Ac no edg ents

This thesis could not have b

Contents

on)Tj 37.7692 0 Td (of)3984 0 T-3d (5Tj 72.4375 0n)Tj 0 Td (e)Tj 9.473 0 Td (deriv)Tj 24.9363 0 Td 83tiv)Tj 19.413

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 e cient 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 signi cantly 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

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

$$(x)u_{tt} = (\ (x)u_x)_x \qquad (1.1)$$

with spatially varying material parameters (x) and (x). The coe cients (x) and (x) represent the compressibility and the speci c 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

In this thesis, we consider bandlimited functions restricted to an interval (see Section 2.5 for a precise de nition). 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 $\{e^{c\,\lambda_k x}\}_{k=1}^{N}$ where $|\lambda_k| < 1$. Since we do not impose that $\lambda_k = \lambda_k$, these functions are not necessarily periodic. The basis spanned by $\{e^{c\,\lambda_k x}\}_{k=1}^{N}$ 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,

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

$$, (t_k) = e^{\ tL}, (t_{k\ 1}); \tag{1.2}$$

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

The computation of $e^{\ tL}$ 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 signi cantly 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 $e^{\ tL}$ grows with the size of the time step t, 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 eld 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 Wubbeling [44] which involves repeated solution of equations on the form

$$\begin{aligned}
&u_{yy} = \ u_{xx} \quad (x;y)\,!\,^2(1 + f)u \quad Au;\ (x;y)\,2\,[\ \ 1;1] \quad [\ \ 1;1] \\
&u(x;\ \ 1) = g(x) \\
&u_y(x;\ \ 1) = h(x) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad : \\
&u(\ \ 1;y) = r(y) \\
&u(1;y) = s(y)
\end{aligned} \tag{1.3}$$

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 re ections. An alternative approach has been proposed in [44], where this problem is solved for constant coe cient by using Fourier-techniques to lter out the high frequencies of the solution. In this paper, we consider the case of x-dependent coe cient . We show that by forcing the spatial operator A to be negative de nite, we obtain an equation which can be solved in a stable manner by computing the matrix exponential. In order to obtain a negative de nite 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 coe cients.

1.2 Biological imaging

There is a signi cant interest in studying the ne 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 elds, 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 di culties 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 70^0, and the measurement usually contain a signi cant amount of noise. Also, the specimen is signi cantly 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 exibility than the commonly used technique ltered 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 ags `-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 e ciency 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 nite di erence 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 to than

w

and

$$\|u\|_1 = \max_{x\in[-1,1]} |u(x)|.$$

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

$$\|u\|_{L^2(\mathbb{R})} = \sqrt{\int_{-1}^{1} |u(x)|^2\,dx}$$

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 *Bandlimited functions* Let $c > 0$ and define the space

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

We refer to 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

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

In some situations, we will use the following space which is dense in B_c.

Definition Let $c > 0$ and define the space

$$F_c = B_c \cap L^1(\mathbb{R}).$$

As an example, consider the function

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

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, $v \in F_c$. Note that the function u in (2.2) is not absolutely integrable, and hence u does not belong to F_c.

The space B_c has the following properties.

Proposition The space of bandlimited functions B_c has the following properties:

1. Every function $u \in B_c$ 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 $u \in B_c$ and define $\varepsilon > 0$ by

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

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

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 de ned as

De nition Pro te spheroid e f nctions Consider the bandwidth $c > 0$ and de ne the operator $F_c : L^2([-1,1]) \to L^2([-1,1])$ by

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

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

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

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

Each eigenvalue of F_c corresponds to an eigenvalue of Q_c by

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

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 The prolate spheroidal wave functions are complete in $L^2([-1,1])$ and B_c.

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

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

and

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

where λ_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

formly

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

for all $j = 0;1;\ldots$. As discussed in [9], the existence of K

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 B_c. Following [9] we define

Definition (Bandlimited functions on an interval) Define the space E_c of bandlimited functions of bandwidth c on an interval as

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

We characterize the space E_c in the following theorem.

Theorem The space E_c of bandlimited functions on an interval satisfies the following properties:

1. $E_c \subset C^1([-1,1])$
2. For every $\varepsilon > 0$ and $u \in F_c$ there exists a function $v \in E_c$ such that $|u(x) - v(x)| < \varepsilon$ almost everywhere on $[-1,1]$.
3. For every $\varepsilon > 0$ and $u \in E_c$ there exists a function $v \in B_c$ such that $\|u - v\|_2 < \varepsilon$.
4. For every $\varepsilon > 0$ and $u \in B_c$ there exists a function $v \in E_c$ such that $\|u - v\|_2 < \varepsilon$.

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 $u \in E_c$. 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 $E_c \subset C([-1,1])$. By the definition of 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 $|(ib_k)^n| \le c^n$ it follows that $\{(ib_k)^n a_k\}_k \in l^1$ and hence $u^{(n)}(x) \in E_c$ for each $n = 0, 1, \ldots$. Since every function in E_c is continuous, it follows that $E_c \subset C^1([-1,1])$.

(2) By Proposition 3 it follows that any function $u \in F_c$ can be written on the form

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

almost everywhere and since $u \in L^1(\mathbb{R})$, $\hat{u}$ is con

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

(4) Let $u \in B_c$. Then, since F_c is dense in B_c, there exists a function $v \in F_c$ such that

$$\|u - v\|_2 < \varepsilon/2: \tag{2.9}$$

From the second part of the theorem we know that there exists $\breve{u} \in E_c$ such that $|v(x) - \breve{u}(x)| < \frac{\sqrt{2}}{2}\sqrt{\varepsilon}$ almost everywhere on $[-1;1]$. Then

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

which combined with (2.9) gives us that

$$\|u - \breve{u}\|_2 \leq \|u - v\|_2 + \|v - \breve{u}\|_2 < \varepsilon:$$

□

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 $e^{ic\mu_k x}$ where $|\mu_k| \leq 1$. The phases μ_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. Let ω be a real, non-negative, integrable weight function supported in $[-\alpha;\alpha]$, $0 < \alpha < 1/2$, and let

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

The constants $_m$ 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 di erent algorithm where a large number of the weights are su ciently close to zero to be disregarded (see [9] for more details). Therefore, the number of nodes and weights actually used is typically signi cantly 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.

Coro ry Let c and be positive numbers. Then, for N su ciently large, there exist constants $fw_1; : : : ; w_N g$ and $f_1; : : : ; _n g$, such that for any $x 2 [1; 1]$

$$\int_{-1}^{1} e^{ictx}\, dt \approx \sum_{k=1}^{N} w_k e^{ic_k x}$$

to the prolate spheroidal wave functions using the functions $e^{ic_k x}$, 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

$$\alpha_k = -\alpha_{N-k+1} \tag{2.11}$$

and

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

We use the quadrature nodes and weights to construct a basis for E_c according to the following theorem.

Theorem Consider the bandwidth c and a function $u \in E_c$ represented by

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

Let $\varepsilon > 0$, and let $\{\alpha_l\}_{l=1}^N$ 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

$$\left\| u(x) - \sum_{l=1}^{N} u_l e^{ic\alpha_l x} \right\|_1 \leq A \left(\sum_{k \in \mathbb{Z}} |a_k| \right) \varepsilon$$

and

$$\left\| u(x) - \sum_{l=1}^{N} u_l e^{ic\alpha_l x} \right\|$$

bounded. The sequence $\{a_k g_k\} \in l^1$ by assumption and, therefore, $u_l = \sum_{k\in\mathbb{Z}} a_k \, \alpha_{kl}$ is well-defined for $l = 1, \ldots, N$. It follows that

$$\left\| u(x) - \sum_{l=1}^{N} u_l e^{ic_l x} \right\|_1 = \left\| \sum_{k\in\mathbb{Z}} a_k e^{ib_k x} - \sum_{k\in\mathbb{Z}} a_k \sum_{l=1}^{N} \alpha_{kl} e^{ic_l x} \right\|_1$$

$$= \left\| \sum_{k\in\mathbb{Z}} a_k \left(e^{ib_k x} - \sum_{l=1}^{N} \alpha_{kl} e^{ic_l x} \right) \right\|_1 \leq A \sum_{k\in\mathbb{Z}} |a_k| \, .$$

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

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

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

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

and hence

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

□

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 nite but arbitrary precision. Even though the

2.6.2 Approximate prolate spheroidal wave functions

In the last section we pointed out that a disadvantage of the exponential

the initial approximate prolate spheroidal wave functions approximates well the correspond-

De ne q^j as the vector with elements given by $q^j_m = \sqrt{w_m}\ _j(\ _m$

Hence, if N is even, we have that rank(A_r) N=2 and by the same argument rank(A_i) N=2.

Assume that there exists an eigenvector q such that $q \in N(A_r) \setminus N(A_i)$. Then = = = 0 which contradicts that 6= 0. Hence, if $q \in N(A_r)$, then $q \in R(A_i)$. Therefore,

$$N(A_r) \quad R(A_i): \tag{2.15}$$

Assume that rank(A_r) < N=2. Then dim($N(A_r)$) N=2 + 1 which by (2.15) implies that rank(A_i) N=2 + 1 which is a contradiction. Hence, rank(A_r) = N=2 and by the same argument it follows that rank(A_i) = N=2. Since dim($N(A_r)$) = rank(A_i) = N=2 it follows from (2.15) that $N(A_r) = R(A_i)$. (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 $_m = \ _{N\ m+1}$. Using the symmetry of the nodes $_m$ it follows that

$$A_{N\ m+1;l} = \sqrt{w_{N\ m+1}}e^{ic\ _{N-m+1}\ _l}\sqrt{w_l} = \sqrt{w_m}e^{\ ic\ _m\ _l}\sqrt{w_l} = \overline{A_{ml}}: \tag{2.16}$$

Therefore, since q^j is an eigenvector of A with eigenvalue ,

$$q^j{}_{N\ m+1} = \frac{1}{\ _j}\sum_{l=1}^{N} A_{N\ m+1;l}q^j{}_l = \frac{1}{\ _j}\sum_{l=1}^{N}\overline{A_{ml}}q^j{}_l: \tag{2.17}$$

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

$$q^j{}_{N\ m+1} = \frac{\overline{\ _j}}{\ _j}q^j{}_m = \ \ q^j{}_m:$$

We now have that

$$_j(\ _{N\ m+1}) = \sqrt{\frac{1}{w_{N\ m+1}}}q^j{}_{N\ m+1} = \ \ \sqrt{\frac{1}{w_m}}q^j{}_m = \ \ _j(\ _m): \tag{2.18}$$

□

Coro ry The eigenfunctions $_j(x)$ de ned in De nition 14 are even or odd.

