, respectively, by using Proposition 22.

4. Construct the spatial operator

$$L = \begin{bmatrix} 0 & 0 & B[\frac{\partial^b}{\partial x} \otimes I_y] \\ 0 & 0 & B[I_x \otimes \frac{\partial^b}{\partial y}] \\ A[\frac{\partial}{\partial x} \otimes I_y] & A[I_x \otimes \frac{\partial}{\partial y}] & 0 \end{bmatrix}.$$

5. Use the algorithm in Appendix B.1 to reduce the separation rank of each block in L.

6. Select Δt (see below) and compute the matrix exponential $e^{\Delta t L}$ using the scaling and squaring algorithm [29]. The linear combinations and products of the matrix blocks given in the separated representation are computed using the methods described in Section 5.1.1.

7. **For** $k = 1 : N_{time}$

   $\mathbf{u}(t_k) = e^{\Delta t L}\mathbf{u}(t_{k-1})$

   end

If the factors in the separated representation, (which are matrices in one dimension), are large, we use the PLR representation described in Section 5.2 to speed up the computations in Step 6 and 7 of the algorithm.

In most numerical methods for wave propagation, the time step is restricted by the Courant-Friedrich-Lewy (CFL) condition, see, e.g., Iserles [33]. To avoid numerical instability according to the CFL conditions, the spatial step is controlled by the speed of the propagating waves. For example, for the wave equation with constant coefficients in one dimension, the ratio of the time step and the spatial step cannot exceed one. When using the matrix exponential, the time step Δt can be chosen as large as desired without causing instabilities as long as the operator L has eigenvalues with non-positive real part. However, a large time step will increase the separation rank and the ranks in the PLR representation. On the other hand, a large time step means that we need fewer time steps in Step 7 of the algorithm. We have found that choosing the time step between 0.5-2 periods gives a good compromise between an efficient representation of the matrix exponential and a small number of time steps in Step 7 of the algorithm.

For this scheme we solve the equation at the quadrature nodes $(x, y) = (\theta_k, \theta_l)$ used for the bandlimited representation. For illustrations, before displaying the solution as a picture, we interpolate the result to an equally spaced grid using the matrix C in Section 2.6.4.

6.3.2 Comparison of the results

We compare the bandlimited semigroup method to two other methods. In the first comparison we use the algorithm in Section 6.3.1, but replace Step 6 and 7 with the explicit RK4 solver in time. In the second comparison we write the acoustic equation (6.1) as a first order system in time using (6.3), and discretize it in space using the fourth order finite difference stencil. We solve the equation on an equally spaced grid including the endpoints. We use fourth order boundary stencils which we construct according to [25] and use the explicit RK4 solver in time.

To evaluate the performance of our method we introduce the following characteristic time and length scales. Consider the equation

$$\begin{cases} u_{tt} = u_{xx} + u_{yy}, \ (x, y) \in (-1, 1) \times (-1, 1) \\ u(x, y, 0) = \sin(c(x + 1)) \sin(c(y + 1)) \\ u_t(x, y, 0) = 0 \\ u(\pm 1, y) = u(x, \pm 1) = 0 \end{cases}$$

where $c = k\pi/2$ for some integer k, and its solution given by

$$u(x, y, t) = \sin(c(x + 1)) \sin(c(y + 1)) \cos(\sqrt{2}ct).$$

Let us adopt the following convention. We define the characteristic (temporal) period

$$\Delta t = \frac{\sqrt{2}\pi}{c},$$

and the characteristic length scale

$$\Delta s = \frac{\pi}{c}.$$

If we consider a fixed time t, then Δs corresponds to the spatial wave length.

6.3.3 Numerical results for constant coefficients

In this section we demonstrate the accuracy of the algorithm and compare its computational speed to that of the two methods described in Section 6.3.2. We also propagate a sharp pulse to demonstrate numerical dispersion.

Comparison of accuracy and speed

For the experiments in this section, we solve

$$\begin{cases} u_{tt} = u_{xx} + u_{yy},\ (x,y)\in(-1,1)\times(-1,1) \\ u(x,y,0) = \sin\left(\frac{\pi(x+1)}{2}\right)\sin\left(\frac{\pi(y+1)}{2}\right) + \sin\left(b(x+1)\right)\sin\left(b(y+1)\right) \\ u_t(x,y,0) = 0 \\ u(\pm1,y) = u(x,\pm1) = 0 \end{cases}, \qquad (6.5)$$

where $b = k\pi/2$ for some integer $k > 1$, and the solution is given by

$$u(x,y,t) = \sin\left(\frac{\pi(x+1)}{2}\right)\sin\left(\frac{\pi(y+1)}{2}\right)\cos\left(\frac{\pi}{\sqrt{2}}t\right)$$
$$+\sin\left(b(x+1)\right)\sin\left(b(y+1)\right)\cos(\sqrt{2}bt).$$

We note that this solution contains both low frequency (the first term) and high frequency (the second term) modes. For the experiments in this section, we measure the error of the vector $\mathbf{u} = [v\ w\ u]^T$ when using a bandlimited scheme, and the error of the vector $\mathbf{u} = [v\ u]^T$ when using the fourth order scheme. The functions v and w are defined in Section 6.1. We measure the error using the relative max norm, that is, if $\tilde{\mathbf{u}}$ approximates the exact solution $\mathbf{u}$, then

$$error = \frac{\|\mathbf{u}-\tilde{\mathbf{u}}\|_\infty}{\|\mathbf{u}\|_\infty}.$$

In the first experiment, we solve (6.5) using $b = 22.5\pi$. We propagate the solution and evaluate the error over a range of $1 - 10^4$ characteristic periods, and also record the computational (CPU) time it took to produce the solution.

For the bandlimited semigroup method , we construct quadrature nodes and weights for the bandwidth $c = 23\pi$ which corresponds to an oversampling factor of approximately 1.4 for periodic functions. We set the accuracy in the construction to $\epsilon = 10^{-7}$ resulting in 64 nodes, and select the time step $\Delta t = \frac{\sqrt{2}}{23}$ corresponding to approximately 0.98 characteristic periods, and represent the operator using the separated and PLR representations from Chapter 5. This gives the separation rank 5 for the blocks in the exponential operator. For the comparison method, we use the same spatial discretization as for the bandlimited semigroup method, but use the RK4 solver in time with the timestep $\frac{\Delta t}{128}$. The result is shown in Figure 6.1. We see that the bandlimited semigroup method is significantly faster than the other method.

In order for the fourth order scheme to reach similar accuracy, we need more than 1024 samples in space corresponding to an oversampling factor of approximately 22 (for periodic

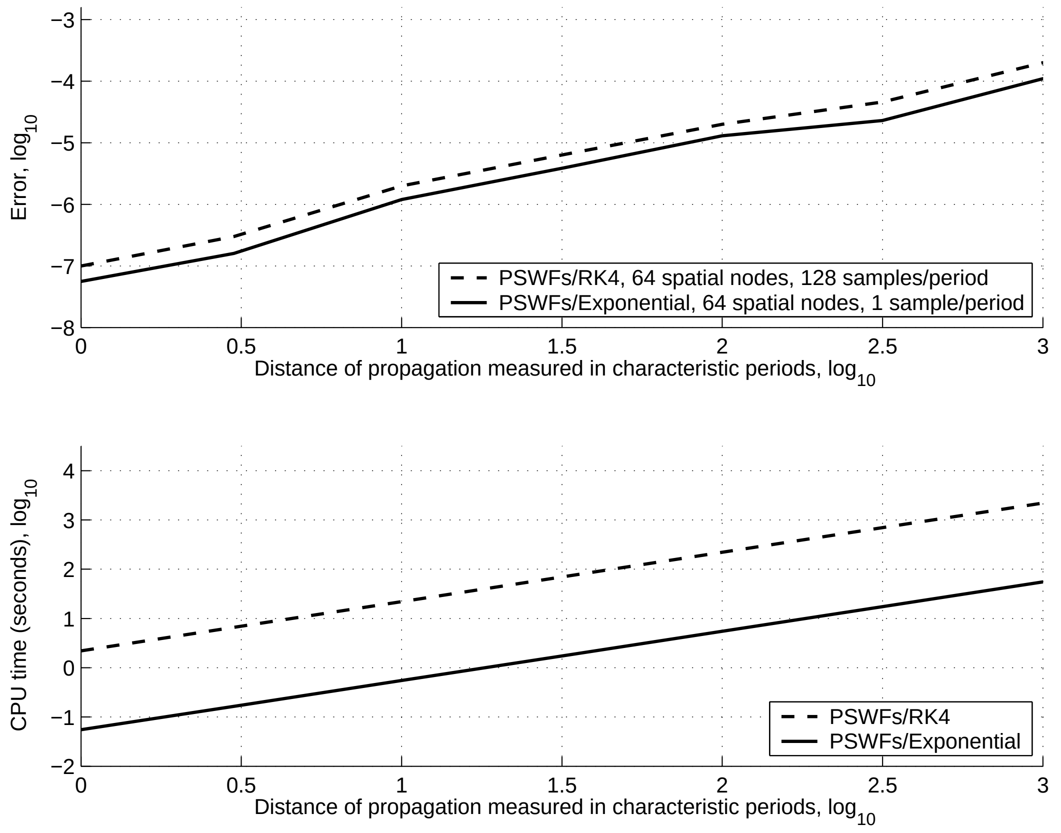


Figure 6.1: Accuracy for solving (6.5) using the approximate prolate spheroidal wave functions as the basis. Relative error ($\log_{10}$) in the max-norm for approximating the solution to (6.5) (top) and the computational time (bottom).

functions) and a timestep $t = \frac{\Delta t}{128}$. With this sampling rate, the computational time per characteristic period is almost four minutes, or more than 5000 times slower than the bandlimited semigroup method. However, such oversampling factor is significantly larger than is typically used. In the next experiment, we therefore solve the same equation as in the previous experiment, but use 400 samples in space for the fourth order scheme, corresponding to an oversampling factor of approximately 8.7 (for periodic functions), and a timestep $\frac{\Delta t}{32}$. For the bandlimited semigroup method we use the same data as in the previous experiment. The result is shown in Figure 6.2. In this experiment, the computational times for the two

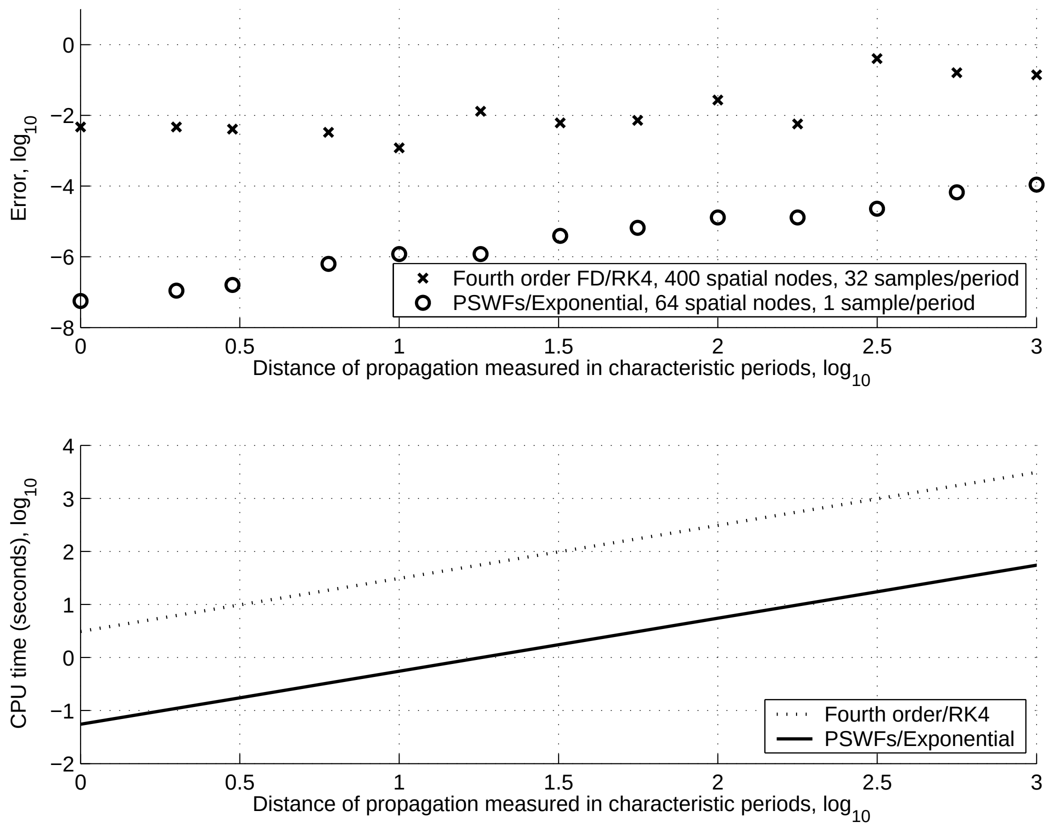


Figure 6.2: Relative error ($\log_{10}$) in the max-norm for approximating the solution to (6.5) (top), and the computational time (bottom).

methods are comparable, but the bandlimited semigroup method is significantly more accurate. The error profile for the fourth order scheme oscillates significantly over time, while the error profile for the bandlimited semigroup method is roughly linear.

In the next experiment, we demonstrate that the cost for higher accuracy is small for the bandlimited semigroup method. Let us fix $b = 19.5$, and solve the model problem

(6.5) using the bandlimited semigroup method for the bandwidth $c = 20\pi$ with 52, 56, 60, 64, and 68 nodes. For all solutions, we use the time step $\Delta t = \frac{\sqrt{2}}{20}$ (approximately one characteristic period). The result is shown in Figure 6.3. We observe that using 60 nodes

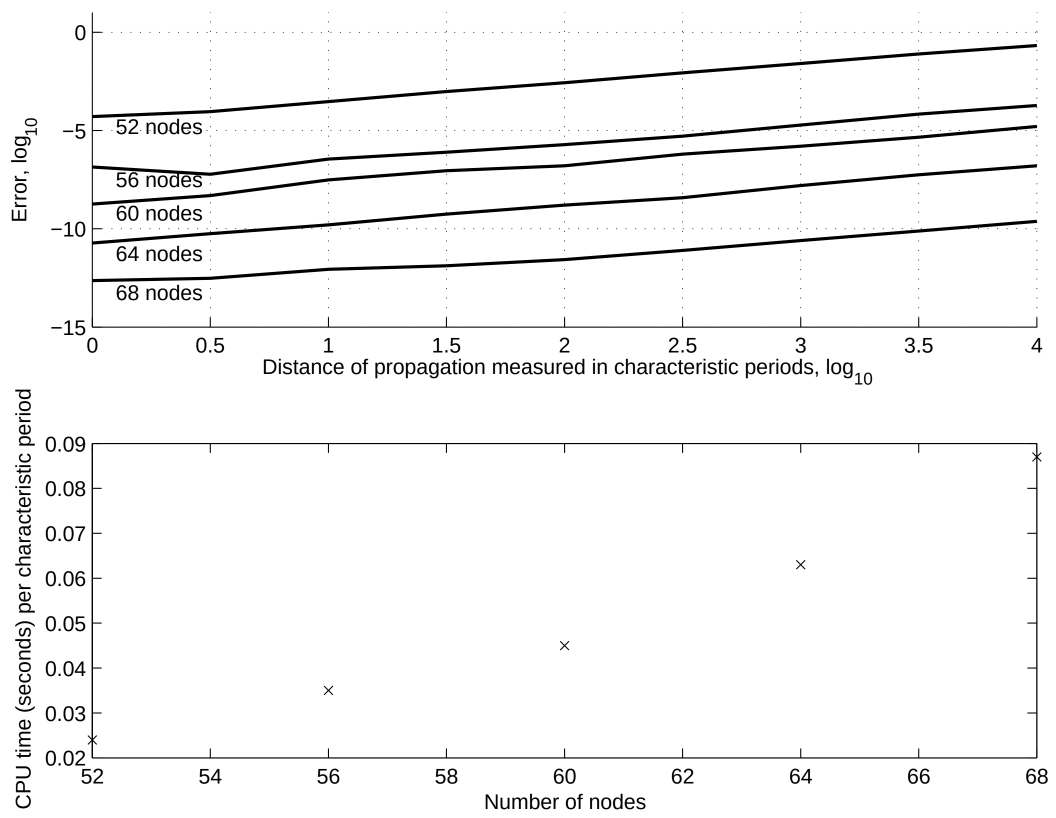


Figure 6.3: Relative error ($\log_{10}$) in the max-norm for approximating the solution to (6.5) for $b = 20\pi$ using two different sampling rates (top). The CPU time for propagating the wave one characteristic period for a varying number of nodes (bottom).

takes approximately two times longer than using 52 nodes but gives approximately 4 more digits of accuracy. We also note that the error increases linearly over time.

Numerical dispersion

Due to inaccuracies of differentiation, the different Fourier modes of a pulse propagate with different speeds. After some time the shape of the pulse deteriorates. To examine this phenomenon, let us consider the wave equation in one dimension, $u_t + cu_x = 0$, the solutions of which correspond to right-traveling waves. Solutions of this equation take the form

$$u(x,t) = e^{i\omega(x-ct)},$$

which we refer to as a Fourier mode of frequency ω traveling to the right with velocity c. Exact differentiation of this solution yields

$$\frac{\partial}{\partial x}u = i\omega e^{i\omega(x-ct)}.$$

If the error in the representation of the differentiation operator is of the form

$$\frac{\partial}{\partial x}u \simeq if(\omega)e^{i\omega(x-ct)},$$

then the Fourier mode propagates with the velocity $cf(\omega)/\omega$. Unless $f(\omega) = \omega$, which corresponds to the exact differentiation, the Fourier modes of different frequencies travel with different velocities. For example, in the case of the second order centered finite difference approximation of the derivative, $f(\omega) = \sin(\omega)$.

In this section we compare numerical dispersion using the bandlimited semigroup method and the fourth order comparison scheme described in Section 6.3.2. Let us solve

$$\begin{cases} u_{tt} = u_{xx} + u_{yy}, \ (x,y) \in (-2,2) \times (-2,2) \\ u(x,y,0) = \operatorname{sinc}^2(27\pi x)\operatorname{sinc}^2(27\pi y) \\ u_t(x,y,0) = 0 \\ u(\pm 2, y) = u(x, \pm 2) = 0 \end{cases}. \tag{6.6}$$

The solution is a sharp pulse originating at the origin of the domain (cf. the example in Section 2.2), and expanding outward. In the absence of numerical dispersion, the shape of the pulse should be maintained.

For the bandlimited semigroup method, we construct 128 quadrature nodes and weights for the bandwidth $c = 54\pi$. We set the accuracy in the construction to $\epsilon = 10^{-7}$. We use a domain consisting of 2-by-2 subdomains with each subdomain of the size 128-by-128 nodes, and use the coupling parameters $a = b = 1/2$ in Table 2.3 for the construction of the derivative matrix. We use the time step $\Delta t = \frac{2\pi}{c}$ corresponding to propagating two characteristic wavelengths, and represent the operator using the separated representation and the PLR representation from Chapter 5. This gives a separation rank of 5-6 for the blocks in the exponential operator. For the fourth order scheme, we use 432 samples in space and the timestep $\Delta t = \frac{\pi}{10c}$ corresponding to propagating a tenth of the characteristic wavelength. This sampling rate gives the two schemes approximately the same computational time. The results are shown as sequences of images in Figure 6.4 and Figure 6.5. We note that for the bandlimited semigroup method, the shape of the pulse is maintained. For the fourth order scheme, the pulse begins to deteriorate into ripples, corresponding to Fourier modes traveling with the wrong velocity.