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

and

$$S_{ij} = \begin{cases} 2 & \text{if } i = N \quad j + 1 \\ 2\frac{\sin(c(\ _i + \ _j))}{c(\ _i + \ _j)} & \text{if } i \neq N \quad j + 1 \end{cases}: \tag{2.24}$$

Using these matrices it follows from (2.14) that

$$_j(x) = H_{jj} \sum_{l=1}^{N} W_{ll} Q_{lj} e^{icx\ _l}: \tag{2.25}$$

The inner product matrix S de ned in (2.20) is

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

Bandwidth c	Number of nodes N	Condition number
4	21	1:48
8	31	2:11
12	40	2:06
16	49	2:34
20	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 $E_{c;}$ de ned in De nition 13. By de ning the matrix $B = HQ^T W$ we see that from (2.25) we have that

$$_j(x) = \sum_{l=1}^{N} B_{jl} e^{ic\ _l x}:$$

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

$$\det(B) = \det(H)\det(W) = \prod_{l=1}^{N} \frac{p}{w_l}:$$

Since $_l$ 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 $E_{c;}$ de ned in De nition 13.

2.6.3 Interpolating functions

In many applications, it is convenient to work with function values of a function rather than with expansion coe cients 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 coe cients 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 de ne the interpolating basis functions for bandlimited functions on an interval as follows.

De nition B sis of interpo ting f nctions De ne the matrices Q, W, and H according to (2.21)-(2.23), and the matrix $R = WQHQ^T W$. The sequence of functions

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

for k = 1; : : : ; N is

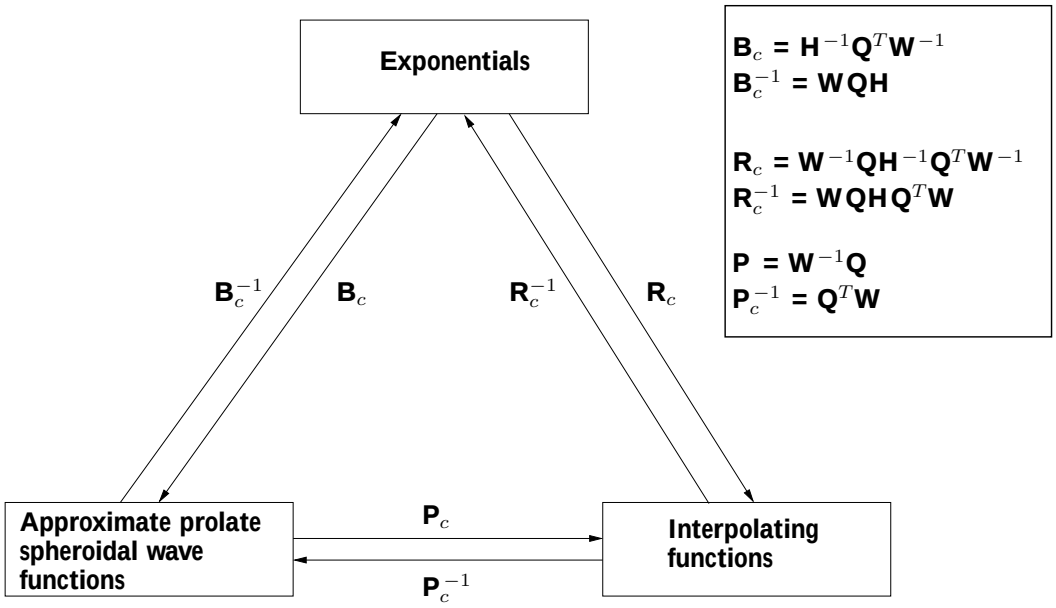


Figure 2.4: Transformation matrices between three di erent bases for bandlimited functions on an interval. The matrices H, Q, and W are de ned in (2.21)-(2.23). Note that the matrices H and W are diagonal.

Table 2.2: Condition number for transformation matrices for the bandwidth c = 8:5 . The accuracy = 10 7 requires 32 nodes and the accuracy = 10 14 requires 41 nodes.

Transformation matrix	= 10 7	= 10 14
$\mathbf{P}_c$	2.7	3.5
$\mathbf{B}_c$	1:1 10^8	2:5 10^{14}
$\mathbf{R}_c$	1:2 10^8	3:1 10^{14}

Table 2.3: Condition number for transformation matrices for the bandwidth c = 17 . The accuracy = 10 7 requires 51 nodes and the accuracy = 10 14 requires 62 nodes.

Transformation matrix	= 10 7	= 10 14
$\mathbf{P}_c$	2.8	3.8
$\mathbf{B}_c$	1:2 10^8	3:4 10^{14}
$\mathbf{R}_c$	1:3 10^8	4:0 10^{14}

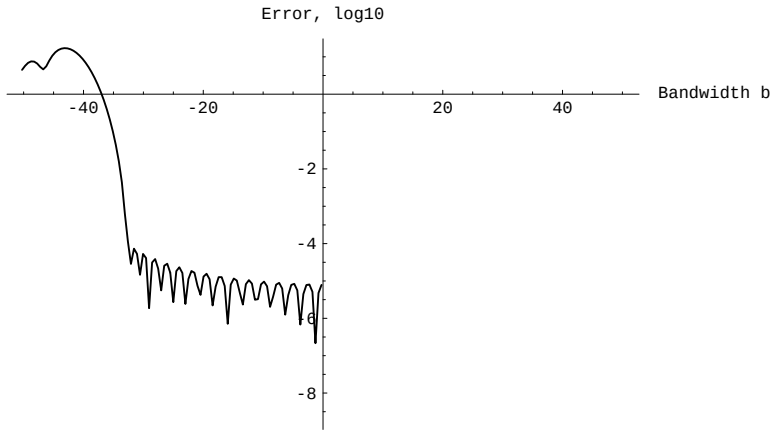
and

$$f(x) \approx \sum_{k=1}^{N} \alpha_k \, \psi_k(x) \tag{2.33}$$

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, $c = 5.5, 7, 8.5$, and 10.5. 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



where

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

This transformation matrix does not contain the greatly varying factor $\frac{1}{\lambda_j}$ 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

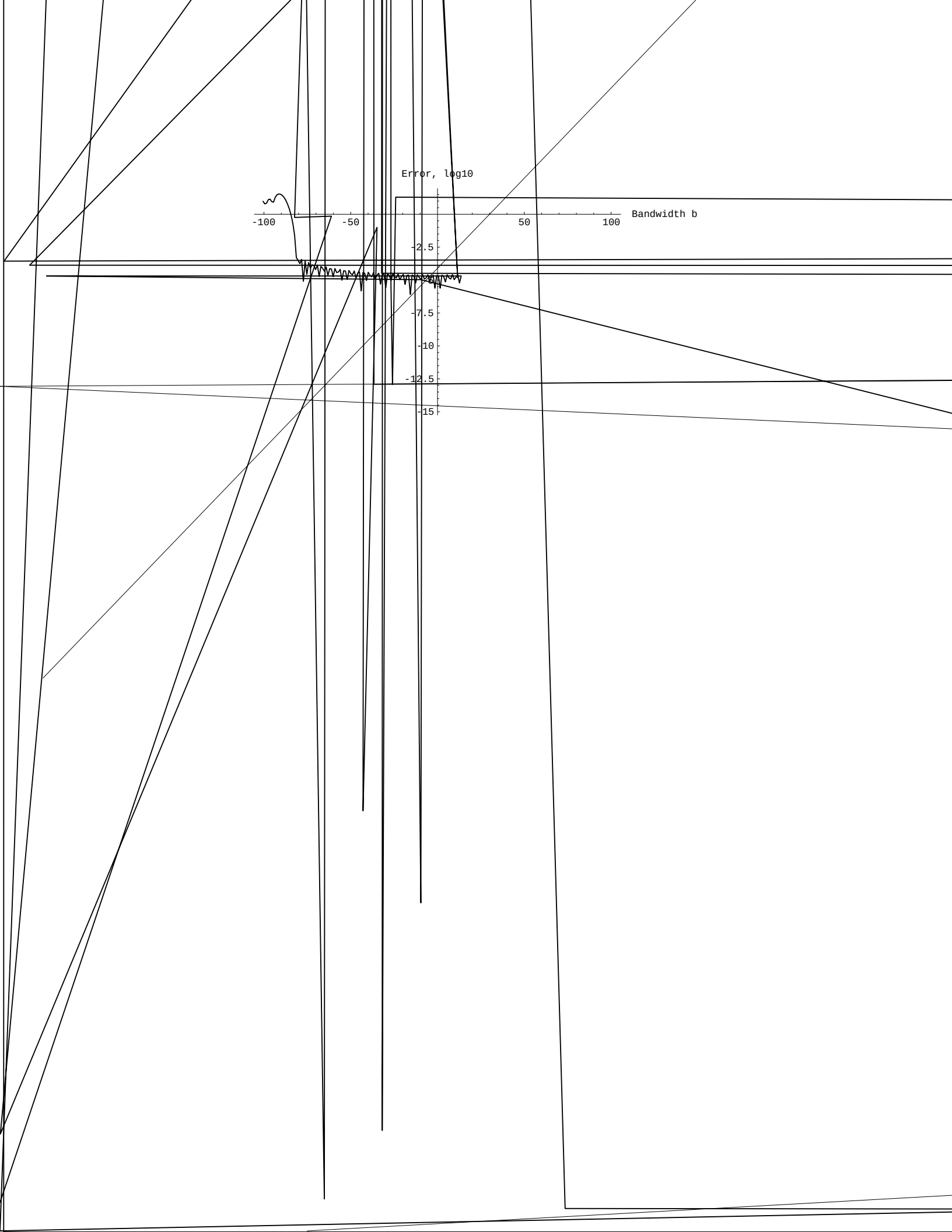
$$\psi_k(x) = \sum_{n=1}^{N} (H Q^T W)_{kn} e^{ic_n x}: \tag{2.36}$$