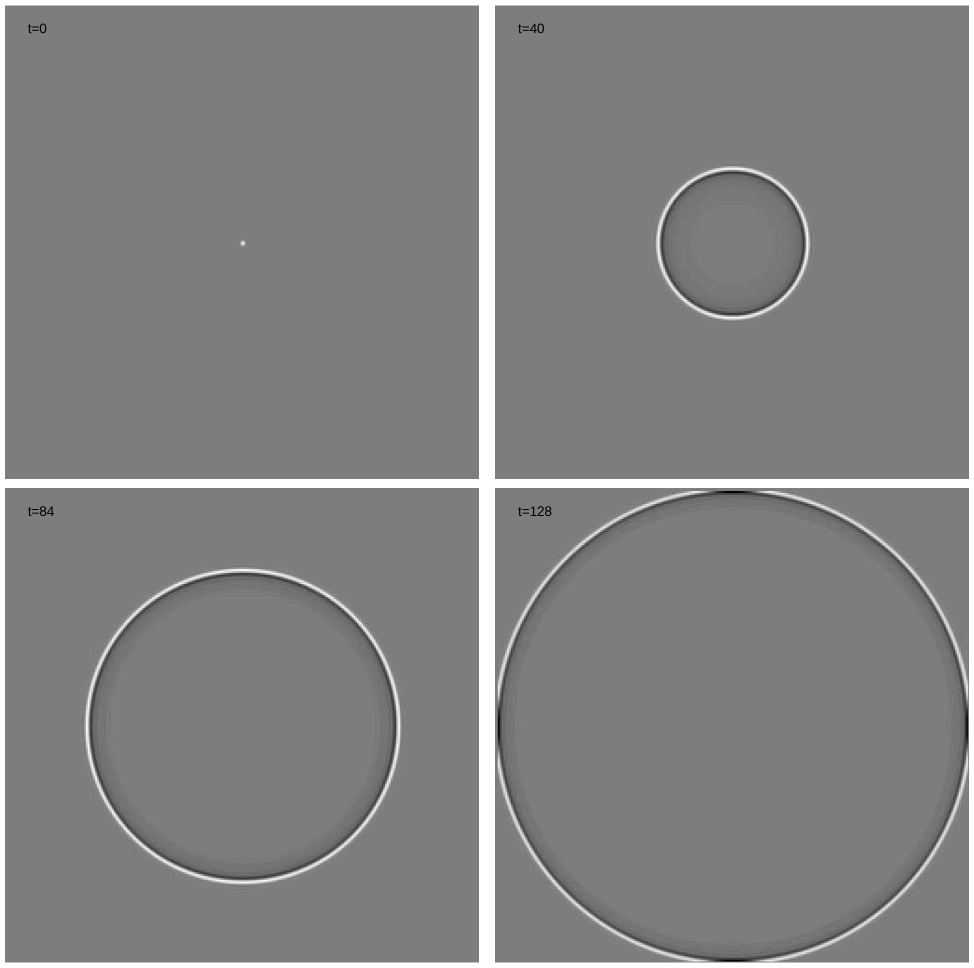


Figure 6.4: Solution of (6.6) using the bandlimited semigroup method. The shape of the pulse is maintained throughout the propagation.

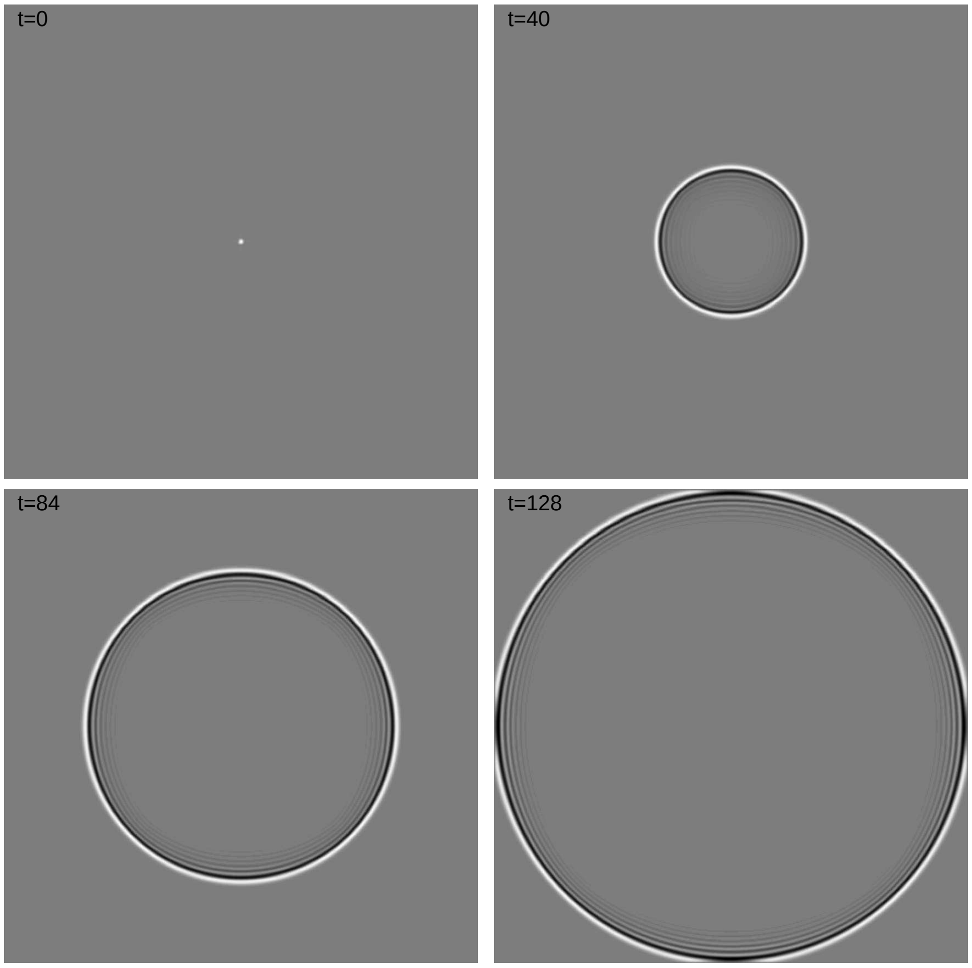


Figure 6.5: Solution of (6.6) using a fourth order scheme. Note the ripples near the wave front which are caused by numerical dispersion.

6.3.4 Numerical results for variable coefficients

In this section we solve the acoustic equation for variable coefficients. Since we do not have an analytical solution to the equation, we evaluate the methods by displaying a sequence of images and study the shape of the pulse as it propagates throughout the domain. Let us solve

$$\begin{cases} u_{tt} = \frac{1}{\kappa(y)}\left(u_{xx} + u_{yy}\right), \ (x, y) \in (-1, 1) \times (-1, 1) \\ u(x, y, 0) = e^{-1000(x^2+y^2)} \\ u_t(x, y, 0) = 0 \\ u(\pm 1, y) = u(x, \pm 1) = 0 \end{cases} \tag{6.7}$$

where

$$\kappa(y) = \frac{1}{1 - \frac{\sin(\pi(y+1))}{2}}.$$

The solution is a sharp pulse originating at the origin of the domain, and expanding out wards with varying velocity.

For the bandlimited semigroup method, we construct 128 quadrature nodes and weights for the bandwidth $c = 54\pi$. We set the accuracy in the construction to $\epsilon = 10^{-7}$ and use the coupling parameters $a = b = 1/2$ in Table 2.3 for the construction of the derivative matrix. We use the time step $\Delta t = \frac{2\pi}{c}$ corresponding to propagating two characteristic wavelengths, and represent the operator using the separated representation and the PLR representation from Chapter 5. We use a level one PLR representation for each matrix in one dimension in the representation of $e^{\Delta t L}$. This gives a separation rank of 7-8 for the blocks in the exponential operator. Using the PLR representation for computing $e^{\Delta t L}u$ is in this case approximately 25% faster than using a dense representation of the matrices in one dimension. The time gain increases for larger problems. For the fourth order scheme, we use 216 samples in space and the timestep $\Delta t = \frac{\pi}{10c}$ corresponding to propagating a tenth of a characteristic wavelength. This sampling rate gives the two schemes approximately the same computational time. The results are shown as sequences of images in Figure 6.6 and Figure 6.7.

Both of the solutions behave qualitatively the same way by propagating faster in the upper part of the domain where the wave velocity is higher. Both solutions also reflect and switch phases at the boundaries. We note that for the bandlimited semigroup method, the shape of the pulse is maintained. For the fourth order scheme, the pulse begins to deteriorate into ripples, corresponding to Fourier modes traveling with the wrong velocity.

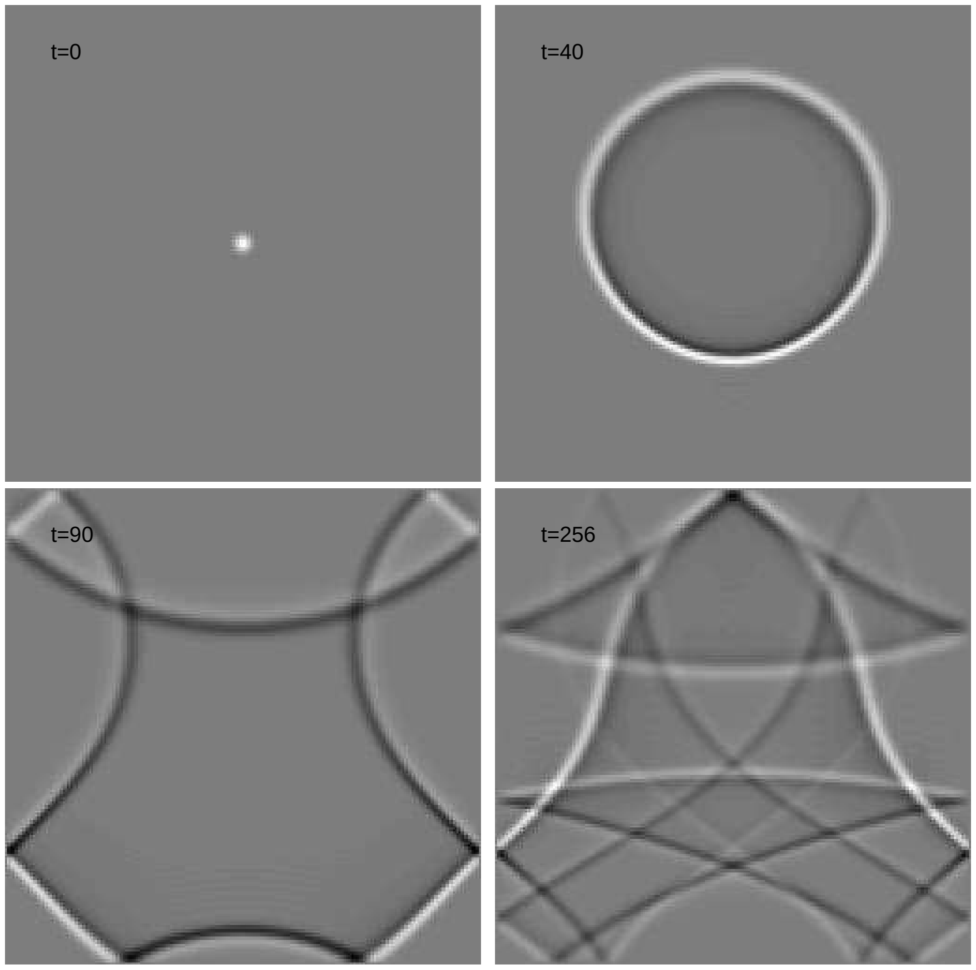


Figure 6.6: Solution of (6.6) using the bandlimited semigroup method.

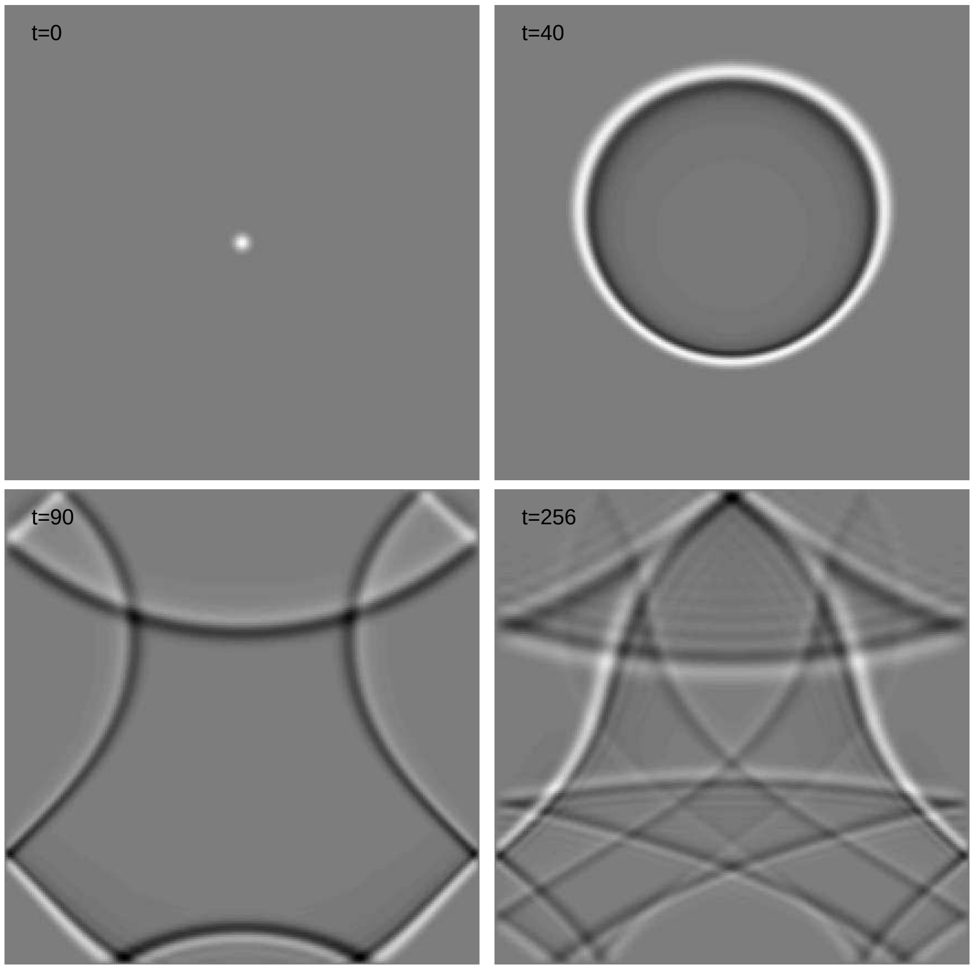


Figure 6.7: Solution of (6.6) using a fourth order scheme. Note the ripples which are caused by numerical dispersion.

Chapter 7

Wave propagation on space-like surfaces

In this chapter we consider wave propagation on space-like surfaces. This problem appears, for example, in solving the inverse problem for acoustic wave propagation. Let us consider the equation

$$\begin{aligned} &\frac{1}{\kappa(x)}\Delta u + \omega^2(1+f(x))u = 0,\ \omega\in\mathbb{R},\ x\in\mathbb{R}^n \\ &u(x) = e^{i\omega x\cdot\nu} + w(x),\ ||\nu|| = 1 \end{aligned} \tag{7.1}$$

where $w(x)$ satisfies the Sommerfeld radiation condition and $\kappa(x) = 1$ for x outside some bounded domain. We refer to $e^{i\omega x\cdot\nu}$ as the incident wave and κ as the background compressibility. The wave velocity is given by $1/\sqrt{\kappa}$.

For the inverse problem, our goal is to determine the scatterer f, where u is known at a boundary outside the scatterer. The inverse problem for acoustics in two and three dimensions has been studied extensively, see Colton and Kress [15], Natterer and Wübbeling [44], and Chen [14], and references therein. In this thesis, we consider problems in two dimensions.

To introduce our approach, let us first consider a scheme proposed by Natterer and Wübbeling [44]. They introduced an iterative scheme to solve the inverse problems (7.1) for constant background in situations where the Born or Rytov approximations are not valid. The method requires repeated solution of forward problems of the type

$$\begin{aligned} &u_{yy} = -u_{xx} - \kappa(x,y)\omega^2(1+f)u \equiv Au,\ (x,y)\in[-1,1]\times[-1,1] \\ &u(x,-1) = g(x) \\ &u_y(x,-1) = h(x) \\ &u(-1,y) = r(y) \\ &u(1,y) = s(y) \end{aligned} \quad . \tag{7.2}$$

We refer to this type of problem as wave propagation on space-like surfaces. This equation is inherently unstable. In [44] this difficulty is solved for constant background by using the Fourier techniques to filter out the high frequencies of the solution u, but does not generalize to variable background.

A more typical approach in applications is to factor the equation into waves going along some preferred direction (e.g. vertically), rather than using the approach by Natterer and Wübbeling. For example, this method has been studied for geophysical applications, see Gautesen and de Hoop [27] and references therein. The apparent advantage of factorization is that it can deal, although approximately, with variable background. The advantage of Natterer and Wübbeling's approach, however, is that it does not suppress the backscattered waves. In this thesis, we propose a generalization to this method for variable background. It is a first step to solve the multi-dimensional inverse problem. We demonstrate how this can be done in the case of y-independent coefficients, and plan to discuss the general problem elsewhere.

We note that the operator A is self-adjoint and, therefore, has a real spectrum. In general, A has both negative and positive eigenvalues. By projecting the operator A onto the eigenspace associated with the negative eigenvalues, we obtain a negative definite operator. In Section 7.1 we illustrate how a negative definite operator gives a stable problem. This shifts the problem to the question of constructing a fast algorithm for computing spectral projectors. As discussed in Beylkin et al. [10], fast computations of spectral projectors is also of interest for many-body problems in atomic physics. In the second section of this chapter we construct an algorithm for fast computation of spectral projectors using the sign iteration combined with the operator representations from Chapter 5. In the third section, we use the spectral projectors along with the tools for wave propagation presented in this thesis to solve the wave propagation problem on space-like surfaces.

7.1 Spectral projectors for wave propagation problems on space-like surfaces

Let us see how using spectral projectors gives a stable scheme for solving (7.2). We establish stability for the case when the compressibility κ of the background and the scatterer f are y-independent and plan to address the general case in future work.

If we disctretize (7.2) in the x-direction and incorporate the boundary conditions into the spatial operator according to Chapter 3, we can establish stability from the following well-known results from the theory of ODEs.

Theorem 25 *Let $\mathbf{u}\in\mathbb{R}^N$ and let A be a negative definite diagonalizable matrix. Then the* *solution* $\mathbf{u}(y)$ *to the equation*

$$\begin{aligned}\mathbf{u}_{yy} &= A\mathbf{u},\ \ y \geq 0\\ \mathbf{u}(0) &= \mathbf{u_0}\\ \mathbf{u}_y(0) &= \mathbf{v_0}\end{aligned} \tag{7.3}$$

is stable, that is, there exists a constant K *such that* $\|\mathbf{u}(y)\| \leq K$ *for all* $y \geq 0$.

Proof. We first write (7.3) as the first order system

$$\begin{bmatrix}\mathbf{v}\\ \mathbf{u}\end{bmatrix}_y = \begin{bmatrix}0 & A\\ I & 0\end{bmatrix}\begin{bmatrix}\mathbf{v}\\ \mathbf{u}\end{bmatrix} \equiv L\tilde{\mathbf{u}} \tag{7.4}$$

where I is the N-by-N identity matrix. Since A is diagonalizable, there exists an invertible matrix S such that $L = S^{-1}L_DS$ where L_D is a diagonal matrix with the eigenvalues λ of L along the diagonal. Define $\mathbf{w} = S\tilde{\mathbf{u}}$. Since L is y-independent, S commutes with $\frac{d}{dy}$ and we can write (7.4) as $\frac{d\mathbf{w}}{dy} = L_D\mathbf{w}$ or, equivalently, solve

$$\frac{dw_k}{dy} = \lambda_k w_k$$

for $k = 1, 2, \dots, 2N$ where $[w_1(y)\ w_2(y) \cdots w_{2N}(y)]^T = \mathbf{w}(y)$. Since $w_k(y) = e^{\lambda_k y}w_k(0)$ we can define $C = \sup_k |w_k(0)|$ to establish the bound $|w_k(y)| \leq |e^{\lambda_k y}|C$. Since A is negative definite, the eigenvalues λ_k are pure imaginary according to Appendix C.2 and hence

$$\|\mathbf{w}(y)\| = \sqrt{\sum_{k=1}^{2N} |w_k(y)|^2} \leq \sqrt{2N}C$$

for $y \geq 0$. Therefore,

$$\|\mathbf{u}(y)\| \leq \|\tilde{\mathbf{u}}(y)\| = \|S^{-1}\mathbf{w}(y)\| \leq \|S^{-1}\|\sqrt{2N}C$$

which is bounded for any fixed N. $\square$

Hence, by projecting the operator A in (7.2) onto the eigenvectors corresponding to negative eigenvalues, we can solve the equation in a stable manner.