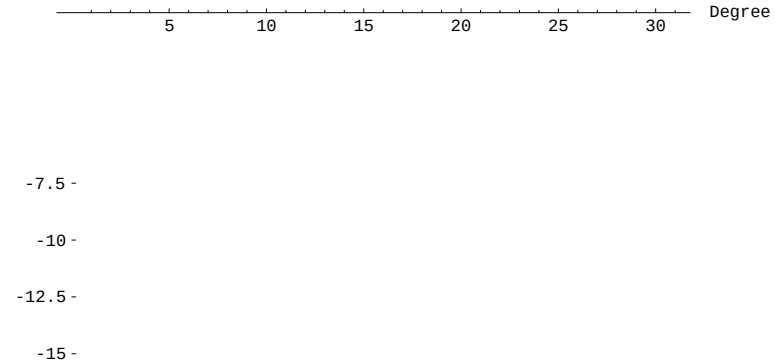
Since

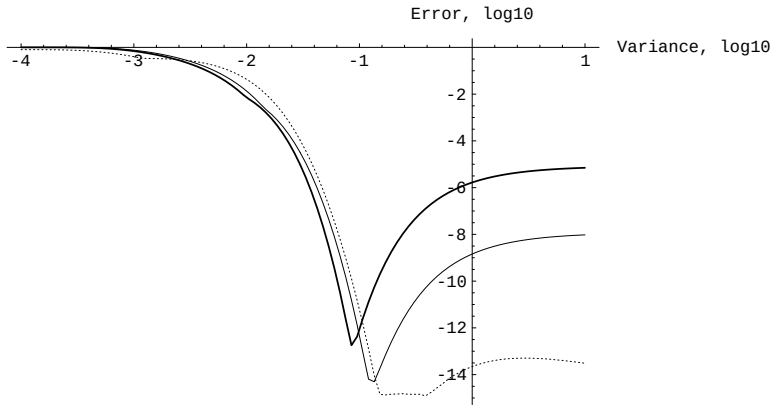
$$(H Q^T W)_{kn} = \frac{1}{\lambda_k} Q_{nk} \sqrt{w_n}$$

we see that the summation in (2.36) is multiplied by a factor $\frac{1}{\lambda_k}$. For large k, the denominator λ_k, and hence the computation of ψ_k for large k may contain large errors. On the other hand, the computation of the expansion coefficients α_k is stable and for functions of the type e^{ibx}, equation (2.8) shows that the expansion coefficients $\alpha_k \approx \lambda_k$. 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.


Error, log10
Bandwidth b
-100
-50
50
100
-2.5
-7.5
-10
-12.5
-15


5
10
15
20
25
30
Degree
-7.5
-10
-12.5
-15


Error, log10
Variance, log10
-4
-3
-2
-1
1
-2
-4
-6
-8
-10
-12
-14

Chapter 3

Derivative matrices with boundary and interface conditions

When solving ordinary and partial di erential equations using spectral methods, we expand the solution into a set of basis functions which we can di erentiate 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 coe cients. We decompose the domain into a collection of subdomains such that v

De ne G as the linear span of $f_i(x)g_{i=1}^N$ equipped with the inner product

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

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

$$S_{kl} = (\ _l(x);\ _k(x))\ ;$$

$$K_{kl} = \quad _l(x);\frac{d\ _k(x)}{dx}\quad ;$$

$$E_{kl} = \overline{\ _k(\ 1)}\ _l(\ 1);$$

$$F_{kl} = \overline{\ _k(1)}\ _l(1);$$

and

$$G_{kl} = \overline{\ _k(1)}\ _l(\ 1):$$

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

$$K_{kl} = \int_{-1}^{1} {}_l(x)\overline{\frac{d}{dx}\ _k(x)}\,dx = \ _l(1)\overline{\ _k(1)}\quad \ _l(\ 1)\overline{\ _k(\ 1)}\quad \int_{-1}^{1}\overline{\ _k(x)}\frac{d}{dx}\ _l(x)\,dx$$

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

and hence

$$K = F\quad E\quad K\ : \tag{3.1}$$

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

3.1 Derivativ

for some set of coe cients s_l. We seek coe cients s_l such that

$$\frac{du}{dx} = \sum_{l=1}^{N} s_l \, _l(x): \tag{3.2}$$

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

$$\begin{aligned} \int_{-1}^{1} \frac{du}{dx} \overline{_k(x)} \, dx &= \sum_{l=1}^{N} s_l \int_{-1}^{1} \, _l(x) \overline{_k(x)} \, dx \\ &= \sum_{l=1}^{N} S_{kl} s_l: \end{aligned} \tag{3.3}$$

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

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

Combining (3.3) and (3.4), we obtain

$$\sum_{l=1}^{N} S_{kl} s_l = u(1) \overline{_k(1)} - u(-1) \overline{_k(-1)} - \sum_{l=1}^{N} K_{kl} s_l: \tag{3.5}$$

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

$$\begin{aligned} \sum_{l=1}^{N} S_{kl} s_l &= \sum_{l=1}^{N} s_l \, _l(1) \overline{_k(1)} - \sum_{l=1}^{N} s_l \, _l(-1) \overline{_k(-1)} - \sum_{l=1}^{N} K_{kl} s_l \\ &= \sum_{l=1}^{N} F_{kl} s_l - \sum_{l=1}^{N} E_{kl} s_l - \sum_{l=1}^{N} K_{kl} s_l: \end{aligned} \tag{3.6}$$

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

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

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 di erentiable, 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 di erentiation 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 de ne the derivative matrices D_0^+, D_0^-, and D_0 as

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

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

and

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

These matrices correspond to the boundary condition u(1) = 0, u(1) = 0, and u(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),

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

and

$$D_2 = S^{-1}(F \quad G \quad K):$$

Let us de ne D_p as

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

Using (3.1) we nd that

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

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

De nition Let E, F, G, S, and K be the matrices in De nition 19. We de ne 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

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

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

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

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

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

and

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

If $a \neq -1$ 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

$$I_l = [x_l, x_{l+1}] = [-3+2l, -1+2l],$$

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 G defined in Definition 19 and define $\{\phi_{kl}(x)\}_{k,l}$ for $k = 1, \ldots, N$ and $l = 1, \ldots, M$ by

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

Let 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 $\phi_{kl}(x_l) = \phi_k(-1)$ and $\phi_{kl}(x_{l+1}) = \phi_k(1)$. It follows that

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

and

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

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

Let the function $f(x) 2 G_M$ be written as

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

for some set of coe cients s_{il}. We note that for each interior interface x_l, there are two possible expansions available, namely from the left

$$f(x_l) = \sum_{i=1}^{N} s_{i;l\ 1}\ {}_{i}(1) \tag{3.16}$$

and from the right

$$f(x_l) = \sum_{i=1}^{N} s_{il}\ {}_{i}(\ 1) : \tag{3.17}$$

These two expansions correspond to taking the limit from the respective direction.

From (3.19) and (3.15) we get

$$\int_{x_l}^{x_{l+1}} \frac{df}{dx}\,\overline{\phi}_{il}(x)\,dx = f(x_{l+1})\,\overline{\phi}_i(1) \quad f(x_l)\,\overline{\phi}_i(\quad 1)$$

$$\sum_{l'=0}^{N}\sum_{j=1}^{N} s_{jl'} \int_{x_l}^{x_{l+1}} {}_{jl'}\frac{d\overline{\phi}_{il}}{dx}\,dx$$

$$= f(x_{l+1})\,\overline{\phi}_i(1) \quad f(x_l)\,\overline{\phi}_i(\quad 1)$$

$$\sum_{j=1}^{N} s_{jl} \int_{x_l}^{x_{l+1}} {}_{jl}\frac{d\overline{\phi}_{il}}{dx}\,dx$$

$$= f(x_{l+1})\,\overline{\phi}_i(1) \quad f(x_l)\,\overline{\quad\quad}$$

$$(1)^{N}$$

1)

and

$$\frac{df(x_l)}{dx} = \sum_{i=1}^{N} s_{il}\ \ _i(\ \ 1): \tag{3.24}$$

Unless f is di erentiable, the derivative $\frac{df}{dx}$ is not well-de ned at the interface points. If we impose that $f \in C^1(I) \setminus G_M$, then $\frac{df}{dx}$ is well de ned 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 de ne $s_l = [s_{1l}; s_{2l}; \ldots; s_{Nl}]^T$ and $s_l = [s_{1l}; s_{2l}; \ldots; s_{Nl}]^T$ for $l = 1; \ldots; M$ where the coe cients s_{il} and s_{il} are the expansion coe cients 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 rst consider arbitrary boundary conditions at the left end point. We have that

$$f(\ \ 1) = \sum_{i=1}^{N} s_{i0}\ \ _i(\ \ 1):$$

When describing f(1), the right end point of the rst 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 $a \in [0; 1]$,

$$f(1) = \sum_{i=1}^{N} (1 \quad a)s_{i1}\ \ _i(1) + as_{i2}\ \ _i(\ \ 1):$$

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

$$\sum_{j=1}^{N} S_{ij}s_{j1} = \sum_{j=1}^{N} (1 \quad a)s_{j1}\ \ _j(1) + as_{j2}\ \ _j(\ \ 1) \left(\overline{\ \ _i(1)} \right) \sum_{j=1}^{N} s_{j1}\ \ _j(\ \ 1) \left(\overline{\ \ _i(\ \ 1)} \right)$$
$$\sum_{j=1}^{N} K_{ij}s_{j1}$$

or, equivalently using matrix-vector notation,

$$Ss_1 = ((1 \quad a)F \quad E \quad K)\, s_1 + aGs_2 \tag{3.25}$$

where we used the matrices de ned in De nition 19. If f(1) = 0, then (3.25) reduces to

$$Ss_1 = ((1 \quad a)F \quad K)\, s_1 + aGs_2: \tag{3.26}$$

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

$$f(x_{M+1}) = \sum_{i=1}^{N} s_{iM}\ _i(1):$$

In describing $f(x_M)$, 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

$$f(x_M) = \sum_{i=1}^{N} (1 - b)s_{iM}\ _i(-1) + bs_{i;M-1}\ _i(1):$$

Inserting the expansions for $f(x_{M+1})$ and $f(x_M)$ into (3.22) we have

$$\sum_{i=1}^{N} S_{ij}s_{jM} = \sum_{j=1}^{N} s_{jM}\ _j(1) \left(\overline{\ _i(1)} \right.$$
$$\left. \sum_{j=1}^{N} (1 - b)s_{jM}\ _j(-1) + bs_{j;M-1}\ _j(1) \right) \overline{\ _i(-1)}$$
$$\sum_{j=1}^{N} K_{ij}s_{jM}$$

or, equivalently, in matrix-vector notation,

$$S\mathbf{s}_M = -bG\ \mathbf{s}_{M-1} + (F + (b - 1)E - K)\mathbf{s}_M: \tag{3.27}$$

If $f(x_{M+1}) = 0$ then the expression in (3.27) reduces to

$$S\mathbf{s}_M = -bG\ \mathbf{s}_{M-1} + ((b - 1)E - K)\mathbf{s}_M: \tag{3.28}$$

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:

$$f(x_l) = \sum_{j=1}^{N} (1 - b)s_{j;l}\ _j(-1) + bs_{j;l-1}\ _j(1)$$
$$f(x_{l+1}) = \sum_{j=1}^{N} (1 - a)s_{j;l}\ _j(1) + as_{j;l+1}\ _j(-1):$$

Using this expression in (3.22) yields

$$\sum_{j=1}^{N} S_{ij} s_{jl} = \sum_{j=1}^{N} (1 - a) s_{j;l-1} \overline{\phi_j(1)} \overline{\phi_i(1)} + a s_{j;l+1} \phi_j(-1) \overline{\phi_i(1)} + \sum_{j=1}^{N} (1 - b) s_{j;l-1} \phi_j(-1) \overline{\phi_i(-1)} + b s_{j;l-1} \phi_j(1) \overline{\phi_i(-1)} \Bigg)$$

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 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.