Let us now see how this approach relates to the approach by Natterer and Wübbeling. In their approach the instability is recognized as a high frequency phenomenon and stability is imposed by low pass filtering the solution. To see this, consider the operator $L = -\frac{\partial^2}{\partial x^2} - \omega^2 I$ where ω is real and I denotes the identity operator. The eigenfunctions of L are linear combinations of the functions $u(x) = e^{\pm i\sqrt{\omega^2+\lambda}x}$ where λ is an eigenvalue of L. If we impose the boundary condition $u(\pm 1) = 0$, then $\lambda = -\omega^2 + \frac{k^2\pi^2}{4}$ for $k = 0, 1, \dots$. If λ is negative, then we must have that $\frac{k\pi}{2} \leq |\omega|$. Now consider the equation

$$u_{yy} = -u_{xx} - \omega^2 u$$

with the boundary conditions $u(\pm 1) = 0$. The solutions to this equation take the form

$$u(x,y) = \sin\left(\frac{k\pi(x+1)}{2}\right) \cos\left(\sqrt{\omega^2 - \frac{k^2\pi^2}{4}}(y+1)\right)$$

for $k = 1, 2, \dots$. Hence, it is clear that these solutions are bounded for all $y \geq 0$ if $\frac{k\pi}{2} \leq |\omega|$. In other words, by filtering out high frequency modes in the x-direction, we effectively project the solution onto bounded functions.

7.2 Computation of spectral projectors

In this section we consider the problem of computing spectral projectors. Consider a diagonalizable matrix A with pure real spectrum. Given its spectral decomposition

$$A = \sum_k \lambda_k P_k$$

where $\{\lambda_k\}_k$ are the eigenvalues of A, and $\{P_k\}_k$ are projectors, we construct a fast algorithm to compute the spectral projector

$$P^{(\mu)} \equiv \sum_{\lambda_k < \mu} P_k$$

such that

$$P^{(\mu)} A = \sum_{\lambda_k < \mu} \lambda_k P_k$$

without computing the individual operators P_k. For self-adjoint matrices, $P_k = \mathbf{e_k}\mathbf{e_k}^T$ where $\mathbf{e_k}$ is the eigenvector corresponding to the eigenvalue λ_k. For general diagonalizable matrices, $P_k = \mathbf{e_k}\mathbf{f_k}^T$ where $\mathbf{e_k}$ and $\mathbf{f_k}$ are the right and left eigenvectors, respectively. However, constructing the spectral projector by computing eigenvectors can be costly. In our approach, we use the method by Beylkin et al.[10] where the sign function is computed by an iterative scheme that only requires matrix-matrix multiplications and additions. This shifts the difficulty to representing the matrix so that the matrix products can be computed efficiently. We use the separated representation and the PLR representation from Chapter 5 to represent the operators so that matrix products can be computed rapidly.

7.2.1 The spectral decomposition of a diagonalizable matrix

In [10] fast algorithms for computing spectral projectors are constructed for self-adjoint matrices. The derivative matrices for bandlimited functions constructed in Section 4.1 are neither self-adjoint nor normal. However, these matrices are diagonalizable and have pure imaginary spectrum. The second derivative operators constructed in Section 4.3 are not normal, but diagonalizable and have pure real spectrum. In this and the next section, we construct spectral projectors for diagonalizable matrices with pure real or pure imaginary spectrum. Let us first consider the spectral decomposition of a matrix which is diagonalizable but not necessarily self-adjoint.

Proposition 26 *Let A be a matrix where all eigenvalues have algebraic multiplicity one. Let $\mathbf{e_k}$ and $\mathbf{f_k}$ be the right and the left eigenvectors of A, respectively, and let λ_k be the eigenvalue corresponding to the eigenvector $\mathbf{e_k}$. Scale $\mathbf{e_k}$ (or $\mathbf{f_k}$), such that ${\mathbf{f_k}}^T\mathbf{e_k} = 1$. Define*

$$P_k = \mathbf{e_k}{\mathbf{f_k}}^T.$$

Then

1. $P_k\mathbf{e_k} = \mathbf{e_k}$
2. $P_kP_l = \delta_{kl}P_k$
3. $I = \sum_k P_k$
4. $A = \sum_k \lambda_k P_k$
5. $P_k\mathbf{x} \in \operatorname{span}\{\mathbf{e_k}\}$

The proof is given in Appendix D.1. We note that if A is self-adjoint, then $\mathbf{f_k} = \mathbf{e_k}$.

7.2.2 The sign function

Let us show how spectral projectors can be constructed by computing the sign function. We define the sign function for real values x by

$$\operatorname{sign}_{\mathrm{Re}}(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases},$$

and the sign function for imaginary values z by

$$\operatorname{sign}_{\mathrm{Im}}(z) = \begin{cases} i, & \mathrm{Im}(z) > 0 \\ 0, & \mathrm{Im}(z) = 0 \\ -i, & \mathrm{Im}(z) < 0 \end{cases}.$$

We will write sign( ) without suffix when there is no chance of confusion. We define the sign function for the diagonalizable matrix A with pure real or pure imaginary spectrum by

$$\mathrm{sign}(A) = \sum_k \mathrm{sign}(\lambda_k)\mathbf{e_k}\mathbf{f_k}^T.$$

For a diagonalizable matrix A with pure real spectrum, we can express the spectral projector onto eigenvectors corresponding to eigenvalues less than μ by

$$P^{(\mu)} = \sum_{\lambda_k<\mu} P_k = \frac{I - \mathrm{sign}_{\mathrm{Re}}(A-\mu I)}{2}. \tag{7.5}$$

Similarly, for diagonalizable matrices A with pure imaginary spectrum, we define

$$P^{(\mu)} = \sum_{\mathrm{Im}(\lambda_k)<\mu} P_k = \frac{I + i\ \mathrm{sign}_{\mathrm{Im}}(A-\mu i I)}{2}. \tag{7.6}$$

We need an algorithm to compute the sign function of a matrix. Such algorithms have been given by Kenney and Laub [35], and by Beylkin et al. [10]. The algorithm in [10] is iterative and requires a normalization to initialize the iteration. However, the normalization used in [10] does not guarantee convergence for non-self adjoint matrices. The algorithm below computes the sign function of matrices that are diagonalizable with pure real or pure imaginary spectrum, but not necessarily self adjoint.

Theorem 27 *Let A be a diagonalizable matrix with pure real or imaginary spectrum. Let S be the similarity transform such that $A = S^{-1}\mathrm{diag}(\lambda_i)S$ where $\{\lambda_i\}_i$ are the eigenvalues of A. If $\alpha < \frac{1}{\mathrm{cond}(S)\|A\|_2}$, then the recursion*

$$\begin{aligned} A_0 &= \alpha A \\ A_{k+1} &= \begin{cases} \frac{3A_k - A_k^3}{2} & \textit{if } A \textit{ has real spectrum} \\ \frac{3A_k + A_k^3}{2} & \textit{if } A \textit{ has imaginary spectrum} \end{cases} \end{aligned} \tag{7.7}$$

converges to sign(A).

Proof. We first observe that if A_0 is diagonal, then A_k is diagonal for all $k = 1, 2, \ldots$. Hence, since A is diagonalizable, it suffices to establish convergence in the diagonal basis. Let us consider the real case. Consider the scalar recursion $x_{k+1} = \frac{3x_k - x_k^3}{2}$. This recursion has the super stable fixed points $x = \pm 1$, and the unstable fixed point $x = 0$. Since the ratio

$$\frac{x_{k+1}}{x_k} = \frac{3}{2} - \frac{x_k^2}{2} \geq 1,$$

we have that $x_k \to 1$ if $x_0 \in (0,1)$ and $x_k \to -1$ if $x_0 \in (-1,0)$. Since we consider the diagonal basis, it remains to show that $|\alpha\lambda_i| < 1$ for all eigenvalues λ_i of A. From the assumption on α it follows that

$$\|\alpha \text{ diag}(\lambda_i)\|_2 = \alpha\|SAS^{-1}\|_2 \leq \alpha\|S\|_2\|A\|_2\|S^{-1}\|_2 = \alpha \text{ cond(S)}\|A\|_2 < 1.$$

The proof for the case of pure imaginary spectrum is analogous. □

The theorem above is shown in [10] for self-adjoint matrices where it is also shown that the number of iterations needed for (7.7) to converge to accuracy ϵ is $O(\log_2(\text{cond}(A)) + O(\log_2(\log_2(1/\epsilon)))$.

Using (7.7) we can now compute spectral projectors by a sequence of matrix-matrix multiplications. We need a fast algorithm for computing matrix products and for this purpose, we use the separated representation from Section 5.1 for multi-dimensional operators, and the PLR representation from Section 5.2 for one-dimensional matrices to control the complexity of the algorithm.

Let us justify why the PLR representation is appropriate for representing spectral projectors. Consider a self-adjoint operator L where the eigenfunctions are orthonormal polynomials. We use the Christoffel-Darboux formula (see Section 5.2.1) to see that the spectral projectors for L take the form $P = D_1AD_2$ where D_1 and D_2 are diagonal operators and $A = \frac{1}{x-y}$. As demonstrated in Section 5.2.1 and Section 5.2.2, the PLR representation performs well for this type of operators.

7.3 A numerical scheme for wave propagation on space-like surfaces

In this section we consider the problem (7.2) of wave propagation on space-like surfaces. We relate this equation to the corresponding equation in the time domain in Appendix D.2. We solve (7.2) for zero boundary conditions and y-independent coefficients. The case with y-dependent coefficients requires solution of an integral equation and will be studied elsewhere. We formulate the equation as a first order system in y and use the spectral projectors from the previous section along with the tools from this thesis to construct a fast algorithm for solving wave propagation problems on space-like surfaces.

7.3.1 Wave propagation on space-like surfaces as a first order system

Let us convert (7.2) to a first order system in time. Once the equation is given in the form $w_y = Lw$, we can write this equation as an integral equation which we can solve by fixed point iteration, or, if L is y-independent, by computing the matrix exponential e^{yL}.

By introducing the function v, we write (7.2) as

$$\begin{bmatrix} v \\ u \end{bmatrix}_y = \begin{bmatrix} 0 & A \\ I & 0 \end{bmatrix} \begin{bmatrix} v \\ u \end{bmatrix} \equiv L \begin{bmatrix} v \\ u \end{bmatrix} \tag{7.8}$$

where I denotes the identity operator and $A \equiv -\frac{\partial^2}{\partial x^2} - \kappa\omega^2(1+f)$. In Appendix C.2 it is shown that the eigenvalues λ of L are of the form $\lambda = \sqrt{\mu}$ where μ is an eigenvalue of A. Hence, if A is negative definite, then L has a pure imaginary spectrum. If we define the projector $P = (I - \text{sign}(A))/2$ according to (7.6), then the solution to the equation

$$\begin{bmatrix} v \\ u \end{bmatrix}_y = \begin{bmatrix} 0 & PAP \\ I & 0 \end{bmatrix} \begin{bmatrix} v \\ u \end{bmatrix} \equiv L_p \begin{bmatrix} v \\ u \end{bmatrix} \tag{7.9}$$

is stable according to Theorem 25.

7.3.2 Numerical results

In this section we construct an algorithm to solve (7.2) for zero boundary conditions and y-independent compressibility coefficients. The algorithm can be generalized for non-zero boundary conditions by adding a forcing term to the equation. By formulating the equation as an integral equation which we solve iteratively, we can solve problems with y-dependent coefficients.

We provide two numerical examples. In the first example, we let $f = 0$ which gives us the Helmholtz equation as an initial value problem. We solve this equation numerically, and compare the solution to the exact solution. In the final example, we consider variable coefficients. This problem cannot be solved analytically, but we display the solution and verify stability of the algorithm.

The following algorithm solves (7.2) numerically in two dimensions for zero boundary conditions and y-independent coefficients.

Algorithm: Numerical scheme for wave propagation on space-like surfaces

1. Write (7.2) as a first order system in time according to (7.9).
2. Construct the derivative matrix $L_0 = DD_0$ using the algorithm in Section 4.3, but without the projection step (see remark below).
3. Compute $A = -L_0 - \kappa\omega^2(1+f)I$ where I denotes the identity operator.
4. Construct the spectral projector $P = (I - sign(A))/2$ according to Theorem 27.
5. Construct the spatial operator
$$L = \begin{bmatrix} 0 & PAP \\ I & 0 \end{bmatrix}.$$
6. Select the step size Δy (see discussion in Section 6.3.1 on how to choose the step size), and compute the matrix exponential $e^{\Delta y L}$ using the scaling and squaring algorithm [29].
7. **For** $k = 1 : N_y$

```
        $\mathbf{u}(y_k) = e^{\Delta y L}\mathbf{u}(y_{k-1})$
    end
```

If the matrices P, A, or PAP are large, we use the PLR representation described in Section 5.2 to speed up the computations in Step 3, 5, and 6 of the algorithm. In this algorithm, it is essential not to project the second derivative matrix L_0. In the algorithm in Section 4.3, the projection step maps highly oscillatory eigenfunctions corresponding to eigenvalues larger than the bandwidth c, to eigenfunctions with the eigenvalue zero. However, in this algorithm the computation of the matrix $A = -L_0 - \kappa\omega^2(1+f)I$ will shift these eigenvalues to non-zero eigenvalues, and the oscillatory eigenfunctions may now cause instabilities when solving (7.2). By not projecting the matrix L_0, the highly oscillatory eigenfunctions are still associated with large eigenvalues. However, in the algorithm above, these eigenfunctions will be "removed" in step 4, as long as $|\omega| \leq c$.

Numerical results for constant coefficients

Let us solve
$$\begin{cases} u_{xx} + u_{yy} + \omega^2 u = 0, \ (x,y)\in(-1,1)\times(-1,1) \\ u(x,-1) = \sin\left(\frac{\pi(x+1)}{2}\right) + \sin\left(b(x+1)\right) \\ u_y(x,-1) = 0 \\ u(\pm 1, y) = 0 \end{cases}, \quad (7.10)$$

where $b = k\pi/2$ for some integer $k \geq 1$ and $\omega = \sqrt{b^2 + (23\pi)^2}$. The solution is given by

$$u(x,y) = \sin\left(\frac{\pi(x+1)}{2}\right)\cos\left(\sqrt{\omega^2 - \frac{\pi^2}{4}}(y+1)\right)$$

$$+ \sin\left(b(x+1)\right)\cos(\sqrt{\omega^2 - b^2}(y+1)).$$

We note that this solution contains both low frequency (the first term) and high frequency (the second term) modes. We measure the error using the relative max norm, that is, if $\tilde{u}$ approximates the exact solution u, then

$$error = \frac{\|u - \tilde{u}\|_\infty}{\|u\|_\infty}.$$

We construct quadrature nodes and weights for the bandwidth $c = 23\pi$ which corresponds to an oversampling factor of approximately 1.4 for periodic functions. We set the accuracy in the construction to $\epsilon = 10^{-7}$. We use the step size $\Delta y = \frac{1}{32}$ corresponding to approximately 3 samples per wavelength in the y-direction. The result is shown in Figure 7.1. We note

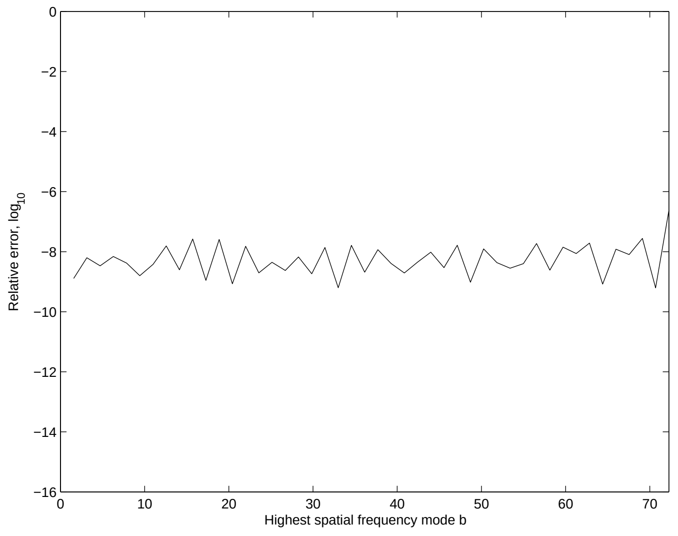


Figure 7.1: Relative error ($\log_{10}$) in the max-norm for approximating the solution to (7.10) for $b = \frac{\pi}{2}, \pi, \ldots, 23\pi$.

that the error is close to the accuracy chosen for the quadrature, and also that the error is essentially uniform within the bandwidth.

Numerical results for variable coefficients

Let us solve the Helmholtz equation as an initial value problem for the case with variable background in the x-direction. Consider the equation

$$\begin{cases} u_{xx} + u_{yy} + \kappa(x)(27\pi)^2 u = 0, \ (x, y) \in (-1, 1) \times (-1, 7) \\ u(x, 0) = g(x) \\ u_y(x, 0) = 0 \\ u(\pm 1, y) = 0 \end{cases} \tag{7.11}$$

which simulates the pressure in a wave guide with "soft" (zero pressure) boundary conditions. We solve this equation using the numerical scheme given earlier in this section using 128 quadrature nodes for the bandlimited functions with the bandwidth $c = 54\pi$ and the accuracy $\epsilon = 10^{-7}$. Since we propagate using the exponential, we are free to use any step size without causing instabilities. However, for this experiment we choose the step size $\Delta y = \frac{1}{128}$, corresponding to approximately 10 samples per wavelength for easier visualization of the resulting wave field.

For the first experiment, we choose a constant background ($\kappa_1(x) = 1$) and the initial pulse e^{-1000x^2}. We display the resulting wave field in the left image in Figure 7.3. The initial condition can be compared to a wave entering the domain through a "smooth" slit centered at $(x, y) = (0, -1)$. The wave diffracts and then reflect at the boundaries.

For the following two experiments, we choose the background coefficients

$$\kappa_2(x) = \frac{1}{\sqrt{1 + \frac{\sin \pi(x+1)}{2}}}$$

and

$$\kappa_3(x) = \frac{1}{\sqrt{\left(1 + \frac{\sin \pi(x+1)}{2}\right)(1 - 0.9e^{-100x^2})}},$$

respectively. The profile of the compressibility and the velocity $\frac{1}{\sqrt{\kappa(x)}}$ are given in Figure 7.2 We use the initial condition $g(x) = e^{-1000(x+0.5)^2}$ for these two experiments and show the results in Figure 7.3. We note that the waves travel faster in the left part of the domain as expected, causing the rays to bend. We also note the dark line in the last image due to the sharp scatterer centered along $x = 0$.

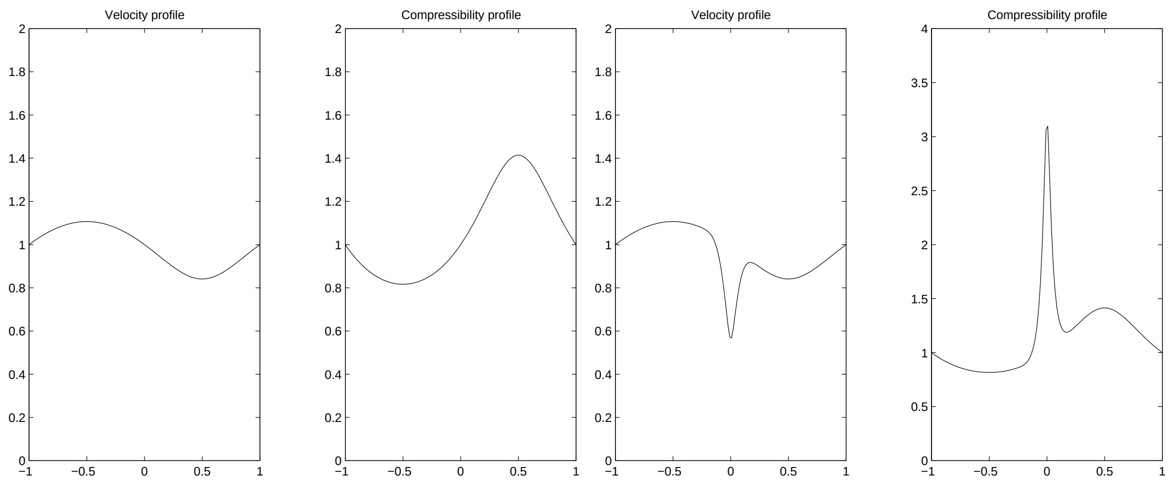


Figure 7.2: Velocity and compressibility profiles for $\kappa_2(x)$ (top), and $\kappa_3(x)$ (bottom).