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

Ch r cter iz tion	a	b	Bo nd ry 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	$f(x_{M+1}) = 0$	0	0	S^{-1}

(37 4.8 0 0 -145.2 2059.08 5393.88 cm BI /IM true /W 1 /H 1 /BPC 1 ID EI 0 g 10 0 0 10 0 0 cm i /R197 427ue /W

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 De nition 20 we have that K di erentiates a function at a subinterval. Assuming eigenfunctions on the form (3.33), 25 0 Td (w)Tj 8.03252 0 Td3 Td (diagonal)Tj 47.

4.1.1 Construction of derivative matrices

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

$$K = icHQ^T W \quad SWQH: \tag{4.1}$$

where H, Q, S, and W are de ned in (2.21)-(2.24). Let us summarize the steps to construct derivative matrices in following algorithm.

A gorith Constr ction of deri ti e trices ith respect to nd ited f nc tions

1. **Construct N quadrature nodes and weights according to De nition 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 De nition 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 De nition 20, and for multiple intervals by using (3.32) and Tables 3.1-3.3.**
4. **Compute $D = P_c D P_c^{\ 1}$ to represent the derivative matrix with respect to function values.**

4.1.2 Accuracy of the derivative matrix for varying bandwidth

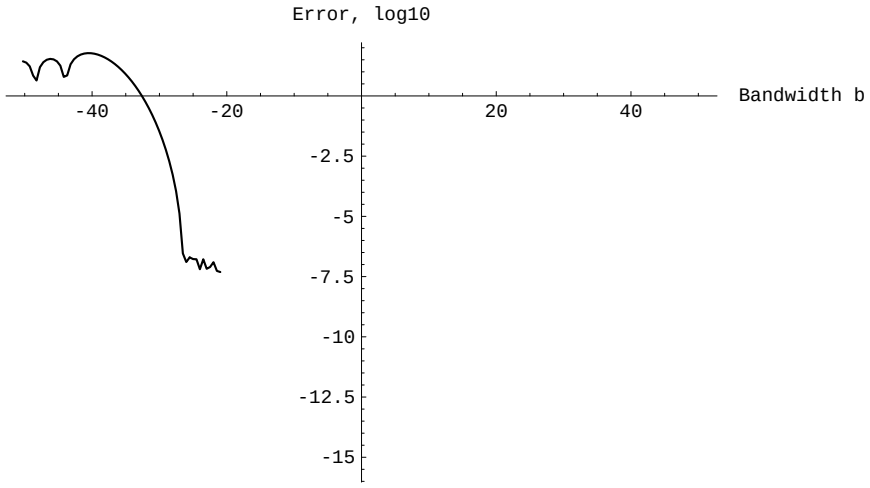
Using the algorithm above, we present two graphs that alcuracy

We note from Figure 4.1 that the error pro le is almost uniform for $|j| \leq c$. 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 32 (for periodic functions). We construct four derivative matrices with the bandwidth c set to 18:5 , 20:5 , 23 , 8

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 nite di erence 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 16 . We construct two derivative matrices with respect to approximate prolate spheroidal wave functions. We construct one derivative matrix with the accuracy = 10 ^{7} and bandwidth c = 8:5 , and another with the accuracy = 10 ^{13} and bandwidth c = 5:5 . For comparison, we construct a second order central nite di erence derivative matrix using a second order boundary stencil for the rst and the last row of the matrix. Finally, we construct a spectral derivative matrix with respect to the rst 32 Chebyshev polynomials using the algorithm in [25, Appendix C]. We di erentiate the function f(x) = sin(bx) for 200 values of b ranging between 16 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.



We next consider an experiment

interface points x

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

$$\begin{aligned} Du &= \lambda u \\ u(-1) &= u(1) \end{aligned} : \tag{4.3}$$

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 $k \quad c$. For k such that $\frac{c}{\pi} < k \quad N$ 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 $k > \frac{c}{\pi}$

Then the projected matrix D is given by

$$D = \sum_{|k| \le c} \alpha_k P_k.$$

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

$$D_{proj} = \sum_{|k| \le c},$$

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 de ne the linear operator L_0 as $L_0 = DD_0$ and consider the eigenvalue problem

$$\begin{aligned} L_0 u &= \quad u \\ u(\ \ 1) &= u(1) = 0 \end{aligned} \quad : \tag{4.4}$$

It is easily seen that

$$u_k(x) = \sin \quad \frac{k}{2}(x + 1)$$

for $k = 1; 2; : : :$ are eigenfunctions to (4.4) with the corresponding

2. Construct the matrix S according to (2.27), the matrix K according to (4.1), and the matrices E, F, and G de ned in De nition 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 De nition 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 De nition 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

$$L_{proj} = \sum_{|j-k|\geq c} \lambda_k e_k f_k^T$$

where e_k and f_k are the left and the right eigenvector of L_0, respectively, scaled such that $f_k^T e_k = 1$.

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 rst experiment we use 32 nodes. Note that the Nyquist frequency for 32 nodes corresponds to the bandwidth c = 16 (for periodic functions). We construct four derivative matrices with the bandwidth c set to 5:5 , 7 , 8:5 , and 10:5 , respectively. In order to obtain these bandwidths using 32 nodes, we set the accuracy to 10^{-13}, 10^{-10}, 10^{-7}, and 10^{-4}, respectively. We di erentiate the function $f(x) = \sin b_k x$ where b_k corresponds th39 1[illegible] and (fo[illegible]

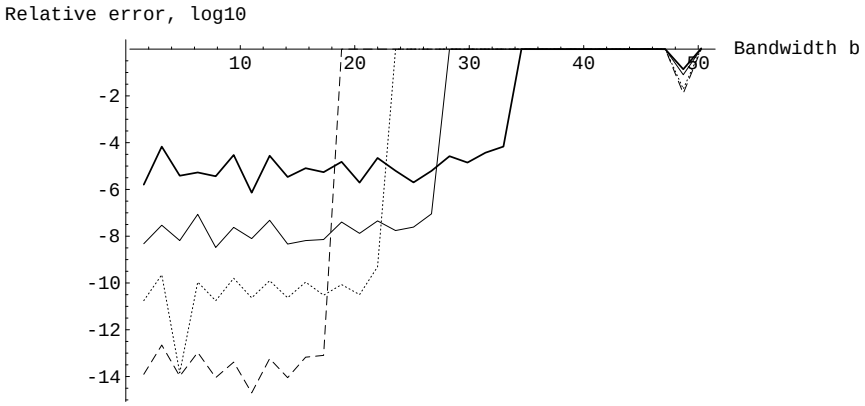


Figure 4.8: Relative error ($\log_{10}$) for the second derivative of the function $\sin b_k x$ where $b_k = k = 2$ for $k = 1; \ldots; 32$. The derivative matrices are constructed with respect to 32 basis functions. We use a basis of approximate prolate spheroidal wave functions with maximum bandwidth $c = 10{:}5$ (thick solid curve), $c = 8{:}5$ (thin solid curve), $c = 7$ (dotted curve), and $c = 5{:}5$ (dashed curve). The error of each differentiation is measured at 32 equally spaced points (including the end points) on the interval $[-1; 1]$.

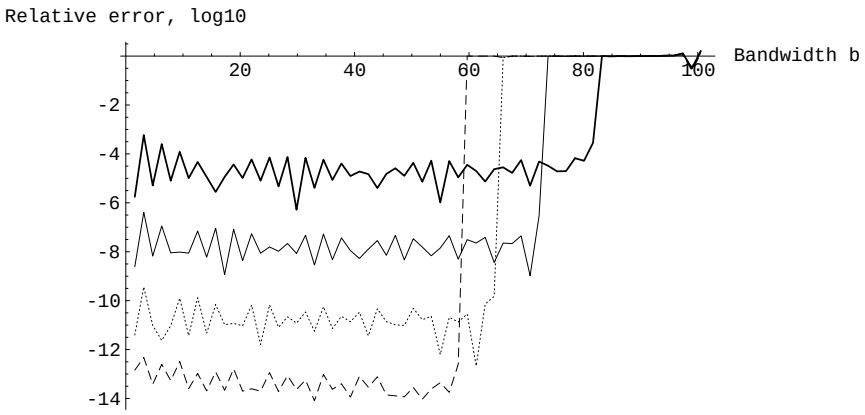


Figure 4.9: Relative error ($\log_{10}$) for the second derivative of the function $\sin b_k x$ where $b_k = k = 2$ for $k = 1; \ldots; 64$. The derivative matrices are constructed with respect to 32 basis functions. We use a basis of approximate prolate spheroidal wave functions with maximum bandwidth $c = 26$ (thick solid curve), $c = 23$ (thin solid curve), $c = 20{:}5$ (dotted curve), and $c = 18{:}5$ (dashed curve). The error of each differentiation is measured at 64 equally spaced points (including the end points) on the interval $[-1; 1]$.