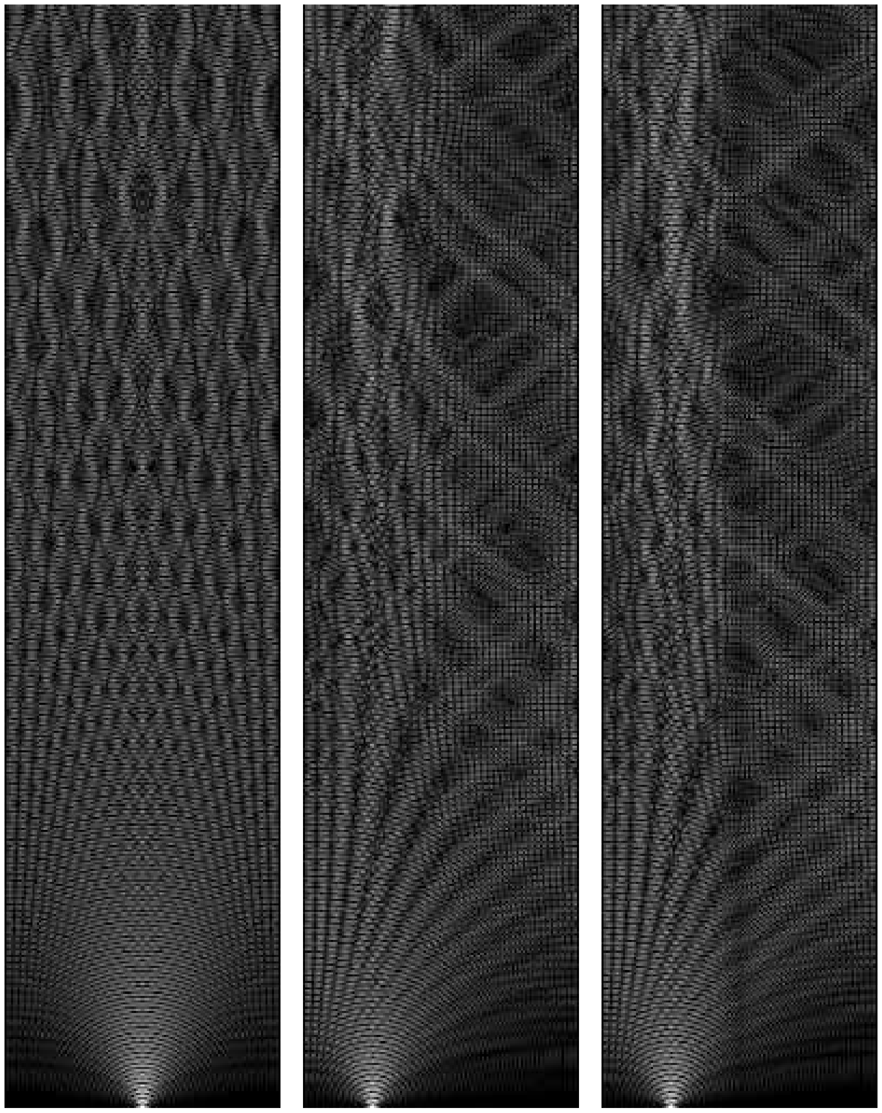


Figure 7.3: Absolute value of the wave field from the solution to (7.11) for the three compressibilities $\kappa_1(x)$ (top), $\kappa_2(x)$ (center), and $\kappa_3(x)$ (bottom).

Chapter 8

A fast reconstruction algorithm for electron microscopy

In this chapter we summarize the paper "A fast reconstruction algorithm for electron microscopy" by Beylkin, Mastronarde, and Sandberg [6]. We provide a version of the paper in Appendix E.

In the paper we consider the problem of three-dimensional tomographic reconstruction of the density of a biological specimen using transmission electron microscopy. The three-dimensional problem is solved as a sequence of two-dimensional problems. The specimen is illuminated by an electron beam and the intensity of the beam is recorded after transmission. We refer to such a measurement as a projection. The projections are recorded for a range of angles typically between $\pm 70^o$ (due to physical limitations). We model the decay of intensity of the beam by line integrals and, therefore, interpret the collected data as the (discretized) Radon transform of the density.

The problem of reconstructing an object by measuring projections has a rich history and many applications. For example, the x-ray tomography, radio astronomy, as well as seismic processing are using results of the basic inversion technique first considered by Radon [45]. The Radon inversion formula was re-discovered by Cormack [17] for x-ray tomography, and by Bracewell [12] for radio astronomy. For an introductory overview of the subject, see Deans [18]. Reconstruction algorithms for electron microscopy imaging of biological specimens have been described by DeRosier and Klug [19] via a Fourier based method, and by Gilbert [28] via direct summation.

The goal of the paper is to construct a fast reconstruction algorithm that produces the same results as the direct summation technique [28] traditionally used for electron microscopy tomography. Our goal is to construct an algorithm that scales as $O(N^2 \log N)$ compared to $O(N^3)$ for the direct summation technique.

The well-known Fourier slice theorem relates projection data to the Fourier transform of the image. The collected projection data corresponds to samples on a polar grid in the

Fourier space where the polar angles are not necessarily equally spaced. The standard two-dimensional Fast Fourier Transform (FFT) requires sampling on an equally spaced rectangular grid and, hence, fast Fourier reconstruction methods require some interpolation scheme in the Fourier space. Such Fourier based techniques have previously been proposed by, e.g., Lanzavecchia and Bellon [41]. They used an improvement of the moving window Shannon technique (Lanzavecchia and Bellon [40]) to interpolate the data on a polar grid to an equally spaced square grid in Fourier space. Our approach is different, and involves the Unequally Spaced Fast Fourier Transform introduced by Dutt and Rokhlin [21], and by Beylkin [4].

The algorithm is described and evaluated in detail in Appendix E, using data sets collected by The Boulder Laboratory for 3-D Electron Microscopy of Cells at University of Colorado at Boulder. The algorithm has been incorporated into the IMOD software package [32]. We give an overview of Appendix E in the following section.

8.1 Preliminaries

In Section E.2 of the paper, we formulate the problem. The notation and experimental set-up is described schematically in Figure E.1. The goal is to estimate the density $g(x, z)$ of a two-dimensional slice of the specimen at an M-by-N equally spaced grid from the measurements of transmitted electron beams. We measure the intensity of the electron beam after transmission through the specimen at the (equally spaced) points $t_1, \ldots, t_M$, and repeat the measurements for the (not necessarily equally spaced) angles $\theta_l,\ l = 1, \ldots, N_\theta$. The measurement data is given by the array $R_{\theta_l}(t_k)$ which is assumed to measure line integrals of the density, that is,

$$R_{\theta_l}(t_k) \equiv \int_{C_{t_k,\theta}} g(x, z)\ ds$$

where

$$C_{t,\theta} = \{\ (x, z) \in \mathbb{R}^2 \mid t = x \cos\theta + z \sin\theta\ \} \tag{8.1}$$

(see Figure E.1).

In Section E.2.2 and Section E.2.3 we review the reconstruction formula known as direct summation. The resulting sum can be shown to approximate the inverse of the Radon transform of the density g. The reconstruction formula takes the form

$$g(x_m, z_n) = \sum_{l=1}^{N_\theta} w_l\ [\rho * R_{\theta_l}](t(x_m, z_n)) \tag{8.2}$$

where w_l are scalar weights, $t(x_m, z_n)$ are given by (E.1), and ρ is a bandlimited convolution operator. This reconstruction technique is often referred to as filtered back projection or direct summation. The direct summation method therefore consists of two steps; a convolution step (which can be computed efficiently using the FFT), and a summation step which dominates the computational cost. In the summation step we sum over N_θ terms $M \times N$ times for the total computational cost of $O(N_\theta MN)$. From the definition of t in (8.1), we note than in general we do not have measurement for arbitrary values of $t(x_m, z_n)$ but we must interpolate the measured data $R_{\theta_l}(t_k)$ to estimate $R_{\theta_l}(t(x_m, z_n))$. The interpolation is typically linear.

8.2 Inversion in the Fourier domain

In Section E.3 we derive a reconstruction formula in the Fourier domain that matches the direct summation formula (8.2) exactly. For a fixed $z = z_n$, we Fourier transform (8.2) with respect to x. We see that $\hat{g}_n(\omega) \equiv \int_{-\infty}^{\infty} g(x, z_n) e^{2\pi i x\omega}\, dx$ takes the form

$$\hat{g}_n(\omega) = \sum_{l=1}^{N_\theta} v_l(\omega) e^{-2\pi i \xi_l(\omega) z_n}$$

where $\xi_l(\omega) = \omega \tan\theta_l$. To derive the expression for $v_l(\omega)$, we must interpolate the measurement data $R_{\theta_l}(t_k)$ in order to have an expression for $R_{\theta_l}(t)$ for any t. We observe that piecewise polynomial interpolation using B-splines can represented by a convolution kernel with a closed form expression in the Fourier domain which appears as a factor in the expression for $v_l(\omega)$, namely,

$$v_l(\omega) = \frac{w_l}{\cos\theta_l} e^{-2\pi i x_s \frac{\omega}{\cos\theta_l}} \hat{\rho}(\frac{\omega}{\cos\theta_l}) \hat{\beta}(\frac{\omega}{\cos\theta_l}) \sum_{m=0}^{M-1} r_{ml} e^{2\pi i m \frac{\omega}{\cos\theta_l}}, \tag{8.3}$$

where $\hat{\beta}$ denotes the Fourier transform of the B-spline kernel. Using the Fourier-representation of the B-spline kernel, we can obtain any order of piecewise polynomial interpolation at no extra computational cost.

8.3 Implementation

In Section E.4, we discretize the expression for $\hat{g}_{nk} \equiv \hat{g}_n(\omega_k)$ for the frequencies $\omega_1, \dots, \omega_{M_f}$ where $M_f \geq M$. The expression for $\hat{g}_{nk}$ takes the form

$$\hat{g}_{nk} = c_k \sum_{l=1}^{N_\theta} v_l(\frac{k}{M_f}) e^{-2\pi i \xi_l(\frac{k}{M_f}) z_n} \tag{8.4}$$

where c_k are scalars, and v_l is given by (8.3). We can now estimate $g(x_m, z_n)$ by the following three steps:

1. For each $l = 1, \dots, N_\theta$ and $k = 1, \dots, M_f$, compute $v_l(\omega_k)$ defined by (8.3).
2. For each $n = 1, \dots, N$ and $k = 1, \dots, M_f$, compute $\hat{g}_{nk}$ defined by (8.4).
3. For each $n = 1, \dots, N$, compute $g(x_m, z_n)$ by applying the inverse FFT of $\hat{g}_{nk}$ with respect to k.

The key to a fast algorithm is that the sums in (8.4) and (8.3) can be written on the form

$$\hat{u}_n = \sum_{k=1}^{M} u_k e^{\pm 2\pi i \xi_k n}, \;\; n = -\frac{N}{2}, -\frac{N}{2}+1, \; \dots, \frac{N}{2}-1$$

and

$$\hat{u}(\xi_k) = \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} u_n e^{\pm 2\pi i n \xi_k}, \;\; k = 1, 2, \dots, M,$$

respectively, for a given set of real points $\{\xi_k\}_{k=0}^{M}$, where $|\xi_k| < 1/2$ for each k. We note that M may be different from N. Such sums can be computed efficiently using the Unequally Spaced Fast Fourier Transform (USFFT) in [21] and [4]. We choose the algorithm in [4] which gives the final algorithm the computational cost $O(N_\theta M \log M) + O(N_\theta M_f \log M_f)$.

In Section E.4.3 we provide an expression for selecting M_f which has to be sufficiently large to avoid aliasing artifacts. Typically, M_f is approximately 1.5-2 times larges than M. In Section E.4.5 we show how to incorporate any (odd) order of piecewise polynomial interpolation into the algorithm. In the direct summation method, higher order interpolation can be cumbersome to implement and increases the computational cost. With our Fourier-based approach, such interpolation is trivial to implement. In fact, higher order interpolation can even speed up the algorithm, since the interpolation effectively low-pass filter the data, thus reducing the number of frequencies ω_k that contribute significantly to the final image.

8.4 Results

In Section 2.7, the algorithm is tested on data sets collected by The Boulder Laboratory for 3-D Electron Microscopy of Cells at University of Colorado at Boulder. The new method and the direct summation method are shown to produce visually identical reconstructions. The reconstructions are also evaluated using the Fourier Ring Correlation (FRC) test (see

Saxton and Baumeister [49]). The FRC test evaluates the performance in the presence of noise, and the proposed algorithm and the direct summation method are shown to give practically identical results according to the FRC test.

Speed comparisons of the two methods are made on three computer architectures; Athlon MP, Pentium 4, and SGI R12000. The Fourier-based method is demonstrated to be 1.5-2.5 faster than the direct summation method for typical data sizes. The relative speed gain is even higher for larger sizes. The gain is particularly large when fixing the image size and varying the number of projections.

References

[1] B. Alpert, G. Beylkin, D. Gines, and L. Vozovoi. Adaptive solution of partial differential equations in multiwavelet bases. *J. Comput. Phys.*, 182(1):149–190, 2002. Univ. of Colorado, APPM preprint #409, June 1999; `ftp://amath.colorado.edu/pub/wavelets/papers/mwa.pdf`.

[2] B. Alpert, L. Greengard, and T. Hagstrom. An integral evolution formula for the wave equation. *J. Comput. Phys.*, 162:536–543, 2000.

[3] J. Bazer and R. Burridge. Energy partition in the reflection and refraction of plane waves. *SIAM J. Appl. Math.*, 34:78–92, 1978.

[4] G. Beylkin. On the fast Fourier transform of functions with singularities. *Appl. Comput. Harmon. Anal.*, 2(4):363–381, 1995.

[5] G. Beylkin and J. M. Keiser. On the adaptive numerical solution of nonlinear partial differential equations in wavelet bases. *J. Comput. Phys.*, 132:233–259, 1997. Univ. of Colorado, APPM preprint #262, 1995.

[6] G. Beylkin, D. Mastronarde, and K. Sandberg. A fast reconstruction algorithm for electron microscopy tomography. *Journal of structural biology*, 2003. Submitted.

[7] G. Beylkin and M. J. Mohlenkamp. Numerical operator calculus in higher dimensions. *Proc. Natl. Acad. Sci. USA*, 99(16):10246–10251, August 2002. Univ. of Colorado, APPM preprint #476, August 2001; `http://www.pnas.org/cgi/content/abstract/112329799v1`.

[8] G. Beylkin and M. J. Mohlenkamp. Numerical analysis for the multiparticle Schrödinger equation. *In preparation*, 2003.

[9] G. Beylkin and L. Monzón. On generalized Gaussian quadratures for exponentials and their applications. *Appl. Comput. Harmon. Anal.*, 12(3):332–373, 2002.

[10] Gregory Beylkin, Nicholas Coult, and Martin J. Mohlenkamp. Fast spectral projection algorithms for density-matrix computations. *J. Comput. Phys.*, 152(1):32–54, June 1999. Univ. of Colorado, APPM preprint #392, August 1998. `ftp://amath.colorado.edu/pub/wavelets/papers/spectr-proj.ps.Z`.

[11] C. Bouwkamp. On spheroidal wave functions of order zero. *J. Math. Phys.*, 26:79–92, 1947.

[12] R.N. Bracewell. Strip integration in astronomy. *Aust. J. Phys.*, 9:198–217, 1956.

[13] L. M. Brekhovskikh and O.A. Godin. *Acoustics of layered media I.* Springer Verlag, 1990.

[14] Y. Chen. Inverse scattering via Heisenberg's uncertainty principle. *Inverse problems*, 13:253–282, 1997.

[15] D. Colton and R. Kress. *Inverse acoustic and electromagnetic scattering theory.* Springer-Verlag, 1992.

[16] J.W. Cooley and J.W. Tukey. An algorithm for the machine computation of complex Fourier series. *Math. Comp.*, 19:297–301, 1965.

[17] A.M. Cormack. Representation of a function by its line integrals, with some radiological applications. *J. Appl. Phys.*, 34:2722–2727, 1963.

[18] S. Deans. *The Radon transform and some of its applications.* Krieger Publishing Company, 1993.

[19] D.J. DeRosier and A. Klug. Reconstruction of three dimensional structures from electron micrographs. *Nature*, 217:130–134, 1968.

[20] D.R. Durran. *Numerical methods for wave equations in geophysical fluid dynamics.* Springer-Verlag, 1999.

[21] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM J. Sci. Comput.*, 14(6):1368–1393, 1993.

[22] J. Frank (Ed.). *Electron Tomography.* Plenum, 1992.

[23] L.C. Evans. *Partial differential equations.* American Mathematical Society, 1998.

[24] C. Flammer. *Spheroidal wave functions.* Stanford Univeristy Press, 1957.

[25] B. Fornberg. *A practical guide to pseudospectral methods.* Cambridge Univeristy Press, 1995.

[26] F.R. Gantmacher. *The theory of matrices.* Chelsea Pub. Co., 1959.

[27] A.K. Gautesen and M. de Hoop. Uniform asymptotic of the square-root Helmholtz operator and the one-way wave propagator. *SIAM J. Appl. Math.*, 63:777–800, 2003.

[28] P. Gilbert. The reconstruction of a three-dimensional structure from projections and its applications to electron microscopy II. Direct methods. *Proc. R. Soc. Lond. B.*, pages 89–102, 1972.

[29] G. Golub and C. Van Loan. *Matrix computations.* Johns Hopkins Unviversity Press, 3:rd edition, 1996.

[30] L. Greengard. Spectral integration and two-point boundary value problems. *SINUM*, 28(4):1071–1080, 1991.

[31] T. Hrycak and V. Rokhlin. An improved fast multipole algorithm for potential fields. Technical report, Yale Univ., 1995. YALEU/DCS/RR-1089.

[32] IMOD. The IMOD home page at the Boulder laboratory for 3-D electron microscopy of cells. 2003. `http://bio3d.colorado.edu/imod`.

[33] A. Iserles. *A first course in the numerical analysis of differential equations.* Cambridge University Press, 1996.

[34] P. Jones, J. Ma, and V. Rokhlin. A fast direct algorithm for the solution of the Laplace equation on regions with fractal boundaries. *J. Comput. Phys.*, 113(1), July 1994.

[35] C.S. Kenney and A.J. Laub. The matrix sign function. *IEEE Trans. Automat. Control*, 40(8):1330–1348, 1995.

[36] J.R. Kremer, D.N. Mastronarde, and J.R. McIntosh. Computer visualization of three-dimensional image data using IMOD. *J. of Structural Biology*, 116:71–76, 1996.

[37] M.S. Ladinsky, D.N. Mastronarde, J.R. McIntosh, K.E. Howell, and L.A. Staehlin. Golgi structure in three dimensions: functional insights from the NRK cell. *J. of Cell Biology*, 144:1135–1149, 1999.

[38] H. J. Landau and H. O. Pollak. Prolate spheroidal wave functions, Fourier analysis and uncertainty II. *Bell System Tech. J.*, 40:65–84, 1961.

[39] H. J. Landau and H. O. Pollak. Prolate spheroidal wave functions, Fourier analysis and uncertainty III. *Bell System Tech. J.*, 41:1295–1336, 1962.

[40] S. Lanzavecchia and P.L. Bellon. A moving window Shannon reconstruction for image interpolation. *J. Vis. Commun. Image Repres.*, 5:255–264, 1994.

[41] S. Lanzavecchia and P.L. Bellon. Fast computation of 3D Radon transform via a direct Fourier method. *Bioinformatics*, 14(2):212–216, 1998.

[42] D.N. Mastronarde. Dual-axis tomography: An approach with alignment methods that preserve resolution. *J. of Structural Biology*, 120:343–352, 1997.

[43] Y. Meyer. *Wavelets and Operators.* Cambridge Univ. Press, 1992.

[44] F. Natterer and Frank Wubbeling. A propagation-backpropagation method for ultrasound tomography. *Inverse problems*, 11:1225–1232, 1995.

[45] J. Radon. Über the bestimmung von funktionen durch ihre integralwerte längs gewisser mannigfaltigkeiten. *Berichte Sächsische Akademie der Wissenschaften Leipzig Math.-Phys. Kl.*, 69:262–267, 1917.

[46] L. Reimer. *Transmission Electron Microscopy: Physics of Image Formation and Analysis*, volume 36. Springer Series in Optical Sciences, 4:th edition, 1997.

[47] T.J. Rivlin. *Chebyshev polynomials.* John Wiley & sons, 2:nd edition, 1990.

[48] V. Rokhlin and N. Yarvin. A generalized one-dimensional fast multipole method with application to filtering of spherical harmonics. Technical report, Yale Univ., 1998. YALEU/DCS/RR-1142.

[49] W.O. Saxton and W. Baumeister. The correlation averaging of a regularly arranged bacterial cell envelope protein. *Journal of Microscopy*, 127:127–138, 1982.

[50] I. J. Schoenberg. *Cardinal spline interpolation.* SIAM, Philadelphia, Pa., 1973. Conference Board of the Mathematical Sciences Regional Conference Series in Applied Mathematics, No. 12.

[51] D. Slepian. Prolate spheroidal wave functions, Fourier analysis and uncertainty IV. Extensions to many dimensions; generalized prolate spheroidal functions. *Bell System Tech. J.*, 43:3009–3057, 1964.

[52] D. Slepian. Prolate spheroidal wave functions, Fourier analysis and uncertainty V. The discrete case. *Bell System Tech. J.*, 57:1371–1430, 1978.

[53] D. Slepian. Some comments on Fourier analysis, uncertainty and modeling. *SIAM review*, 25(3):379–393, 1983.

[54] D. Slepian and H. O. Pollak. Prolate spheroidal wave functions, Fourier analysis and uncertainty I. *Bell System Tech. J.*, 40:43–63, 1961.

[55] G. B. Whitham. *Linear and Nonlinear Waves.* Wiley-Interscience, New York, 1974.

[56] H. Xiao, V. Rokhlin, and N. Yarvin. Prolate spheroidal wavefunctions, quadrature and interpolation. *Inverse Problems*, 17(4):805–838, 2001.

[57] K. Yoshida. *Functional analysis.* Springer, 1995. Reprint of the 6:th edition.

Appendix A

Proof of Corollary 10

We observe that

$$\int_{-1}^{1} e^{ictx}\ dt = \frac{1}{c}v(x),$$

where $v(x) = \int_{-c}^{c} e^{i\omega x}\ d\omega$. By the variable substitution $\tau = \frac{\omega}{2c}$, we have that

$$v(x) = 2c\int_{-\frac{1}{2}}^{\frac{1}{2}} e^{i2c\tau x}\ d\tau. \tag{A.1}$$

Let us introduce the parameter $\gamma \in (0,1)$ and choose an integer N such that $N \geq \frac{2c}{\gamma\pi}$. By defining $\nu = \frac{c}{\gamma N\pi} \leq \frac{1}{2}$ and substitute $t = 2\nu\tau$ in (A.1), we have that

$$v(x) = 2c\int_{-\frac{1}{2}}^{\frac{1}{2}} e^{i\pi 2\nu\tau Nax}\ d\tau = \frac{c}{\nu}\int_{-\nu}^{\nu} e^{i\pi tN\gamma x}\ dt.$$

By introducing $y = N\gamma x$ and the weight function

$$\sigma(t) = \begin{cases} c/\nu, & t\in[-\nu,\nu] \\ 0, & t \notin [-\nu,\nu] \end{cases},$$

we have that

$$v(x) = \int_{-\nu}^{\nu} \sigma(t)e^{i\pi ty}\ dt,\ y = N\gamma x. \tag{A.2}$$

We observe that $|y| \leq \gamma N$ and that $\sigma(t)$ is supported on $[-\nu,\nu] \subseteq [-\frac{1}{2},\frac{1}{2}]$. Hence, the assumptions for using Theorem 9 are fulfilled and accordingly there exist constants v_k and t_k such that $|t_k| < \nu$ and

$$\left|\int_{-\nu}^{\nu} \sigma(t)e^{i\pi ty}\ dt - \sum_{k=1}^{N} v_k e^{i\pi t_k y}\right| < \epsilon.$$

By introducing $w_k = v_k/c$ and $\theta_k = \pi t_k/c$ we have that

$$\left| \int_{-1}^{1} e^{ictx} \, dt - \sum_{k=1}^{N} w_k e^{ic\theta_k x} \right| < \epsilon/c.$$

We note that $|\theta_k| \leq \pi\nu/c = \frac{1}{\gamma N}$ so by choosing N sufficiently large, $|\theta_k| < 1$.

According to Theorem 9, the error for approximating $v(x)$ is bounded by

$$\left| \int_{-\nu}^{\nu} \sigma(t) e^{i\pi t y} \, dt - \sum_{k=1}^{N} v_k e^{i\pi t_k y} \right| < \epsilon \leq$$

$$2\|\sigma\|_1 \left(3 \left(\frac{c}{\gamma N \pi} \right)^{2m} + \frac{2}{2 + (2+\sqrt{3})^N + (2-\sqrt{3})^N} + \frac{2d_m}{1 - e^{-\alpha_m}} e^{-\alpha_m(1-\gamma)N} \right).$$

We conclude the proof by observing that $\|\sigma\|_1 = 2c$.

Appendix B

Algorithms for low rank representations of operators

B.1 Rank reduction

In order to reduce the rank of a separated representation on the form $\sum_i \sigma_i e_i f_i^*$ and obtain an almost orthonormal separated representation, we need a way to "re-orthogonalize" a given separated representation. Such an algorithm is given for operators in an arbitrary number of dimensions in [7]. In this thesis, we present a simpler algorithm that works for two dimensional problems. We first provide a heuristic description of the rank reduction procedure and give a more detailed algorithm below.

Consider a sum of the form

$$L = \sum_{k=1}^{r} s_k e_k f_k^*$$

where we assume that the vectors $e_k, f_k \in \mathbb{C}^N$ have been normalized using the norm $\|v\|_2 = \sqrt{< v, v >}$ where $< \, , \, >$ denotes the standard dot-product. For two dimensional problems, e_k and f_k correspond to vectors in $\mathbb{C}^{N^2}$. In this case, the norm corresponds to the Frobenius norm for matrices.

Let us first orthogonalize the left factors e_k to obtain a new set of left factors $\tilde{e}_k$ which is an orthonormal set. The scalars s_k and the right factors f_k are simultaneously re-computed such that the new decomposition still equals the matrix L. Throughout this process, we measure the norm of the terms, and truncate terms that fall below a given threshold. This means that the rank usually will reduce during this process. We refer to this step as an orthogonalization sweep. One such sweep will give a separated representation where all the left factors are orthonormal. However, there is no guarantee that also the right factors will be orthogonal. Therefore, we perform another orthogonalization sweep but switch the role for the left and the right factors. This will orthogonalize the right factors, but may change the left factors such that they are no longer orthonormal. By iterating this process by alternating the orthogonalization sweeps for the left and the right factors, the decomposition will converge to the SVD. In practice, we have found that one sweep for each factor is usually sufficient to reduce the rank close to the optimal rank, and produce left and right factors that are almost orthonormal.

The following algorithm describes the rank reduction process in more detail.

Algorithm: Rank reduction of a separated representation

```
1. Normalize $\{e_k\}_k$ and $\{f_k\}_k$ and adjust $\{s_k\}_k$ accordingly. Set $i = 1$.

2. Pivot (put the term with the largest $s_i$ first)

3. **While** $i < r$

        $f_i = s_i f_i$
        **For** $j = i + 1 : r$
            $a_j =< e_i, e_j >$
            $\tilde{e}_j = e_j - a_j e_i$ (This means that $< \tilde{e}_j, e_i >= 0$.)
            $b_j = \|\tilde{e}_j\|_2$
            $f_i = f_i + s_j a_j f_j$
        **end**
        **For** $j = i + 1 : r$
            $\tilde{s} = s_j b_j$
            **If** $\tilde{s} > \epsilon$
                $\tilde{e}_j = \frac{\tilde{e}_j}{b_j}$
                $s_j = \tilde{s}$
            **else**
                $r = r - 1$
            **end**
            $s_i = \|f_i\|$
            **If** $s_i > \epsilon$
                $f_i = \frac{f_i}{s_i}$
            **else**
                $r = r - 1$
            **end**
        **end**

   Pivot (put the term with the largest $s_i$ first)

   i=i+1

   **end**
```

B.2 PLR matrix-vector multiplication

Algorithm: Matrix-vector multiplication for a matrix given by a PLR representation

```
**For** $k = 1 : 2^m$
    Compute the contribution from the diagonal blocks:
    $v((k-1)N_m + 1 : kN_m)) = D_k u((k-1)N_m + 1 : kN_m))$
**end**
**For** $k = 1 : m$ (Loop over the levels.)
    Set $N_k = \frac{N}{2^k}$, $i_1 = 0$, and $i_2 = N_k$.
    **For** $l = 1 : 2^{k-1}$ (Loop over the off-diagonal blocks at level $k$)
        Compute the contribution from the $l$:th upper diagonal block at level $k$:
        $v(i_1 + 1 : i_1 + N_k) = v(i_1 + 1 : i_1 + N_k) + U_l^k u(i_2 + 1 : i_2 + N_k)$
        Compute the contribution from the $l$:th lower diagonal block at level $k$:
        $v(i_2 + 1 : i_2 + N_k) = v(i_2 + 1 : i_2 + N_k) + L_l^k u(i_1 + 1 : i_1 + N_k)$
    **end**
    $i_1 = i_1 + 2N_k$
    $i_2 = i_2 + 2N_k$
**end**
```

B.3 PLR products

Consider the matrix product $C = AB$ where A, B, and C are N-by-N matrices given as level m PLR representations. Let us adopt the notation from Figure 5.2 and introduce the following notation. We refer to a block indexed as U_l^k (or L_l^k) as the l:th upper (lower) diagonal block at level k, and to the diagonal block D_l as the l:th diagonal block. We refer to D_l^k as the l:th diagonal block (counted from the upper left corner) of size $\frac{N}{2^k}$-by-$\frac{N}{2^k}$. For example, in Figure 5.2,

$$D_3^2 = \left[\begin{array}{c|c} D_5 & U_3^3 \\ \hline L_3^3 & D_6 \end{array}\right].$$

In other words, D_l^k is a diagonal block which in itself is a PLR representation.

We use the corresponding notation for the matrices B and C, but mark blocks from these matrices with a tilde ($\tilde{}$), and a hat ($\hat{}$), respectively. The operator $\texttt{restrict}(F)_l^{U,k}$ restricts a matrix to the portion that covers the l:th upper diagonal block at level k. The operator $\texttt{restrict}(F)_l^{L,k}$ restricts a matrix to the portion that covers the l:th lower diagonal block at level k.

We can now compute the l:th upper diagonal block of C at level k by using the following algorithm.

1. Compute the following matrices

```
    For $i = k-1 : 1$
        For $j = 1 : 2^{i-1}$
            $F_j^i = U_j^i \tilde{L}_j^i$
            $G_j^i = L_j^i \tilde{U}_j^i$
        end
    end
```

2. $\hat{U}_l^k = D_{2l-1}^k \tilde{U}_l^k + U_l^k \tilde{D}_{2l}^k$

3. Set $j = l$

4. **For** $i = k-1 : 1$ (Loop over parent levels)

```
    If $j$ is odd
        $j = \lceil \frac{j}{2} \rceil$
        $\hat{U}_l^k = \hat{U}_l^k + \texttt{restrict}(F_j^i)_l^{U,k}$
    else
        $j = \lceil \frac{j}{2} \rceil$
        $\hat{U}_l^k = \hat{U}_l^k + \texttt{restrict}(G_j^i)_l^{U,k}$
    end
end
```

Here the symbol $\lceil \ \rceil$ refers to rounding to the nearest larger integer. Note that the data for each off-diagonal block in the product should be stored as a separated representation in order to preserve the PLR structure under multiplication.

We note that this algorithm involves three types of block multiplications. The product $D_{2k-1}^k \tilde{U}_l^k$ corresponds to a PLR matrix multiplied by a separated representation. Such a multiplication can be computed in a fast way by applying the algorithm for matrix-vector multiplication in Appendix B.2 on the left factors in the separated representation of $\tilde{U}_l^k$. We can compute the product $U_l^k \tilde{D}_{2k}^k$ in a similar way. Finally, we can compute the product $U_j^i \tilde{L}_l^k$ using (5.10). After each matrix addition, it is essential to apply the rank reduction algorithm in Appendix B.1.

The algorithm for computing lower diagonal and diagonal blocks are similar. The algorithm to compute the l:th lower diagonal block of C at level k is given by the following algorithm.

```
1. $\hat{L}_l^k = L_l^k \tilde{D}_{2l-1}^k + D_{2l}^k \tilde{L}_l^k$

2. Set $j = l$

3. **For** $i = k-1:1$ (Loop over parent levels)

        **If** $j$ is odd
            $j = \lceil \frac{j}{2} \rceil$
            $\hat{U}_l^k = \hat{U}_l^k + \texttt{restrict}(F_j^i)_l^{L,k}$
        **else**
            $j = \lceil \frac{j}{2} \rceil$
            $\hat{U}_l^k = \hat{U}_l^k + \texttt{restrict}(G_j^i)_l^{L,k}$
        **end**

   **end**
```

The computation of the matrices F_j^i and G_j^i is given in the algorithm for the upper diagonal block.

The algorithm to compute the l:th (dense) diagonal block of C for a level m PLR representation is given by the following algorithm.

```
1. $\hat{D}_l = D_l \tilde{D}_l$

2. **If** $l$ is odd

        $\hat{D}_l = \hat{D}_l + U_{\lceil \frac{j}{2} \rceil}^{m-1} \tilde{L}_{\lceil \frac{j}{2} \rceil}^{m-1}$

   **else**

        $\hat{D}_l = \hat{D}_l + L_{\lceil \frac{j}{2} \rceil}^{m-1} \tilde{U}_{\lceil \frac{j}{2} \rceil}^{m-1}$

   **end**

3. Set $j = l$
```

```
4. For $i = m - 1 : 1$ (Loop over parent levels)
        If $j$ is odd
            $j = \lceil \frac{j}{2} \rceil$
            $\hat{U}_l^k = \hat{U}_l^k + \texttt{restrict}(F_j^i)_l^D$
        else
            $j = \lceil \frac{j}{2} \rceil$
            $\hat{U}_l^k = \hat{U}_l^k + \texttt{restrict}(G_j^i)_l^D$
        end
    end
```

The operator $\texttt{restrict}(F)_l^D$ restricts a matrix to the portion that covers the l:th diagonal block of a matrix represented as a level m PLR representation.

Appendix C

The acoustic equation in two dimensions

C.1 The derivative operator in two dimensions on a subdivided domain

Let us look at the derivative operator $\frac{\partial}{\partial x} \otimes I_y$ introduced in Section 6.1. Consider the case where the domain consists of M-by-M subdomains, where each domain has its own set of N basis functions according to Section 3.2 in each direction. Using the notation from Section 3.2 we write functions on such subdivided domain as

$$u(x,y) = \sum_{k=1}^{M}\sum_{l=1}^{M}\sum_{m=1}^{N}\sum_{n=1}^{N} s_{mk}s_{nl}\phi_{mk}(x)\phi_{nl}(y).$$

From the definition of $\frac{\partial}{\partial x} \otimes I_y$ we have that

$$\begin{aligned}\left[\frac{\partial}{\partial x} \otimes I_y\right] u(x,y) &= \sum_{l=1}^{M}\sum_{n=1}^{N} s_{nl}\left(\sum_{k=1}^{M}\sum_{m=1}^{N} s_{mk}\phi'_{mk}(x)\right)\phi_{nl}(y) \\ &\equiv g(x)\sum_{l=1}^{M}\sum_{n=1}^{N} s_{nl}\phi_{nl}(y).\end{aligned}$$

If we use the construction of the derivative matrix in one dimension from Section 3.2 with the coupling parameters $a = b = 1/2$, the heuristic argument in Section 3.2.5 gives that $g(x)$ is continuous. Hence, $\left[\frac{\partial}{\partial x} \otimes I_y\right] u(x,y)$ is continuous across the vertical interfaces for each y.

C.2 Construction of derivative matrices for the acoustic equation

In order to decrease the norm of the matrix representing the spatial operator L in the acoustic equation (6.2), we project the operator. The reasoning similar to what we do for the second derivative in Section 4.3.

Consider the block matrix

$$A = \left[\begin{array}{c|c} 0 & A_{12} \\ \hline A_{21} & 0 \end{array}\right] \tag{C.1}$$

where each block is an $(N \times N)$ matrix. Let I_N and I_{2N} denote an $(N \times N)$ and $(2N \times 2N)$ identity matrix, respectively, and consider

$$\det(A - \lambda I_{2N}) = \det \left[\begin{array}{c|c} -\lambda I_n & A_{12} \\ \hline A_{21} & -\lambda I_N \end{array}\right].$$

Since the diagonal blocks commute with the off-diagonal blocks we can use the identity

$$\det(A - \lambda I_{2N}) = \det(\lambda^2 I_N - A_{21}A_{12})$$

(see, e.g., Gantmacher [26]). Let $\sigma(\ )$ denote the spectrum of a matrix. Then

$$\det(\lambda^2 I_n - A_{21}A_{12}) = \prod_{\mu_k \in \sigma(\lambda^2 I - A_{21}A_{12})} \mu_k.$$

Since each eigenvalue μ_k of $\lambda^2 I - A_{21}A_{12}$ equals an eigenvalue of $-A_{21}A_{12}$ shifted by λ^2, we have that

$$\prod_{\mu_k \in \sigma(\lambda^2 I - A_{21}A_{12})} \mu_k = \prod_{\nu_k \in \sigma(A_{21}A_{12})} (\lambda^2 - \nu_k).$$

Therefore,

$$\det(A - \lambda I_{2n}) = \prod_{\nu_k \in \sigma(A_{21}A_{12})} (\lambda^2 - \nu_k)$$

and hence an eigenvalue λ of A must be of the form $\lambda^2 = \nu$ where $\nu \in \sigma(A_{21}A_{12})$. If, for example, $A_{12} = I_N$ and A_{21} is negative definite, then λ is imaginary. Based on this result, we construct the following algorithm for constructing derivative matrices D and D_0. (A similar algorithm can be used for periodic boundary conditions.)

Algorithm: Construction of D and D_0 for using to solve the acoustic equation.

1. Construct N quadrature nodes and weights according to Definition 11.

2. Construct the matrix S according to (2.27), the matrix K according to (4.1), and the matrices E, F, and G defined in Definition 19 by using (2.21)-(2.25). Construct the matrix P_c and P_c^{-1} according to (2.31).

3. Construct the derivative matrix D with arbitrary boundary conditions for single intervals by using Definition 20, and for multiple intervals by using (3.32) and Tables 3.1-3.3.

4. Construct the derivative matrix D_0 with zero boundary conditions for single intervals by using Definition 20, and for multiple intervals by using (3.32) and Tables 3.1-3.3.

5. Construct

$$L = \left[\begin{array}{cc} 0 & D_0 \\ D & 0 \end{array} \right].$$

6. Project L to obtain

$$L_{proj} = \sum_{|\lambda_k| \leq c} \lambda_k \mathbf{e}_k \mathbf{f}_k^T$$

where $\mathbf{e}_k$ and $\mathbf{f}_k$ are the left and the right eigenvector of L, respectively, scaled such that $\mathbf{f}_k^T \mathbf{e}_k = 1$.