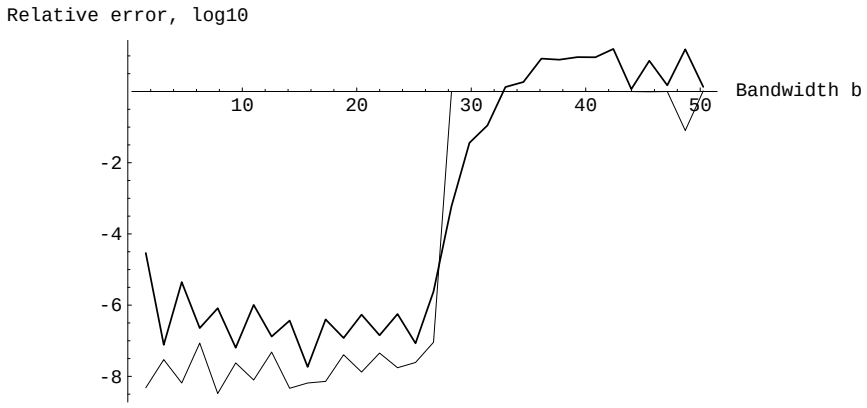


Figure 4.10: Relative error (log e error

4.4.1 Construction of the integration matrix

Let $\{x_k\}_{k=1}^N$ 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 T with elements defined by

$$T_{kl} = \int_{-1}^{x_k} e^{ic\theta_l x}\,dx = \begin{cases} \frac{e^{ic\theta_k\theta_l} - e^{-ic\theta_l}}{ic\theta_l}, & k \neq \frac{N+1}{2} \\ x_k + 1, & k = \frac{N+1}{2} \end{cases} \quad (4.5)$$

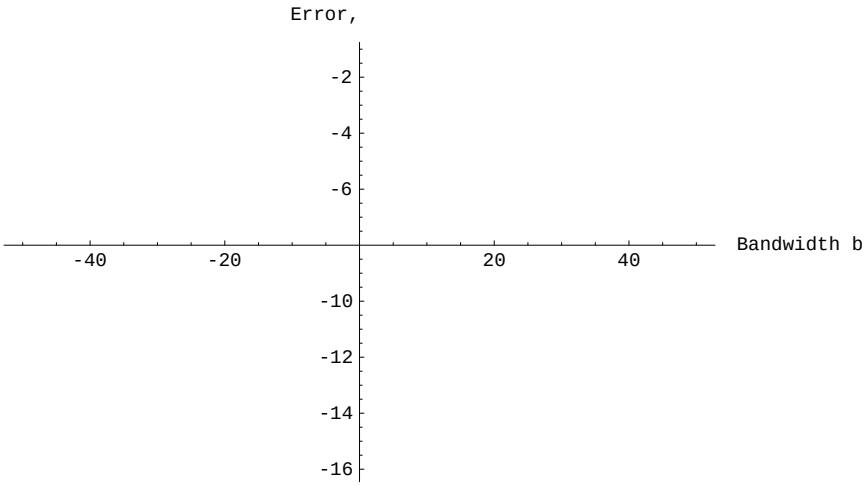
for $k, l = 1, \ldots, N$. Note that if N is even, only the first case applies. Using (2.14) we find that

$$\int_{-1}^{x_k} \phi_l(x)\,dx = H_{kk} \sum_{m=1}^{N} W_{mm} Q_{ml} T_{km}$$

The integration matrix based on approximate

In the second experiment we use 64 nodes corresponding to the Nyquist frequency for $c = 32$ (for periodic functions). We construct four integration matrices with the bandwidth c set to 18:5 , 20:5 , 23 , and 26 , respectively. In order to obtain these bandwidths using 64 nodes, we set the accuracy to 10^{-13}, 10^{-10}, 10^{-7}, and 10^{-4}, respectively. Let $\{\int_{-1}^{x_k} g\}_{k=1}^{64}$ be a set of quadrature nodes for bandlimited functions. We compute the integrals $\int_{-1}^{x_k} e^{ibx}\,dx$ for 200 values of b ranging between 32 . Note that this includes non-periodic functions. The result is shown in Figure 4.12.

We note that the error behaves in the result m5.91ces


Error,
-2
-4
-6
-40
-20
20
40
Bandwidth b
-10
-12
-14
-16

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.

$O(N^4)$ 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

$$= D_{xx} \quad I + I \quad D_{yy}$$

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 $L : \mathbb{C}^N \quad ! \quad \mathbb{C}^N$ be a linear operator. Let $fA_k g_{k=1}^r$ and $fB_k g_{k=1}^r$ be N-by-N matrices, and $fs_k g_{k=1}^r$ be scalars, such that

$$L = \sum_{k=1}^{r} s_k A_k \quad B_k:$$

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

This de nition is speci c for two dimensions, and has no analogue in higher dimensions. In many applications, it su ces to nd an approximation to L in a separated form. Therefore we will use

De nition ep r ted represent tion For the accuracy > 0, we de ne the separated rank representation of an operator L as

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

such that

$$kL \quad Lk < :$$

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 $r \sim r$.

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

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

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 $u \in \mathbb{C}^N$ 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 $L_2 = \sum_{k=1}^{r} s_k^{(2)} A_k^{(2)} \otimes B_k^{(2)}$. We compute linear combinations by

$$L_1 + L_2 = \sum_{k=1}^{r_1+r} s_k A_k \otimes B_k; \tag{5.4}$$

where

$$\{s_k\}_{k=1}^{r_1+r} = \{s_1^{(1)}, \ldots, s_{r_1}^{(1)}, s_1^{(2)}, \ldots, s_r^{(2)}\};$$

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

and

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

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

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

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

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

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

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

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 $U_k = \mathrm{diag}(u_k)$ and $V_k = \mathrm{diag}(v_k)$. □

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

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

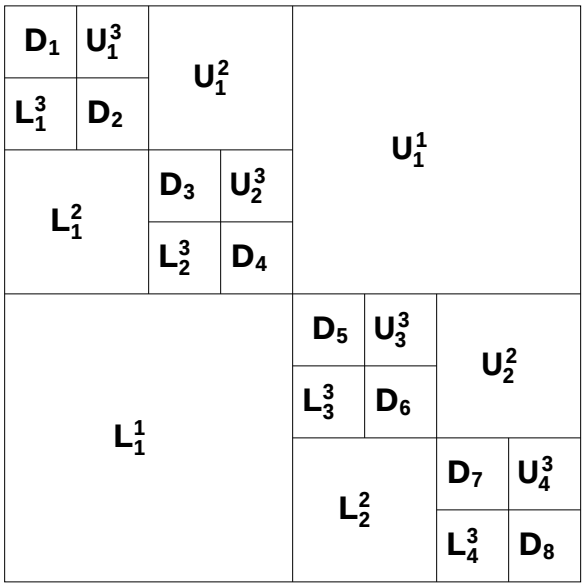
We refer to this partition as a level one PLR representation of A.

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;



D_l are dense matrices

$$U_l^k = \sum_{i=1}^{r_{kl}^{U)}} (\ _{kl})_i (e_{kl})_i (f_{kl})_i$$

$$L_l^k = \sum_{i=1}^{r_{kl}^{L)}} (\ _{kl})_i (g_{kl})_i (h_{kl})_i$$

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

the PLR representation, the coe cients s_k in the separated representation of the o -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

□

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

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

matrix-vector products of the type $L_l^k u$ and $U_l^k u$ where $u \in \mathbb{C}^{N_k}$, and L_l^k and U_l^k are N_k-by-N_k matrices given as separated representations on the form $\sum_i s_i e_i f_i$. 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 $L_2 = \sum_{k=1}^{r} s_k^{(2)} e_k^{(2)} f_k^{(2)^*}$. We compute a linear combination by

$$L_1 + L_2 = \sum_{k=1}^{r_1+r} s_k e_k \tilde{f}_k \tag{5.9}$$

where

$$\{s_k\}_{k=1}^{r_1+r} = \{ s_1^{(1)}, \ldots, s_{r_1}^{(1)}, s_1^{(2)}, \ldots, s_r^{(2)} \},$$

$$\{e_k\}_{k=1}^{r_1+r} = \{e_1^{(1)}, \ldots, e_{r_1}^{(1)}, e_1^{(2)}, \ldots, e_r^{(2)}\},$$

and

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

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

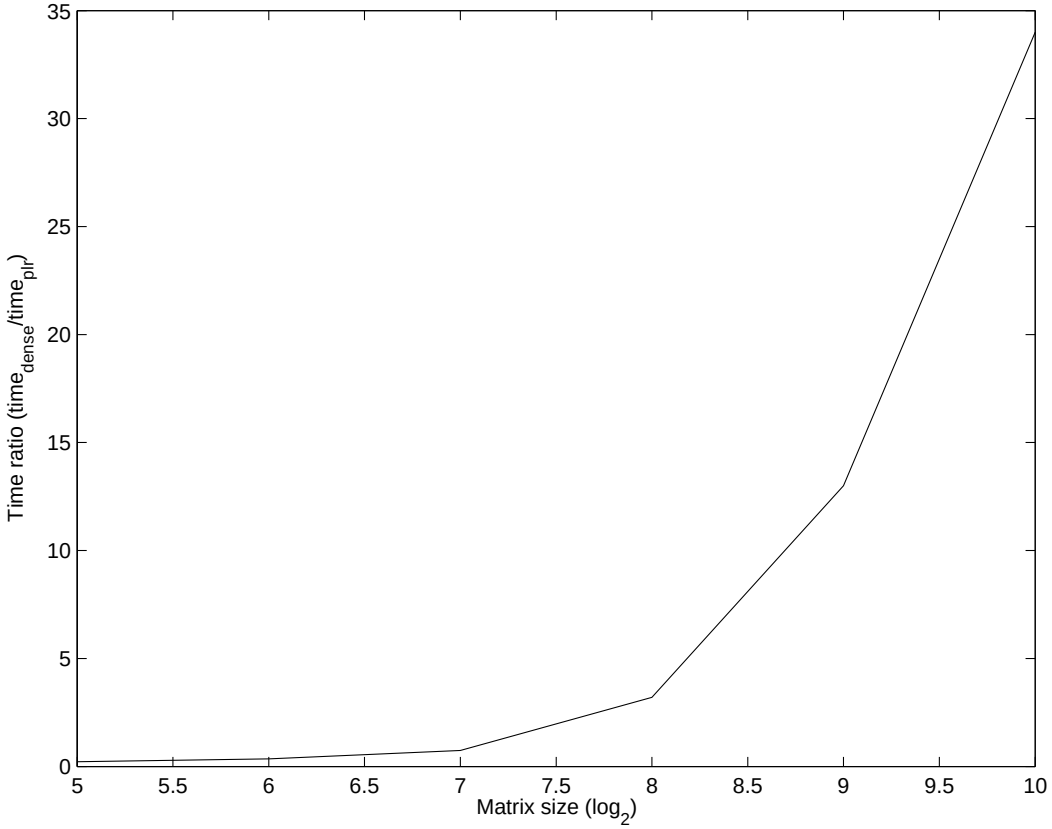


Figure 5.3: Time ratio for computing the square of the matrix in (5.7) using dense multiplication compared with PLR multiplication.

Table 5.1: Compression using the PLR representation. The The The The

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

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

where D is a rectangle. The function $\kappa(x,y)$ is the compressibility of the medium, and the function $\rho(x,y)$ 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 form

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

$$\begin{bmatrix} v \\ w \\ u \end{bmatrix}_t = \begin{bmatrix} 0 & 0 & \rho(x,y)[\frac{\partial^b}{\partial x} \otimes I_y] \\ 0 & 0 & \rho(x,y)[I_x \otimes \frac{\partial^b}{\partial y}] \\ \frac{1}{\kappa(x,y)}[\frac{\partial}{\partial x} \otimes I_y] & \frac{1}{\kappa(x,y)}[I_x \otimes \frac{\partial}{\partial y}] & 0 \end{bmatrix} \begin{bmatrix} v \\ w \\ u \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \int_0^t F(x,y,\tau)\, d\tau \end{bmatrix} \quad (6.2)$$

$$= L \begin{bmatrix} v \\ w \\ u \end{bmatrix} + \mathbf{F},$$

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

$$\begin{pmatrix} v \\ u \end{pmatrix}_t = \begin{bmatrix} 0 & -\frac{1}{I_x} \frac{\partial}{\partial x} \frac{\partial^b}{\partial x} & I_y + I_x \frac{\partial}{\partial y} \frac{\partial^b}{\partial y} \\ I_x & I_y & 0 \end{bmatrix} \begin{pmatrix} v \\ u \end{pmatrix} + \begin{pmatrix} F \\ 0 \end{pmatrix} \tag{6.3}$$

$$= L + F$$

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

$$\begin{aligned} \psi_t &= L\psi + F \\ \psi(0) &= \psi_0 \end{aligned} \tag{6.4}$$

where $\psi = [v\ w\ u]^T$ (for (6.2)), $\psi = [v\ u]^T$ (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

$$\psi(t) = P(t)\psi_0 + P(t)\int_0^t P(\tau)^{-1} F(\tau)\, d\tau$$

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 nite di erence (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 coe cients where we compare the result with the exact solution. In addition, we provide examples with variable coe cients. For variable coe cients 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 coe cients.

good compromise between an e cient 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 $(x; y) = (_k; _l)$ 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 rst 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 rst order system in time using (6.3), and discretize it in space using the fourth order nite di erence 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

$$\begin{cases} \\ \\ \end{cases} u$$

Comparison of accuracy and speed

For the experiments in this section, we solve

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

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

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

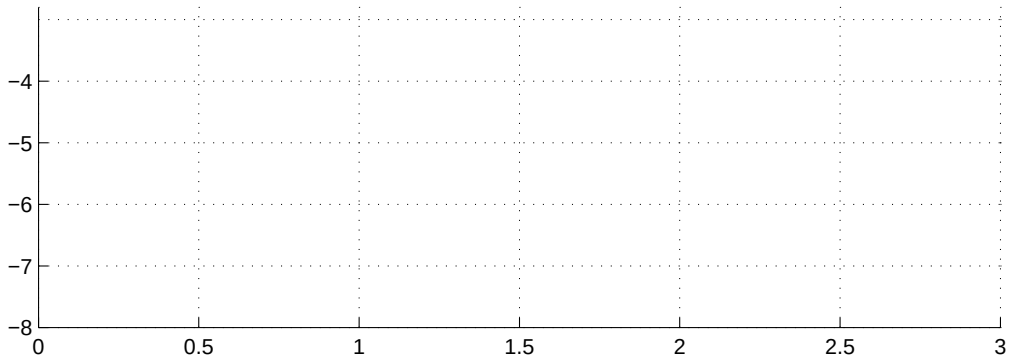
We note that this solution contains both low frequency (the first term) and high frequency (the second term) modes. For the experiments in this section, we measure the error of the vector $\mathbf{f} = [v \ w \ u]^T$ when using a bandlimited scheme, and the error of the vector $\mathbf{f} = [v \ u]^T$ 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 $\mathbf{f}$ approximates the exact solution $\mathbf{g}$, then

$$\text{error} = \frac{\|\mathbf{f} - \mathbf{g}\|}{\|\mathbf{g}\|}.$$

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 $1 \cdot 10^4$ 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 $c = 23$ which corresponds to an oversampling factor of approximately 1.4 for periodic functions. We set the accuracy in the construction to $\varepsilon = 10^{-7}$ resulting in 64 nodes, and select the time step $\Delta t = \frac{\pi\sqrt{2}}{23}$ 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


−4
−5
−6
−7
−8
0
0.5
1
1.5
2
2.5
3

functions) and a timestep $t = \frac{t}{128}$. 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 signi cantly 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 $\frac{t}{32}$. 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

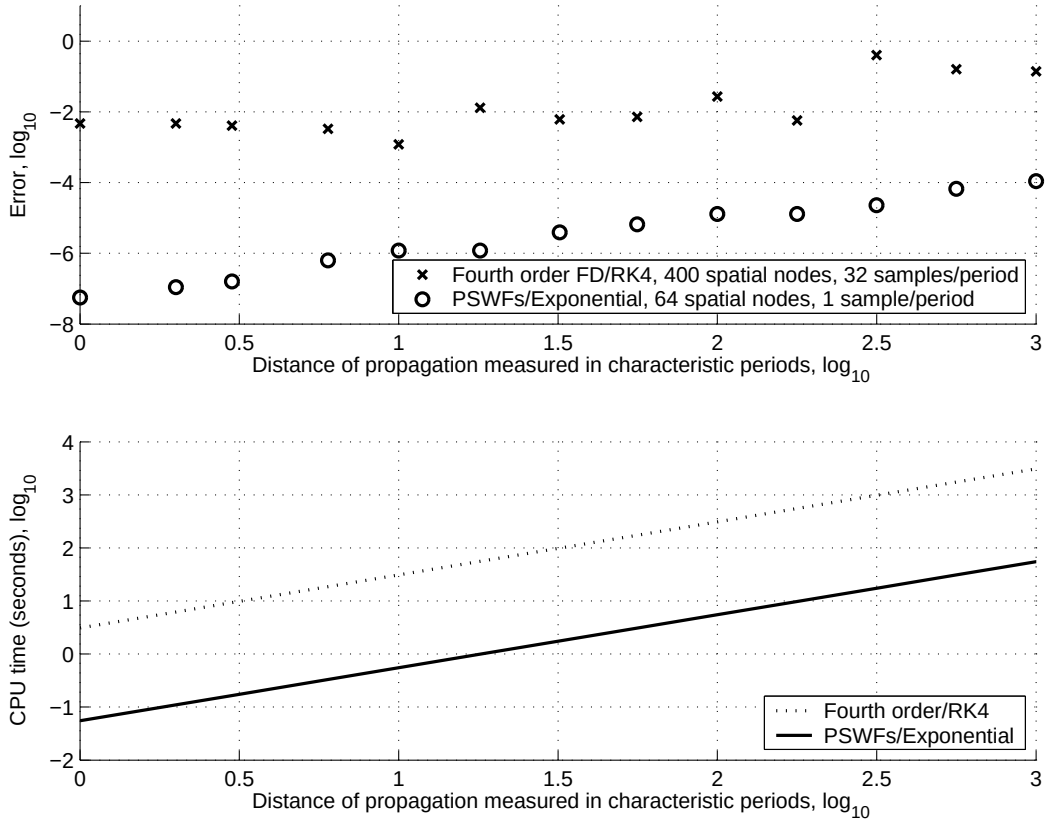


Figure 6.2: Relative error ($\log_{10}$) in the max-norm for approximating the solution to (6.5) (top), and the computational time (bottom).

methods are comparable, but the bandlimited semigroup method is signi cantly more accurate. The error pro le for the fourth order scheme oscillates signi cantly over time, while the error pro le 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 $x_b = 19{:}5$, and solve the model problem

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

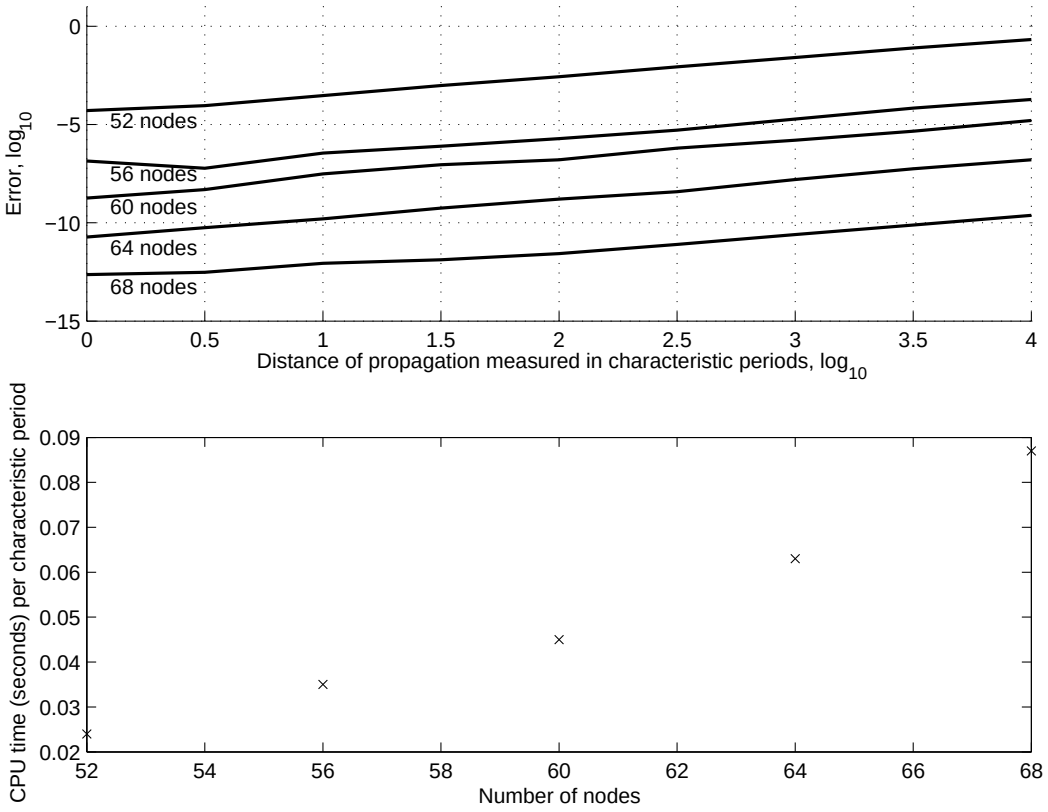


Figure 6.3: Relative error ($\log_{10}$) in the max-norm for approximating the solution to (6.5) for $b = 20$ using two different sampling rates (top). The CPU time for propagating the wave one characteristic period for a varying num(6.5)