7. Construct D^{proj} as

$$D_{ij}^{proj} = (L_{proj})_{ij},\ i = N/2+1, \dots, 2N,\ j = 1, \dots, N/2$$

and

$$(D_0^{proj})_{ij} = (L_{proj})_{ij},\ i = 1, \dots, N/2,\ j = N/2+1, \dots, 2N.$$

8. Compute $\tilde{D} = P_c D^{proj} P_c^{-1}$ and $\tilde{D}_0 = P_c D_0^{proj} P_c^{-1}$ to represent the derivative matrices with respect to the interpolating basis.

Appendix D

Inverse problems

D.1 Proof of Proposition 26

We first observe that since A is diagonalizable, there exists an invertible matrix S such that $A = Sdiag(\lambda_k)S^{-1}$ where the columns of S are right eigenvectors $\mathbf{e_k}$ of A and the columns of $(S^{-1})^T$ are the left eigenvectors $\mathbf{f_k}$. By scaling the eigenvectors such that $\mathbf{f_k}^T\mathbf{e_k} = 1$, we have that

$$\mathbf{f_k}^T\mathbf{e_l} = \delta_{kl}. \tag{D.1}$$

(1) This follows from the definition of P_k and the scaling $\mathbf{f_k}^T\mathbf{e_k} = 1$.
(2) By using (D.1) we have that

$$P_kP_l = \mathbf{e_k}\mathbf{f_k}^T\mathbf{e_l}\mathbf{f_l}^T = \mathbf{e_k}(\delta_{kl})\mathbf{f_l}^T = \delta_{kl}P_k.$$

(3) Since A is diagonalizable, the eigenvectors form a complete basis and, hence, if $\mathbf{x}$ is an arbitrary vector then

$$\mathbf{x} = \sum_k x_k\mathbf{e_k}$$

for some set of coefficients x_k. Using (D.1) it follows that

$$\sum_l P_l\mathbf{x} = \sum_l P_l\left(\sum_k x_k\mathbf{e_k}\right) = \sum_l\left(\sum_k x_k\mathbf{e_l}\mathbf{f_l}^T\mathbf{e_k}\right)$$

$$= \sum_l x_lP_l\mathbf{e_l} = \sum_l x_l\mathbf{e_l} = \mathbf{x}$$

and, since $\mathbf{x}$ is arbitrary, $I = \sum_k P_k$.

(4) Let $\mathbf{x}$ be an arbitrary vector. Then

$$A\mathbf{x} = A\sum_k x_k \mathbf{e_k} = \sum_k x_k A\mathbf{e_k} = \sum_k \lambda_k x_k \mathbf{e_k} = \sum_k \lambda_k P_k \mathbf{x}$$

and, since $\mathbf{x}$ is arbitrary, $A = \sum_k \lambda_k P_k$.

(5) Using (D.1) and that the eigenvectors form a complete basis, we have that

$$P_k\mathbf{x} = P_k \sum_l x_l \mathbf{e_l} = x_k \mathbf{e_k}.$$

D.2 The inverse problem for acoustics in the time domain

The equation defining the inverse problem for acoustics (7.1), is related to the time domain as follows. Consider the case when the density of the acoustic equation (6.1) is constant throughout the space. Consider a scatterer in the bounded domain $\mathcal{D}$ with the compressibility $\kappa_s(x, y)$ supported in $\mathcal{D}$ and a variable background with the compressibility $\kappa(x, y)$. Then the acoustic pressure p satisfies $\Delta p = \kappa p_{tt}$ outside $\mathcal{D}$ and $\Delta p = \kappa_s p_{tt}$ inside $\mathcal{D}$. We can then write the equation for the pressure as $\Delta p = \kappa(1 + f)p_{tt}$ where

$$f(x, y) = \begin{cases} \frac{\kappa_s(x,y)}{\kappa(x,y)} - 1 & (x, y) \in [-1, 1] \times [-1, 1] \\ 0 & (x, y) \notin [-1, 1] \times [-1, 1] \end{cases} .$$

Fourier transforming p with respect to time gives the frequency domain equation in (7.1).

Appendix E

A fast reconstruction algorithm for electron microscopy [6]

Gregory Beylkin
Department of Applied Mathematics, University of Colorado at Boulder
David N. Mastronarde
Boulder Laboratory for 3-D Electron Microscopy of Cells,
Department of Molecular, Cellular, and Developmental Biology, University of Colorado at Boulder
Kristian Sandberg
Department of Applied Mathematics, University of Colorado at Boulder

Abstract. We have implemented a Fast Fourier Summation algorithm for tomographic reconstruction of three-dimensional biological data sets obtained via transmission electron microscopy. We designed the fast algorithm to reproduce results obtained by the standard filtered backprojection. For two-dimensional images, the new algorithm scales as $O(N_\theta M \log M) + O(MN \log N)$ operations, where N_θ is the number of projection angles and $M \times N$ is the size of the reconstructed image. Three-dimensional reconstructions are constructed from sequences of two-dimensional reconstructions. For typical data sets, the new algorithm is 1.5-2.5 times faster than computing the filtered backprojection using direct summation in the space domain. The speed advantage is even greater as the size of the data sets grows. The new algorithm also allows us to use higher order spline interpolation of the data without additional computational cost. The algorithm has been incorporated into a commonly used package for tomographic reconstruction.

Keywords: electron tomography, weighted backprojection, 3-D reconstruction algorithm, Unequally Spaced Fast Fourier Transform (USFFT)

E.1 Introduction

In this paper we describe a Fast Fourier Summation algorithm for tomographic reconstruction of data obtained with transmission electron microscope. For two-dimensional reconstructions, the algorithm scales as $O(N_\theta M \log M) + O(MN \log N)$ operations, where N_θ is the number of projection angles and $M \times N$ is the size of the reconstructed image. This should be compared to computing the standard filtered backprojection using direct summation in the space domain which scales as $O(N_\theta MN)$. Our algorithm has been applied to data of typical sizes and is shown to be 1.5-2.5 times faster than direct summation. For larger data sets, the time gain is even higher.

The method of filtered backprojection for tomographic reconstruction sums filtered projection data in the space domain (direct summation). We designed the algorithm to reproduce the results of the direct summation algorithm within the required accuracy. We show that without any additional cost, we obtain an algorithm which uses higher order spline interpolation of the data whereas the direct summation uses only linear interpolation.

Reconstruction algorithms for electron microscopy imaging of biological specimens have been described by DeRosier and Klug [19] via a Fourier based method, and by Gilbert [28] via direct summation. For a review, see Frank [22].

The problem of reconstructing an object by measuring projections has a rich history and many applications. For example, the x-ray tomography, radio astronomy, as well as seismic processing are using results of the basic inversion technique first considered by Radon [45]. The Radon inversion formula was rediscovered by Cormack [17] for x-ray tomography, and by Bracewell [12] for radio astronomy. For an introductory overview of the subject, see Deans [18].

The well-known Fourier slice theorem relates projection data to the Fourier transform of the image. The collected projection data corresponds to samples on a polar grid in Fourier space where the polar angles are not necessarily equally spaced. The standard two-dimensional Fast Fourier Transform (FFT) requires sampling on an equally spaced rectangular grid and, hence, fast Fourier reconstruction methods require some interpolation scheme in Fourier space. Such methods have been proposed by e.g. Lanzavecchia and Bellon [41]. We propose a technique that uses the one-dimensional unequally spaced fast Fourier transform for performing summation in the Fourier domain as opposed to summation directly in the space domain as in the direct summation algorithm. The method we propose guarantees accuracy while controlling the computational cost. We also gain flexibility for choosing interpolation schemes and incorporating filters which are applied in the Fourier domain without additional computational cost.

We introduce and formulate the inversion problem in Section E.2. We then give a brief review of the direct summation algorithm, where the summation over the projection angles is performed in the space domain. In Section E.3 we derive an inversion formula which effectively sums over the angles in the Fourier domain. We discretize the inversion formula in

Section E.4 where we also describe how to use the Unequally Spaced Fast Fourier Transform (USFFT), select sampling, and show how to apply higher order interpolation. Finally in Section E.5 we demonstrate the algorithm on data sets collected by The Boulder Laboratory for 3-D Electron Microscopy of Cells at University of Colorado at Boulder and compare the results with those obtained by using the direct summation algorithm. We do this for three-dimensional tomographic reconstructions as a part of the IMOD package ([32] and Kremer, Mastronarde, and McIntosh [36]).

E.2 Preliminaries

E.2.1 Formulation of the problem

We consider the problem of estimating the density of a biological specimen. We restrict ourselves to reconstructing densities in the plane and build the three-dimensional volume as a collection of two-dimensional slices. We consider a specimen illuminated by an electron beam and the intensity of the beam is measured after it passes through the specimen. This procedure is repeated for different tilt angles of the electron beam relative to the specimen as schematically shown in Figure 1 below. In practice, the tilt angles θ are limited to some interval and typically range between $\simeq \pm 70^\circ$ with an angular separation of 1–2 degrees. The intensity is measured at M points for each angle θ_l, $l = 1, 2, \dots, N_\theta$. The problem is then formulated as that of finding a discrete approximation to the density of the specimen, $g(x, z)$, on a rectangular (equally spaced) grid with M points in the x-direction and N points in the z-direction. Adopting a common convention used in electron microscopy, we refer to the x-direction as the wide direction and to the z-direction as the thick direction. The number of points in the x-direction is typically 500-2000. The number of points in the z-direction is usually less than the number of points in the x-direction.

As is customary, we assume that the intensity of the electron beam decays along straight lines through the specimen. For a comprehensive treatment of the physics of electron microscopy, see Reimer [46]. We consider a family of straight lines through the specimen given by

$$C_{t,\theta} = \{\ (x, z) \in \mathbb{R}^2 \mid t = x\cos\theta + z\sin\theta\ \} \tag{E.1}$$

(see Figure 1) and define the function $R_\theta(t)$ as the line integral of the density function $g(x, z)$ along $C_{t,\theta}$,

$$R_\theta(t) = \int_{C_{t,\theta}} g(x, z)\ ds. \tag{E.2}$$

We note that evaluating $R_\theta(t)$ for all lines through the support of $g(x, z)$ is equivalent to computing the Radon transform of $g(x, z)$. We assume the measured intensity is described

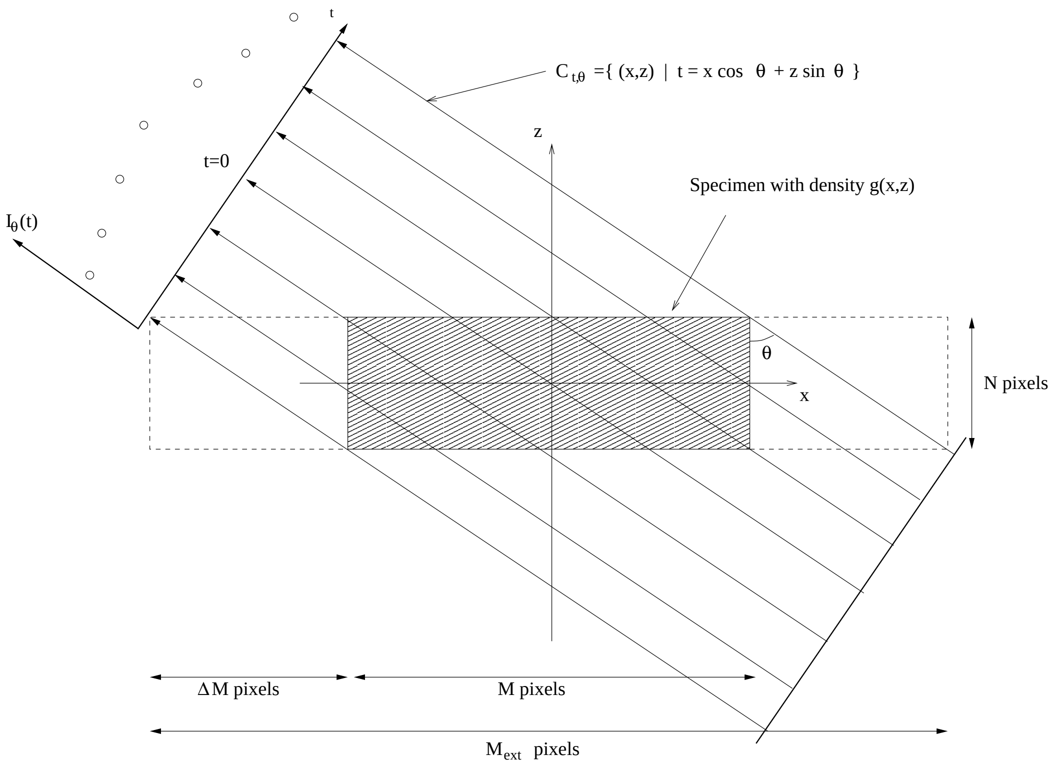

t
$C_{t,\theta}$ ={ (x,z) | t = x cos θ + z sin θ }
z
t=0
Specimen with density g(x,z)
$I_\theta(t)$
θ
x
N pixels
ΔM pixels
M pixels
M_{ext} pixels

Figure E.1: A specimen with density distribution $g(x, z)$ is illuminated by an electron beam through different angles θ. The intensity is recorded as the function $I_\theta(t)$. The intensity of the electron beam is modelled to decay along line integrals through the specimen. The line integrals are defined along the family of lines $C_{t,\theta}$. The variables ΔM and M_{ext} are discussed in Section E.4.3.

by $I_\theta(t) = I_0 e^{-R_\theta(t)}$, where I_0 is the incident intensity. We assume that I_0 is a constant, and set $I_0 = 1$.

Our goal is to approximate $g(x, z)$ by measuring $I_\theta(t)$. On taking $R_\theta(t) = -\ln I_\theta(t)$, our measurements provide us with $R_{\theta_l}(t_k)$, where θ_l and t_k are discretizations of θ and t respectively. Sampling $R_\theta(t)$, typically at equal angles and distances, yields the matrix,

$$r_{kl} = R_{\theta_l}(t_k), \tag{E.3}$$

where $k = 0, 1, \ldots, M-1$ and $l = 1, 2, \ldots, N_\theta$. Each column l of the matrix contains all measurements for the angle θ_l.

The problem can now be formulated as given the measurement data r_{kl}, find an approximation to $g(x_m, z_n)$, where x_m, z_n is some grid, $m = 1, 2, \ldots, M$ and $n = 1, 2, \ldots, N$. The total amount of data is significant since it consists of measurements from a large number of two dimensional slices of a specimen (typically as many as points in the x-direction). Therefore, we want to not only find an accurate approximation of the density distribution, but also to compute it in an efficient manner.

E.2.2 Inversion of the Radon transform

As is well known, (see e.g. Deans [18]), the two-dimensional density $g(x, z)$ can be recovered from the line integrals $R_\theta(t)$ in (E.2) via the integral

$$g(x, z) = \int_0^\pi (\rho * R_\theta)(t(x, z))\, d\theta, \tag{E.4}$$

where ρ is a convolution operator. For each angle θ the projection coordinate t depends on (x, z) according to (E.1) but we will omit the angle dependence in our notation for t. In the Fourier domain the convolution operator is represented by

$$\hat{\rho}(\omega) = |\omega|. \tag{E.5}$$

In practice, this filter is often modified by a bandlimiting window.

E.2.3 Filtered backprojection using direct summation

If we assume that the electron beam is modelled by line integrals over infinitesimally thin straight lines then reconstructing the density of a specimen from its projections can be viewed as the inversion of the Radon transform. Many reconstruction algorithms rely on this fact and solve the inversion problem analytically by discretizing the inverse Radon transform [18], [28]. In this section we describe the widely used direct summation algorithm (also known as R-weighted backprojection).

We discretize (E.4) by the sum

$$g(x_m, z_n) = \sum_{l=1}^{N_\theta} w_l \; [\rho * R_{\theta_l}](t(x_m, z_n)) \tag{E.6}$$

where w_l are weights and $t(x_m, z_n)$ are given by (E.1). For measurements performed over equally spaced angles, the weights w_l are usually set to one. Since we have measurements only for a discrete set $t(x_m, z_n)$, the values $R_{\theta_l}(t(x_m, z_n))$ are estimated by some interpolation scheme, usually piecewise linear interpolation. Let us summarize the steps for estimating the density of $g(x, z)$ from measurements of projections as follows.

1. Filter the data to obtain $(\rho * R_{\theta_l})(t_k), \; k = 0, 1, \dots, M-1$:
   (a) Apply the FFT along the columns of the matrix r_{kl} defined by (E.3).
   (b) Multiply each element of the transformed matrix by the (pre-computed) filter coefficients and the weights w_l if necessary.
   (c) Apply the inverse FFT column wise.
2. Summation:
   (a) For each given (x_m, z_n), compute $t(x_m, z_n)$.
   (b) Find $(\rho * R_{\theta_l})(t(x_m, z_n))$ by linearly interpolating $(\rho * R_{\theta_l})(t_k)$.
   (c) Sum according to (E.6).

Step 2 dominates the computational cost since we have to sum over N_θ terms $N \times M$ times, for the total computational cost of $O(N_\theta M N)$. Usually N_θ, M and N are of the same order of magnitude so the above algorithm has a computational cost of $O(N^3)$.

E.3 Inversion algorithm in the Fourier domain

Let us first derive an algorithm for inversion in the Fourier domain that is identical to the direct summation algorithm. In the following section we will describe a numerical implementation that results in a fast $O(N_\theta M \log M) + O(MN \log N)$ algorithm.

Our goal is to find the density $g(x, z)$ on an equally spaced grid in x and z. In our derivation of the Fast Fourier Summation algorithm, we discretize in z but keep x as a continuous variable until the very end of our derivation. If we fix $z = z_n$ while treating x as a continuous variable, we write $g_n(x) = g(x, z_n)$ and, similarly, $t_n(x) = t(x, z_n)$, where t is defined in (E.1). In the following derivation $\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t) e^{2\pi i t \omega} \, dt$ denotes the Fourier transform of a function $f(t)$.

Consider the sum used for filtered backprojection (E.6),

$$g_n(x) = \sum_{l=1}^{N_\theta} f_l(t_n(x)) \ , \tag{E.7}$$

where we have introduced $f_l(t_n(x)) = w_l(\rho * R_{\theta_l})(t(x, z_n))$. In our approach we will Fourier transform (E.7) with respect to x and perform the summation in the Fourier domain. The Fourier transform of (E.7) with respect to x gives

$$\begin{aligned} \hat{g}_n(\omega) &= \sum_{l=1}^{N_\theta} \int_{-\infty}^{\infty} f_l(x\cos\theta_l + z_n \sin\theta_l) e^{2\pi i x\omega} \ dx \\ &= \sum_{l=1}^{N_\theta} \frac{e^{-2\pi i\omega z_n \tan\theta_l}}{\cos\theta_l} \int_{-\infty}^{\infty} f_l(s) e^{2\pi i s \frac{\omega}{\cos\theta_l}} \ ds \\ &= \sum_{l=1}^{N_\theta} v_l(\omega) e^{-2\pi i \xi_l(\omega) z_n}, \end{aligned} \tag{E.8}$$

where $\xi_l(\omega) = \omega \tan\theta_l$ and

$$v_l(\omega) = \frac{1}{\cos\theta_l} \int_{-\infty}^{\infty} f_l(s) e^{2\pi i s \frac{\omega}{\cos\theta_l}} \ ds \ . \tag{E.9}$$

By definition,

$$f_l(t_n(x)) = w_l(\rho * R_{\theta_l})(t(x, z_n)) = w_l \int_{-\infty}^{\infty} \hat{\rho}(\omega) \hat{R}_{\theta_l}(\omega) e^{-2\pi i\omega t} \ d\omega, \tag{E.10}$$

which combined with (E.9) gives us

$$v_l(\omega) = \frac{w_l}{\cos\theta_l} \hat{\rho}(\frac{\omega}{\cos\theta_l}) \hat{R}_{\theta_l}(\frac{\omega}{\cos\theta_l}). \tag{E.11}$$

Recall that $R_{\theta_l}(t_k)$ corresponds to a discrete set of measurements. In the filtered backprojection algorithm, we use interpolation to define $R_{\theta_l}(t)$ for any t. If linear interpolation is used, then $R_{\theta_l}(t)$ is continuous. Alternatively, let us introduce the "hat" function, or the linear spline,

$$\beta(t) = \left\{ \begin{array}{ll} 1 - |t| & -1 < t < 1 \\ 0 & \text{otherwise} \end{array} \right. . \tag{E.12}$$

We express piecewise linear interpolation of our discrete data using $\beta(t)$ by defining

$$R_{\theta_l}(t) = \sum_{k=0}^{M-1} \beta(t - k + x_s) \ r_{kl}, \tag{E.13}$$

where x_s is a shift parameter which depends on the selection of the coordinate system in x. It is easily verified that the function $R_{\theta_l}(t)$ is continuous with respect to t. Using (E.13) we have

$$\begin{aligned}\hat{R}_{\theta_l}(\omega) &= \sum_{k=0}^{M-1} r_{kl} \int_{-\infty}^{\infty} \beta(t-k+x_s) e^{2\pi i t\omega} dt \\ &= \sum_{k=0}^{M-1} r_{kl} e^{2\pi i (k-x_s)\omega} \int_{-\infty}^{\infty} \beta(s) e^{2\pi i s\omega} ds \\ &= e^{-2\pi i x_s \omega} \hat{\beta}(\omega) \sum_{k=0}^{M-1} r_{kl} e^{2\pi i k\omega}.\end{aligned} \tag{E.14}$$

Combining (E.11) and (E.14) yields

$$\begin{aligned} v_l(\omega) &= \frac{w_l}{\cos\theta_l} e^{-2\pi i x_s \frac{\omega}{\cos\theta_l}} \hat{\rho}(\frac{\omega}{\cos\theta_l}) \hat{\beta}(\frac{\omega}{\cos\theta_l}) \sum_{k=0}^{M-1} r_{kl} e^{2\pi i k \frac{\omega}{\cos\theta_l}} \\ &= F_l(\omega) \hat{r}_l(\frac{\omega}{\cos\theta_l}), \end{aligned} \tag{E.15}$$

where

$$F_l(\omega) = \frac{w_l}{\cos\theta_l} e^{-2\pi i x_s \frac{\omega}{\cos\theta_l}} \hat{\rho}(\frac{\omega}{\cos\theta_l}) \hat{\beta}(\frac{\omega}{\cos\theta_l}), \tag{E.16}$$

and

$$\hat{r}_l(\frac{\omega}{\cos\theta_l}) = \sum_{k=0}^{M-1} r_{kl} e^{2\pi i k \frac{\omega}{\cos\theta_l}}. \tag{E.17}$$

Note that the factor $F_l(\omega)$ is independent of the data once the angles θ_l are known.

The final step is computing $g_n(x)$ from $\hat{g}_n(\omega)$. By taking the inverse Fourier transform of (E.8) we arrive at

$$g_n(x) = \int_{-\infty}^{\infty} \left(\sum_{l=1}^{N_\theta} v_l(\omega) e^{-2\pi i \xi_l(\omega) z_n} \right) e^{-2\pi i \omega x} \; d\omega, \tag{E.18}$$

where $\xi_l(\omega) = \omega \tan\theta_l$ and $v_l(\omega)$ are defined in (E.15). We also observe

$$\begin{aligned} g_n(x) &= \int_{-\infty}^{\infty} \left(\sum_{l=1}^{N_\theta} v_l(\omega) e^{-2\pi i \xi_l(\omega) z_n} \right) e^{-2\pi i \omega x} \; d\omega \\ &= \sum_{j\in\mathbb{Z}} \left(\int_{-\frac{1}{2}}^{\frac{1}{2}} \left(\sum_{l=1}^{N_\theta} v_l(\omega+j) e^{-2\pi i \xi_l(\omega+j) z_n} \right) e^{-2\pi i (\omega+j) x} \; d\omega \right), \end{aligned} \tag{E.19}$$

where we recall

$$v_l(\omega + j) = \frac{w_l e^{-2\pi i x_s \frac{\omega}{\cos\theta_l}} \hat{\rho}(\frac{\omega+j}{\cos\theta_l}) \hat{\beta}(\frac{\omega+j}{\cos\theta_l}) \hat{r}_l(\frac{\omega+j}{\cos\theta_l})}{\cos\theta_l}.$$

Let us consider a bandlimited filter ρ such that the support of $\hat{\rho}$ is contained in $[-\frac{1}{2}, \frac{1}{2}]$. As an important example consider

$$\hat{\rho}(\omega) = \begin{cases} |\omega|, & |\omega| \le \frac{1}{2} \\ 0, & |\omega| > \frac{1}{2} \end{cases}.$$

For this example we observe that $v_l(\omega + j) = 0$ for all $j \neq 0$. Hence (E.19) reduces to

$$g_n(x) = \int_{-\frac{1}{2}}^{\frac{1}{2}} \left(\sum_{l=1}^{N_\theta} v_l(\omega) e^{-2\pi i \xi_l(\omega) z_n} \right) e^{-2\pi i \omega x} \, d\omega. \tag{E.20}$$

Remark. Note that equation (E.20) is equivalent to the sum (E.6) used in the direct summation algorithm, where ρ is a bandlimiting filter. □

E.4 Implementation

E.4.1 Discretization

Let us evaluate (E.20) at pixel locations in our final image given by

$$x_m = -x_s + m, \quad m = 1, 2, \ldots, M_f,$$

where $M_f > M$ will be selected in Section E.4.3. By discretizing ω as

$$\omega_k = -\frac{1}{2} + \frac{k}{M_f}, \quad k = 1, 2, \ldots, M_f\ ,$$

we approximate (E.20) by

$$\begin{aligned} g_n(x_m) &\simeq \frac{1}{M_f} \sum_{k=-\frac{M_f}{2}+1}^{\frac{M_f}{2}} \left(\sum_{l=1}^{N_\theta} v_l(\frac{k}{M_f}) e^{-2\pi i \xi_l(\frac{k}{M_f}) z_n} \right) e^{-2\pi i \frac{k}{M_f} x_m} \\ &= \frac{1}{M_f} \sum_{k=-\frac{M_f}{2}+1}^{\frac{M_f}{2}} \left(\sum_{l=1}^{N_\theta} v_l(\frac{k}{M_f}) e^{-2\pi i \xi_l(\frac{k}{M_f}) z_n} \right) e^{2\pi i \frac{k}{M_f} x_s} e^{-2\pi i \frac{k}{M_f} m} \\ &= \frac{1}{M_f} \sum_{k=-\frac{M_f}{2}+1}^{\frac{M_f}{2}} \hat{g}_{nk} e^{-2\pi i k \frac{m}{M_f}}, \end{aligned} \tag{E.21}$$

for $m = 1, 2, \dots, M_f$, where

$$\hat{g}_{nk} = e^{2\pi i \frac{k}{M_f} x_s} \sum_{l=1}^{N_\theta} v_l(\frac{k}{M_f}) e^{-2\pi i \xi_l(\frac{k}{M_f}) z_n}. \tag{E.22}$$

The final expression (E.21) is an approximation of the exact reconstruction formula (E.20). By choosing M_f large enough (see Section E.4.3), we can approximate the exact expression with arbitrary precision. Since the precision of transmission electron microscopy data is typically low, we have found that setting M_f to $1.5-2$ times the number of projections M usually suffices. We will demonstrate the accuracy of (E.21) in Section E.5.2.

Remark. The z-variable can be discretized as $z_n = z_s + n$. The constant z_s shifts the coordinate system in z. Such shifts are essential in some cases when reconstructing three-dimensional volumes as discussed in Section E.5.1. □

E.4.2 Unequally spaced FFT (USFFT)

As is well-known, the discrete Fourier transform

$$\hat{u}_n = \sum_{k=0}^{N-1} u_k e^{\pm 2\pi i k \frac{n}{N}}, \; n = 0, 1, \dots, N-1 \tag{E.23}$$

can be computed in $O(N \log N)$ operations using the Fast Fourier Transform (FFT) (Cooley and Tukey [16]).

Since our algorithm uses the sum (E.22), we need a fast algorithm to compute the sums

$$\hat{u}_n = \sum_{k=1}^{M} u_k e^{\pm 2\pi i \xi_k n}, \; n = -\frac{N}{2}, -\frac{N}{2}+1, \; \ldots, \frac{N}{2}-1 \tag{E.24}$$

and

$$\hat{u}(\xi_k) = \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} u_k e^{\pm 2\pi i n \xi_k}, \; k = 1, 2, \ldots, M \tag{E.25}$$

for a given real set of points $\{\xi_k\}_{k=0}^{M}$, where $|\xi_k| < 1/2$ for each k. We note that M may be different from N. Also, we can always reformulate our problem so that the condition $|\xi_k| < 1/2$ is fulfilled. Indeed,

$$e^{2\pi i \xi_k n} = e^{2\pi i (\xi_k + l) n}$$

for any integer l and therefore if $|\xi_k| > 1/2$ (but finite) we can always find an integer l such that $|\xi_k + l| < 1/2$.

The sums in (E.24) and (E.25) can be computed using Unequally Spaced Fast Fourier Transform (USFFT) (see Dutt and Roklin [21], and Beylkin [4]). This algorithm effectively contains interpolation that guarantees the accuracy of the result. We use the algorithm in [4] which requires $C_1 M + C_2 N \log N$ operations and produces a prescribed accuracy. For this reason we can match our results with those obtained via the filtered backprojection. We note that since accuracy is guaranteed, the USFFT can be used as a step in an algebraic reconstruction technique although we do not pursue this point further in this paper.

E.4.3 Oversampling

Applying the backprojection operator, we note that the collected data will produce non-zero intensity not only within but also *outside* the the shaded portion of the specimen in Figure 1. We refer to the shaded portion as "the region of interest". The extention of the support outside the region of interest, does not cause any problems in the direct summation algorithm. However, by using (E.21), we will reconstruct a periodic image. If the length M_f is not large enough, the reconstructed area outside the region of interest will wrap around and overlap with the region of interest causing undesirable artifacts. Hence, knowing the full extent of the reconstruction, M_{ext} in Figure E.1, it is essential to choose the number of frequencies M_f large enough. Since the size of M_f affects the computational speed of the algorithm, it is important to choose a minimal M_f.

From Figure E.1 we observe that

$$\Delta M = \frac{M_{ext} - M}{2} = N \tan \theta_{max}$$

where $\theta_{max} = \max\{|\theta_l|\}_{l=1}^{N_\theta}$. This is illustrated in Figure E.2, where there is no wraparound between the left and right sides of the reconstruction.

If M_f is smaller than the spatial support $M_{ext} = M + 2\Delta M$, we will observe aliasing, namely, the left and right part of the image will overlap. As long as the overlapping region is outside the region of interest, no harm is done to the reconstruction. Hence, in order to avoid aliasing artifacts, we must choose

$$M_f \geq M + N \tan\theta_{max}. \tag{E.26}$$

This is illustrated in Figure E.3, where the wraparound does not overlap the region of interest. Choosing M_f larger than M can be thought of as oversampling the image. The

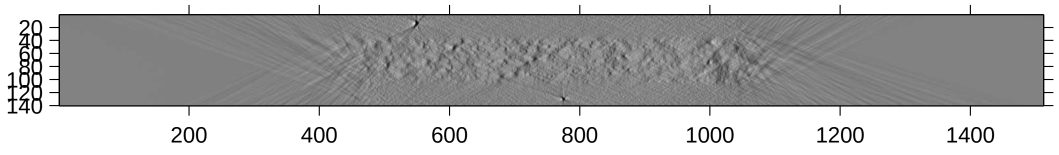


Figure E.2: For this reconstruction $M = 572$, $N = 140$, and $\theta_{max} = 73.31^o$. This gives that $\Delta M \simeq 467$ and, hence, the extent of the horizontal support of the reconstruction is given by $M_{ext} = 1506$. For this reconstruction we chose $M_f = 1512$. We observe that for this choice of M_f, the entire support of the image fits in the reconstructed image.

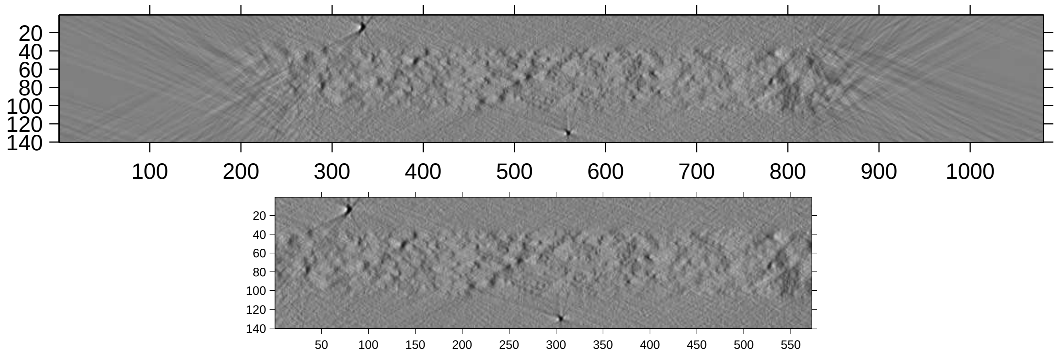


Figure E.3: We construct the same data set as in Figure E.2, but here we chose $M_f = 1080$ which satisfies (E.26). Here we do see some wraparound in the left and right part of the upper image, but it does not overlap the region of interest cropped out in the lower image.

oversampling factor is given by the ratio M_f/M and it is desirable to make this factor as close to one as possible. For typical data sets, we have found that this oversampling factor ranges between $1.5 - 2$.

E.4.4 Numerical algorithm

Our first goal in designing a Fast Fourier Summation algorithm is to match the direct summation algorithm. We do it for two reasons. First, since the direct summation algorithm has been used for a long time in this field, we avoid the issue of acceptance. Second, we demonstrate the flexibility of the Fast Fourier Summation algorithm. As it turns out by changing parameters we can achieve higher order interpolation in the input data in comparison with the linear interpolation used within the direct summation. The results obtained in this manner appear to be less "noisy", but we leave the practical utility of such interpolation outside the scope of this paper.

In order to match the direct summation algorithm we consider linear interpolation and use a bandlimited version of the filter defined in (E.5). We use the discretization in both x and frequency, described in Section E.4.1. We consider a discretization of z into N equally spaced points and denote them z_n. The discretization of the radial weighting and interpolation filters is as follows

$$\hat{\rho}(\omega_k) = \begin{cases} |\omega_k|\,, & \omega_k \leq 1/2 \\ 0, & \omega_k > 1/2 \end{cases}$$

$$\hat{\beta}(\omega_k) = \left(\frac{\sin(\pi\omega_k)}{\pi\omega_k}\right)^2.$$

We discuss other choices of filters in Section 2.6.3 below.

Next we summarize the main steps of the Fast Fourier Summation algorithm. Let us consider a case when given M projection points at N_θ angles, we wish to reconstruct a sequence of N_i images at $M \times N$ grid points. In what follows M_f satisfying (E.26) is the number of spatial frequency modes in the x direction.

Algorithm.

1. Precomputation: For each angle θ_l and each frequency ω_k, compute $F_l(\omega_k)$ defined by (E.16).

2. For each image, evaluate (E.21):

   (a) For each angle θ_l and each frequency ω_k, compute the sum (E.17) using the USFFT and multiply the result by $F_l(\omega_k)$ to obtain $v_l(\omega_k)$ in (E.15). See the Appendix for details of organizing computation of the USFFT. Computational cost: $O(N_\theta M_f) + O(N_\theta M \log M)$.

   (b) For each frequency ω_k, compute the sum in (E.22) using the USFFT. See the Appendix for details. Computational cost: $O(M_f N_\theta) + O(M_f N \log N)$.

(c) Compute the sum in (E.21) using the FFT. Computational cost: $O(NM_f \log M_f)$

The steps are illustrated in Figure E.4.

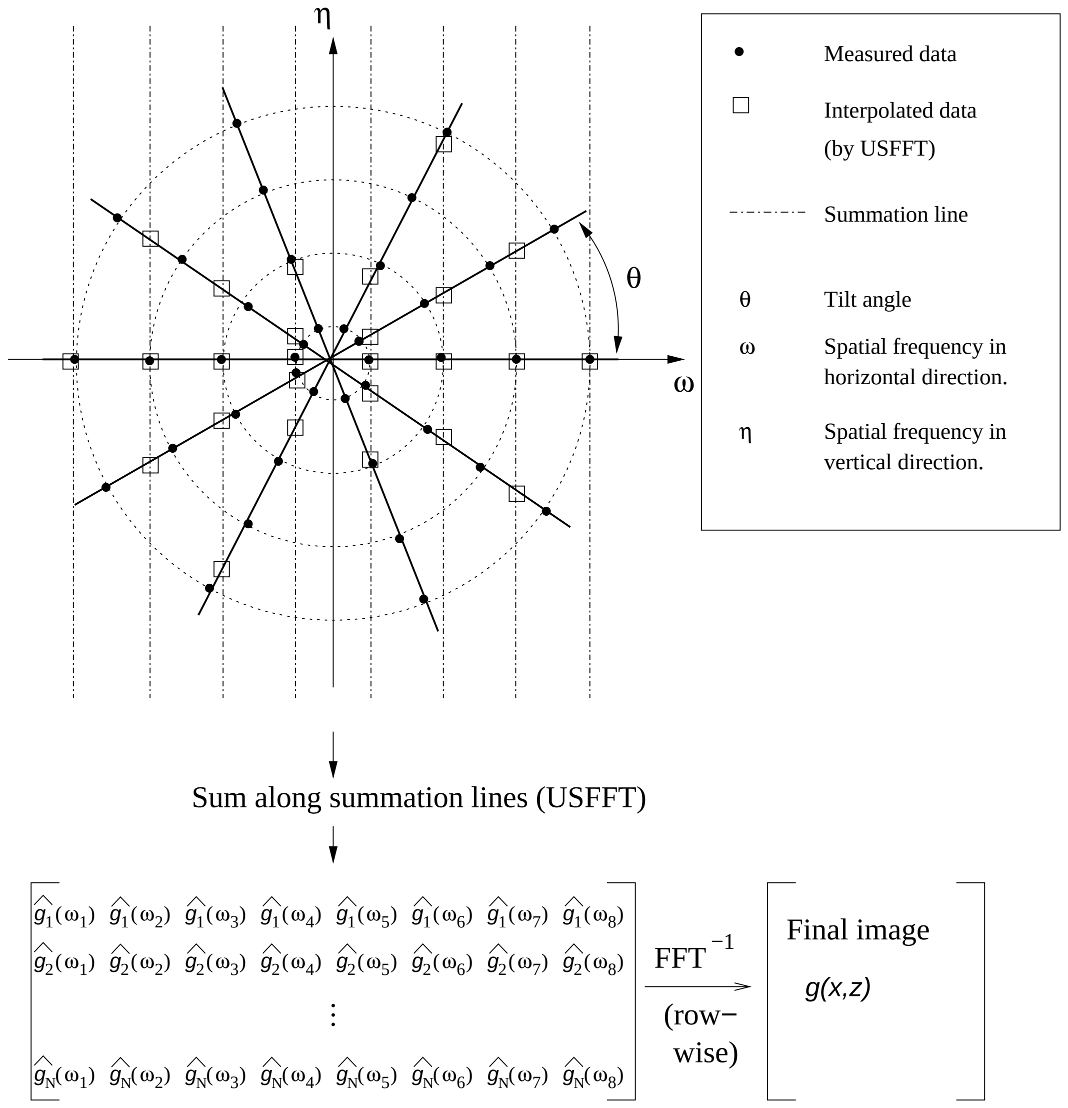

η
θ
ω
● Measured data
□ Interpolated data (by USFFT)
Summation line
θ Tilt angle
ω Spatial frequency in horizontal direction.
η Spatial frequency in vertical direction.
Sum along summation lines (USFFT)
$\hat{g}_1(\omega_1)$ $\hat{g}_1(\omega_2)$ $\hat{g}_1(\omega_3)$ $\hat{g}_1(\omega_4)$ $\hat{g}_1(\omega_5)$ $\hat{g}_1(\omega_6)$ $\hat{g}_1(\omega_7)$ $\hat{g}_1(\omega_8)$
$\hat{g}_2(\omega_1)$ $\hat{g}_2(\omega_2)$ $\hat{g}_2(\omega_3)$ $\hat{g}_2(\omega_4)$ $\hat{g}_2(\omega_5)$ $\hat{g}_2(\omega_6)$ $\hat{g}_2(\omega_7)$ $\hat{g}_2(\omega_8)$
$\vdots$
$\hat{g}_N(\omega_1)$ $\hat{g}_N(\omega_2)$ $\hat{g}_N(\omega_3)$ $\hat{g}_N(\omega_4)$ $\hat{g}_N(\omega_5)$ $\hat{g}_N(\omega_6)$ $\hat{g}_N(\omega_7)$ $\hat{g}_N(\omega_8)$
FFT^{-1} (row−wise)
Final image
g(x,z)

Figure E.4: The measurement given by the data acquisition is represented by the filled dots. Step 2a in the algorithm amounts to interpolating the data to the positions indicated by the squares. Since we are using a bandlimited filter, all points are restricted to within the bandlimit. Step 2b of the algorithm computes the matrix $\hat{g}_n(\omega_k)$ column-wise by adding data along the summation lines. The final step of the algorithm computes the image by applying the inverse FFT row-wise on $\hat{g}_n(\omega_k)$.