Tj /R36 11.9552 Tf 9BT /R562 7.93l8 389.31eomd (time-3Td rio-4M true /Wtak 6242.8196 0 Td ()Tj /R1 10 83 -14.4

of which correspond to right-traveling waves. Solutions of this equation take the form

$$u(x;t) = e^{i!(x\ ct)};$$

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

$$\frac{@}{@x}u = i!e^{i!(x\ ct)}:$$

If the error in the representation of the di erentiation operator is of the form

$$\frac{@}{@x}u\ '\ if(!)e^{i!(x\ ct)};$$

then the Fourier mode propagates with the velocity $cf(!)=!$. Unless $f(!) = !$, which corresponds to the exact di erentiation, the Fourier modes of di erent frequencies travel with di erent velocities. For example, in the case of the second order centered nite di erence approximation of the derivative, $f(!) = \sin(!)$.

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

$$\begin{cases} u_{tt} = u_{xx} + u_{yy};\ (x;y)\,2\,(\ 2;2)\quad(\ 2;2) \\ u(x;y;0) = \mathrm{sinc}^2(27\ x)\,\mathrm{sinc}^2(27\ y) \\ u_t(x;y;0) = 0 \\ u(\ 2;y) = u(x;\ 2) = 0 \end{cases} : \qquad (6.6)$$

The solution is a sharp

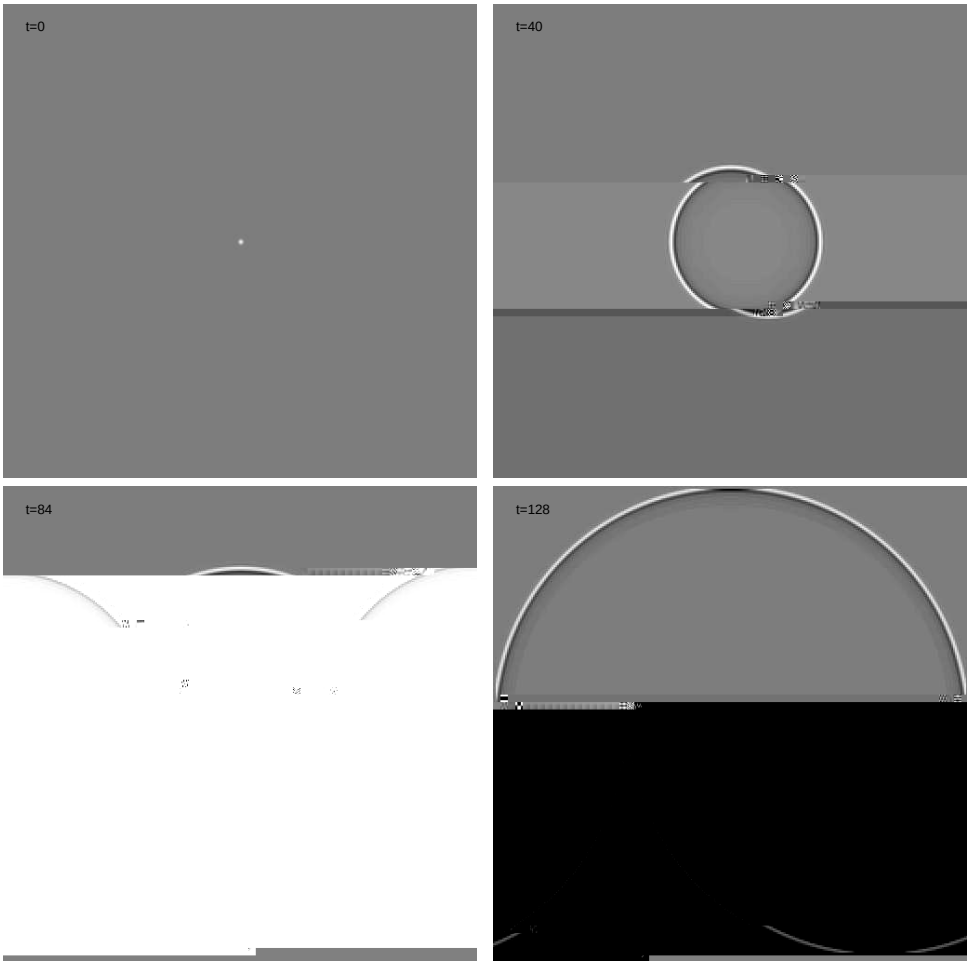


Figure 6.4: Solution of (6.6) using the bandlimited semigroup method. The shape of the pulse is maintained throughout the propagation.

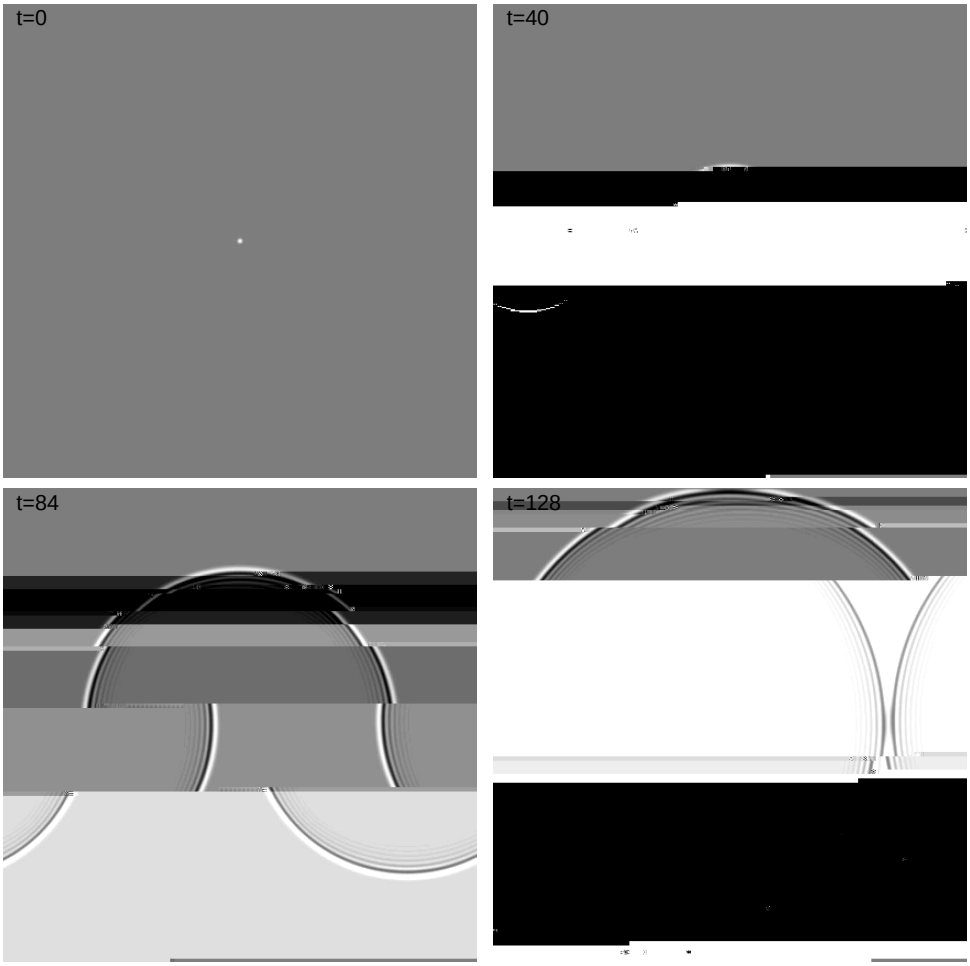


Figure 6.5: Solution of (6.6) using a fourth order scheme. Note the ripples near the wave front which are caused by numerical dispersion.

6.3.4 Numerical results for variable coefficients

In this section we solve the acoustic equation for variable coe cients. 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

$$\begin{cases} u_{tt} = \frac{1}{(y)}(u_{xx} + u_{yy})\,; \quad (x;y)\,2\,(\quad 1;1) \quad (\quad 1;1) \\ u(x;y;0) = e^{\quad 1000(x \quad + y \quad)} \\ u_t(x;y;0) = 0 \\ u(\quad 1;y) = u(x; \quad 1) = 0 \end{cases} \tag{6.7}$$