A Fast Fourier Summation algorithm is important when reconstructing a large number of two-dimensional slices. In most applications, the angles θ_l and filters are the same for all slices, which means that the precomputation step in the algorithm above only needs to be done once while steps 2a-c are repeated for each slice.

We remind that $M_f = cM$ where c is an oversampling factor given by (E.26). In most applications, the tilt angles θ are bounded, and projections at high tilt angles requires thin specimens, that is a small value of N. Hence, the oversampling factor is bounded in most applications and we have found that typically $M_f \leq 2M$. Therefore, the total computational cost of the full three-dimensional reconstruction algorithm is given by $O(N_i N_\theta M \log M) + O(N_i N_\theta M_f \log M_f)$. This should be compared to the cost $O(N_i M N N_\theta)$ for the traditional method of direct summation. Actual speed comparisons are given in Section E.5.2.

Remark. The sum $\sum_{k=0}^{M-1} r_{kl} e^{2\pi i k \frac{\omega}{\cos\theta_l}}$ computed in Step 2 of the algorithm is similar to the sums that can be computed with the FFT. For each fixed angle θ_l, the nodes are equally spaced if ω is sampled on an equally spaced grid. The factor $\frac{1}{\cos(\theta_l)}$ means that we are effectively "stretching" the spacing between these nodes to evaluate the sum at a new set of equally spaced grid points (see Figure 2). This prevents direct use of the FFT. However, the sum is of the type in (E.25) which means that these sums can be evaluated in $O(N_\theta M) + O(N_\theta M_f \log M)$ operations. Again, the details are given in the Appendix. $\square$

E.4.5 Interpolation

We have demonstrated in Section E.3 that the Fast Fourier Summation reproduces the result of the direct summation algorithm that uses piecewise linear interpolation of the data. In direct summation, the interpolation is applied in Step 2b in the algorithm given in Section E.2.3. Let us show that higher order interpolation schemes can be easily incorporated into the Fast Fourier Summation without additional computational costs.

As in the case of linear interpolation, the piecewise interpolation of higher order can be described in the space domain by using the B-splines, (Schoenberg [50]). For the linear interpolation, the linear B-spline is given by the hat function in (E.12). Although higher order interpolation schemes may be cumbersome to implement in the space domain, in our algorithm the interpolation is implemented in the Fourier domain where it is simply a multiplication by an appropriate filter. From the definition of the B-splines as a repeated convolution of the characteristic function, it follows that the Fourier transform of the B-spline of odd order k is given by

$$\hat{\beta}^{(k)}(\omega) = \left(\frac{\sin(\pi\omega)}{\pi\omega}\right)^k, \ k = 1, 3, 5, \ldots . \tag{E.27}$$

Thus, all we need to use higher order interpolation in our reconstruction algorithm, is to apply (E.27) when computing the factor $F_l(\omega)$ given by (E.16). This is done in Step 1 in the algorithm in Section E.4.4.

Since the higher order interpolation filter (E.27) decays faster with order k, increasing the order of interpolation effectively low-passes the data and can therefore be used to reduce high frequency noise. We leave the choice of the order as a parameter in the algorithm. Such choice should be guided by practical considerations and experiences. Since using high order interpolation has a bandlimiting effect, it also implies that we need fewer frequencies for the reconstruction which, in turn, provides a marginal speed improvement.

E.5 Results

E.5.1 Incorporation of the Fast Fourier Summation into IMOD

The Fast Fourier Summation (FFS) was incorporated into the Tilt program of the IMOD package [32], [36]. The basic framework of this program was originally written by Mike Lawrence while at the Medical Research Council in Cambridge. Before the speed comparisons reported here, the direct summation procedures in Tilt were optimized in two ways. First, all boundary checks were moved outside of the inner summation loops. Second, each line of input data was stretched by the cosine of the tilt angle, with the stretched data oversampled by a factor of two to minimize the filtering effect of interpolating twice. The result is that a line of stretched input data can be added into a line of the tomographic slice by stepping through the input line at a fixed interval and with fixed interpolation factors. The advantages of FFS would have been considerably higher than described here without these improvements to the original code.

Further developments in the Tilt program and in the FFS algorithm were spurred by the desire to correct for the plane of the specimen not being parallel to the tilt axis. When the specimen is tilted in the plane of the reconstruction, the thickness of the reconstruction must be increased to contain the material of interest, sometimes by as much as 50% for large data sets. To avoid this, the Tilt program can apply a tilt around the x axis when computing the reconstruction by direct summation. However, this means that one output slice is based on multiple lines of input data, making fast backprojection unusable for this computation. To solve this problem, the FFS algorithm was modified to shift the output slice vertically (in the z dimension) by a chosen amount. Tilt now uses FFS to compute slices perpendicular to the tilt axis, (dashed vertical line in Figure E.5), each shifted appropriately in z so that it covers the region needed for the reconstructed output slice. It then interpolates between these perpendicular slices to obtain an output slice tilted around the x axis (tilted solid line in Figure E.5).

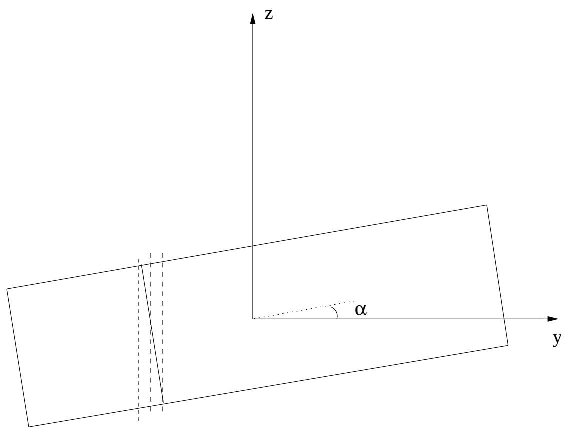


Figure E.5: Side view of section to be reconstructed where the section is tilted by the angle α around the x-axis. Dashed lines are slices computed by the algorithm and the solid line is the output slice interpolated from vertical slices.

E.5.2 Tests

Accuracy

The data set shown here is based on images from the mitotic spindle of a dividing cell from the PtK cell line. The cell was high-pressure frozen, freeze-substituted, embedded in epon-araldite, and sectioned at 300 nm. The section was tilted between ± 70 at 1.5 degree intervals and images were recorded on film in a JEOL microscope operating at 1,000 KeV. The grid was then rotated by 90 degrees in the specimen holder and a second, similar tilt series was taken. Data were digitized at a pixel size of 2.3 nm using a CCD camera. The resolution of both the film and the CCD camera were good enough to ensure that the images have substantial information out to the Nyquist frequency. The overall Modular Transfer Function (MTF) is estimated to be 30% at Nyquist. The single axis and combined tomograms were computed as described previously (see Mastronarde [42]).

Figure E.6 shows that the FFS algorithm produces essentially the same reconstruction as direct summation. One slice from the reconstruction of the test data set computed with FFS appears in Figure E.6a. The two densest features, one above and one below the sectioned material, are colloidal gold particles placed on the surface of the support film as fiducial markers for alignment. The difference between this image and the corresponding slice reconstructed by direct summation appears in Fig. E.6b, at the same contrast as in Fig. E.6a, and again in Fig E.6c with the contrast amplified 29 times. Aside from differences in the edge artifacts produced by the two procedures, the most prominent difference is at the gold particles, where the difference is about 1% of the density of the particles relative to

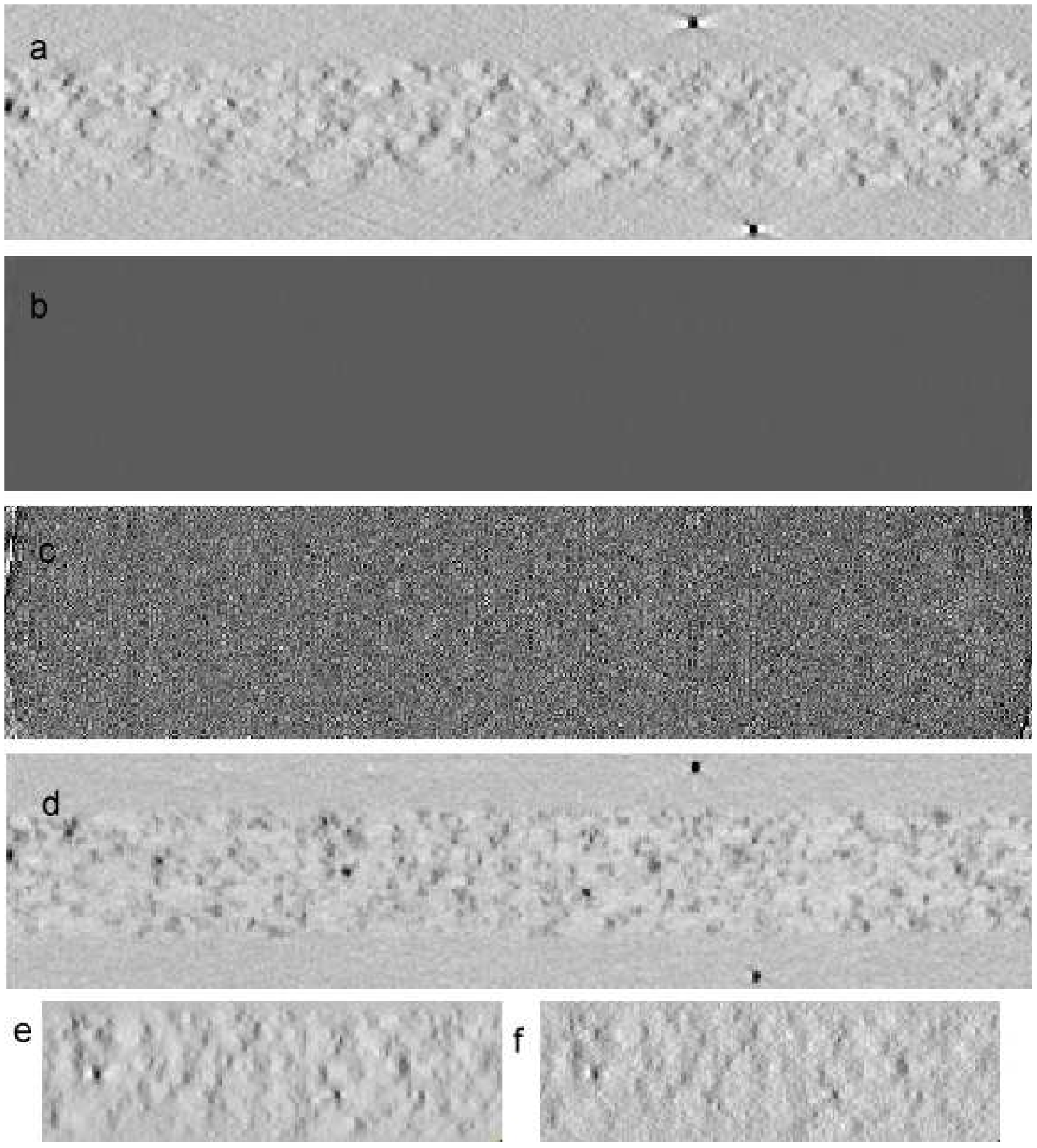


Figure E.6: (a): A test data set computed with the Fast Fourier Summation algorithm. (b): The difference between the image in (a) and the corresponding slice reconstructed by direct summation with the same contrast as in (a). (c): The same dataset as in (b), but with the contrast amplified 29 times. (d): Test data set for the Fourier ring correlation test. (e): Portion of reconstruction of the data set in (d) without noise added. (f): Portion of reconstruction of the data set in (d) with noise added.

the background. The differences within the section are smaller and are less than 2% of the range of densities found there.

For a quantitative assessment of the fidelity of reconstruction by the two methods, a sample volume was reprojected, then tomograms were built from the re-projections and compared with the original volume by Fourier ring correlation (see Saxton and Baumeister [49]). The combined dual-axis tomogram of our test data set was used as the sample volume; Fig. E.6d shows the corresponding slice from this volume. This volume was considered suitable because the characteristic artifacts from single axis tomography, namely the dark rays at the terminal angles of the tilt series and white shadows to the sides of densities, are much reduced there [42]. The volume was reprojected at 1.5 degree intervals between ± 66 degrees, either with no added noise or with added Poisson noise. The latter images were filtered by the estimated MTF of the film and digitizing apparatus. Noise levels equivalent to 1000, 3000 or 9000 electrons/pixel were explored, to represent both low dose and standard exposure situations. Several different high frequency cutoffs were applied in each of these situations. Figures E.6d and e show a central portion of the reconstruction from the noise-free data and the data with 3000 electrons/pixel of noise, respectively.

Figure E.7 shows the correlation between the Fourier transforms of the reconstruction and of the test volume, as a function of spatial frequency, averaged over 190 slices of each reconstruction. Clearly the two methods are equivalently good at reconstructing the test volume, regardless of the amount of noise added. From all of the cases tested, there was a tendency for the Fast Fourier Summation to be slightly worse at low frequencies and slightly better at higher frequencies except near Nyquist; however, in practical terms these differences are insignificant.

Speed

The speed of the Fast Fourier Summation algorithm was explored with a variety of data sizes and under several computer architectures. The width of the input data was varied from 256 to 4096 pixels; the number of projections was varied from 21 to 375, with an extreme angle of either 60 degrees or 70 degrees; and the thickness of the reconstruction was varied from 25 to 400 pixels. For a particular data size, CPU time was measured for the computation of one slice and ten slices, and the incremental time to compute 9 slices was used to compare Fast Fourier Summation and direct summation. Comparisons were done on SGI Octane computers with R10000 and R12000 processors, on a Sun Sparc Ultra-60, and on Intel-architecture computers with Pentium 3, Pentium 4, Athlon Thunderbird, and Athlon MP processors. For the SGI and Sun tests, programs were compiled with the native compilers; the Intel-architecture tests were done both with programs compiled with GNU compilers and with Intel compilers.

The results in Figure E.8, from SGI, Pentium 4, and Athlon processors, illustrate the range of performance benefits found with Fast Fourier Summation. Each graph shows the

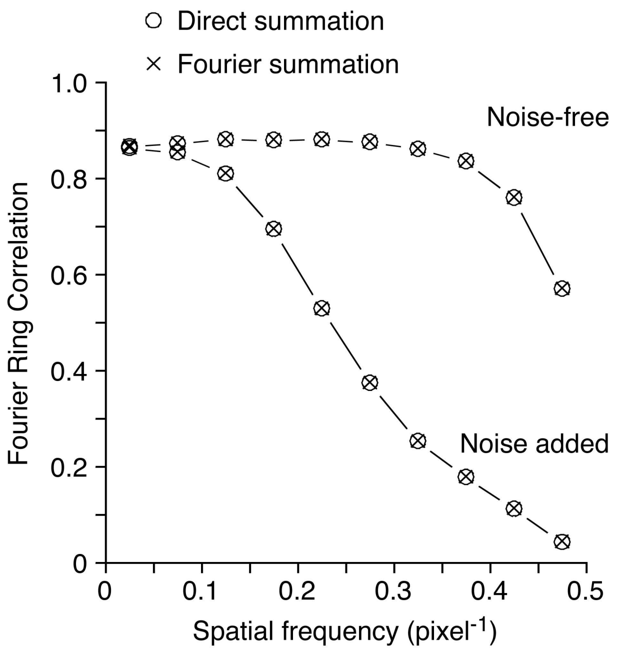


Figure E.7: The graph shows the correlation between the Fourier transforms of the reconstruction and those of the test volume, as a function of spatial frequency, averaged over 190 slices of each reconstruction.

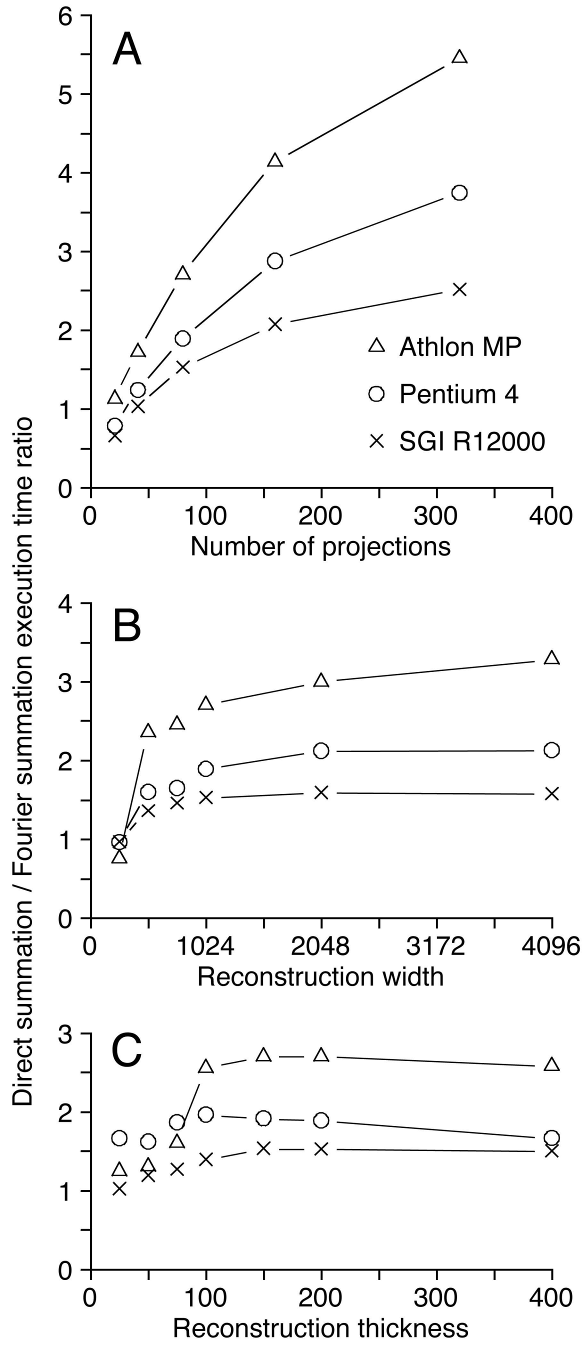


Figure E.8: (Direct summation)/(Fast Fourier Summation) execution time ratio for three computer architectures. (a): Dependence on number of projections with width 1024 and thickness 200. (b): Dependence on reconstruction width with 80 projections and thickness 200. (c): Dependence on thickness with 80 projections and width 1024.

dependence on one of the size dimensions with the other two held at typical values. The strongest dependence is on the number of projections (Fig. E.8a), where the speed benefit climbs 5-fold with an increase from 20 to 320 projections. For the other dimensions, the benefit from Fast Fourier Summation tends to rise with initial increases then flatten out. This initial rise was most abrupt and pronounced with Athlon processors (e.g., Fig. E.8c) and it reflects predominantly a slowing down of the direct summation per unit of computation rather than a speedup of Fast Fourier Summation. Our interpretation is that the architecture of the Athlons is particularly favorable to the direct summation for small data sizes, but at some point a limit in cache or pipeline size is reached and the performance falls abruptly for direct summation.

Overall, the typical benefit from Fast Fourier Summation is about 1.5 - 2.5 fold, with greater benefits available on some computers and with larger data sets. The Fast Fourier Summation is actually slower than direct summation for small data sets (Fig. E.8a,b). To avoid using a slower algorithm, the Tilt program switches to direct summation when the width, thickness, or number of projections falls below a specified limit for the given computer architecture.