where

$$(y) = \frac{1}{1 \quad \frac{\sin(\quad (y+1))}{2}}:$$

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

For the bandlimited semigroup

arLsetmiquadratuje-44347 0451265 0thTwddods

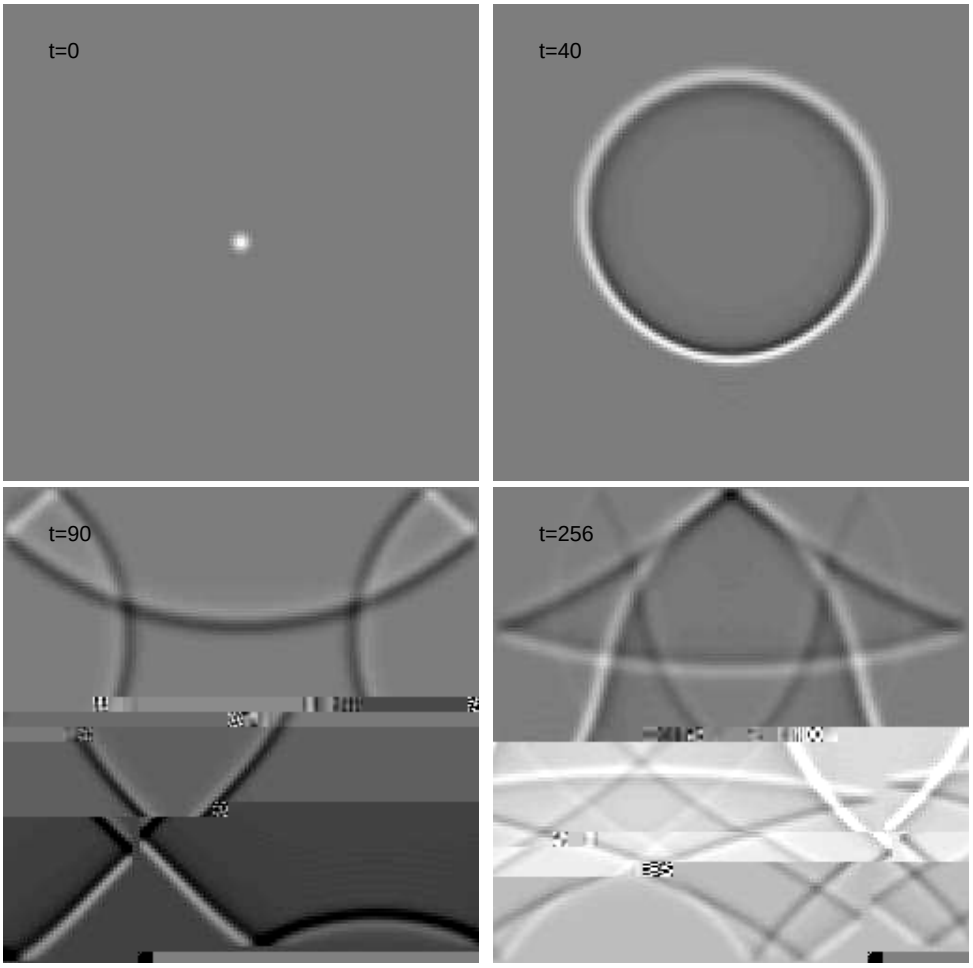


Figure 6.6: Solution of (6.6) using the bandlimited semigroup method.

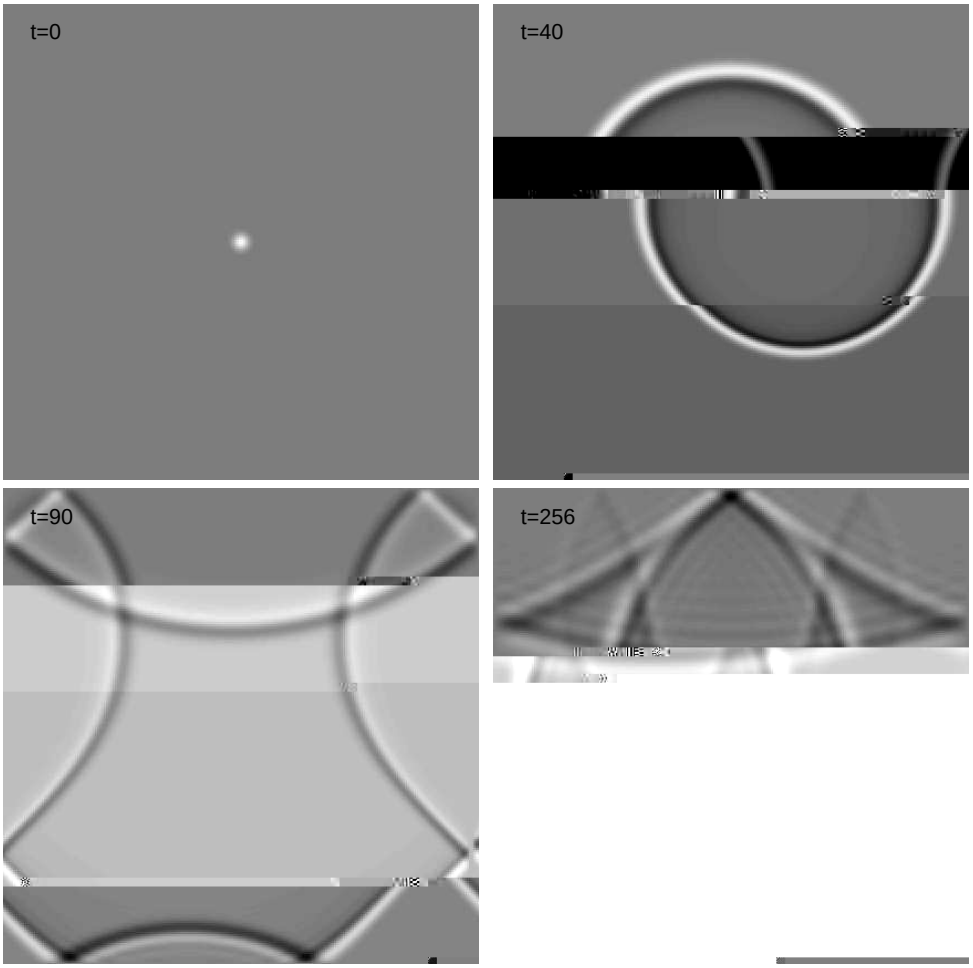


Figure 6.7: Solution of (6.6) using a fourth order scheme. Note the ripples which are caused by numerical dispersion.

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

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

where $w(x)$ satisfies the Sommerfeld radiation condition and $\kappa(x) = 1$ for x outside some bounded domain. We refer to $e^{i\omega x}$ 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 [illegible]. In this thesis, we [illegible]

solution (y) to the equation

$$\begin{aligned} &_{yy} = A \; ; \; y \quad 0 \\ &(0) = _{0} \\ &_{y}(0) = _{0} \end{aligned} \tag{7.3}$$

is stable, that is, there exists a constant K such that k (y)k K for all y 0.

Proof We rst write (7.3) as the rst order system

$$\begin{bmatrix} \\ _{y} \end{bmatrix} = \begin{bmatrix} 0 & A \\ I & 0 \end{bmatrix} \begin{bmatrix} \\ _{y} \end{bmatrix} \quad L \tag{7.4}$$

where I is the N-by-N identity matrix. Since A is diagonalizable, there exists an invertible matrix S such that $L = S^{1} L_D S$ where L_D is a diagonal matrix with the eigenvalues of L along the diagonal. De ne = S . Since L is y-independent, S commutes with $\frac{d}{dy}$ and we

the boundary condition $u(-1) = 0$, then $\lambda = -\omega^2 + \frac{k}{4}$ for $k = 0, 1, \ldots$. If λ is negative, then we must have that $\frac{k}{2} < |\omega|$. Now consider the equation

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

with the boundary conditions $u(-1) = 0$. 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 $k = 1, 2, \ldots$. Hence, it is clear that these solutions are bounded for all $y \geq 0$ if $\frac{k}{2} < |\omega|$. 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

$$P(\sigma) \sum_{\lambda_k < \sigma} P_k$$

such that

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

without computing the individual operators P_k. For self-adjoint matrices, $P_k = e_k e_k^T$ where e_k is the eigenvector corresponding to the eigenvalue λ_k. For general diagonalizable matrices, $P_k = e_k f_k^T$ where e_k and 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

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 $|\lambda_i| < 1$ for all eigenvalues λ_i of A. From the assumption on it follows that

$$\|\mathrm{diag}(\lambda_i)\|_2 = \|SAS^{-1}\|_2 \le \|S\|_2\|A\|_2\|S^{-1}\|_2 = \mathrm{cond}(S)\|A\|_2 < 1.$$

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 $O(\log_2(\mathrm{cond}(A)) + O(\log_2(\log_2(1/\varepsilon)))$.

Using (7.7) we can now compute spectral projectors by a sequence of matrix-matrix m

By introducing the function v, we write (7.2) as

Algorithm: Numerical scheme for wave propagation on spacetime 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 $A = L_0 - \omega^2(1 + f)I$ where I denotes the identity operator.
4. Construct the spectral projector $P = (I - \text{sign}(A))/2$ according to Theorem 27.
5. Construct the spatial operator

$$L = \begin{bmatrix} 0 & PAP \\ I & 0 \end{bmatrix}.$$

6. Select the step size γ (see discussion in Section 6.3.1 on how to choose the step size), and compute the matrix exponential e^{γ}

where $b = k\ =2$ for some integer $k \quad 1$ and $! = \sqrt{b^2 + (23\ )^2}$. T

$$u(x;y) = \sin \frac{(x+1)}{2} \cos \quad !^2 \quad 1)$$

$$+ \sin(b(x+1)) \cos(\sqrt{!^2 \quad b^2} \quad :$$

We note that this solution contains both low frequency (the quenc (the second term) modes. We measure the error using the rela at is, if u approximates the exact solution u, then

$$\text{error} = \text{kd es.}$$

N.a eric res ts for ! ri e coe cients

Let us solve the Helmholtz equation as an initial value problem for the case with variable background in the x-direction. Consider the equation

$$\begin{cases} u_{xx} + u_{yy} + (x)(27)^2 u = 0; \ (x;y) \ 2 \ (\ 1;1) \quad (\ 1;7) \\ u(x;0) = g(x) \\ u_y(x;0) = 0 \\ u(\ 1;y) = 0 \end{cases} \tag{7.11}$$

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 $c = 54$ and the accuracy $= 10^{-7}$. 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 $y = \frac{1}{128}$, corresponding to approximately 10 samples per wavelength for easier visualization of the resulting wave eld.

For the rst experiment, we choose a constant background ($_1(x) = 1$) and the initial pulse e^{1000x}. We display the resulting wave eld 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 di racts and then re ect at the boundaries.

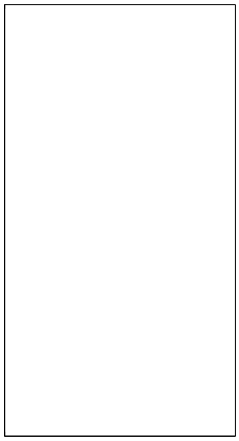
For the following two experiments, we choose the background coe cients

$$_2(x) = \frac{1}{1 + \frac{\sin \ (x+1)}{2}}$$

and

$$_3(x) = \frac{1}{1 + \frac{\sin \ (x+1)}{2} \ (1 \quad 0{:}9e^{\ 100x})};$$

respectively. The pro le of the compressibility and the velocity $\frac{1}{(x)}$ are given in Figure 7.2 We use the initial condition $g(x) = e^{\ 1000(x+0:5)}$ 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 $x = 0$.



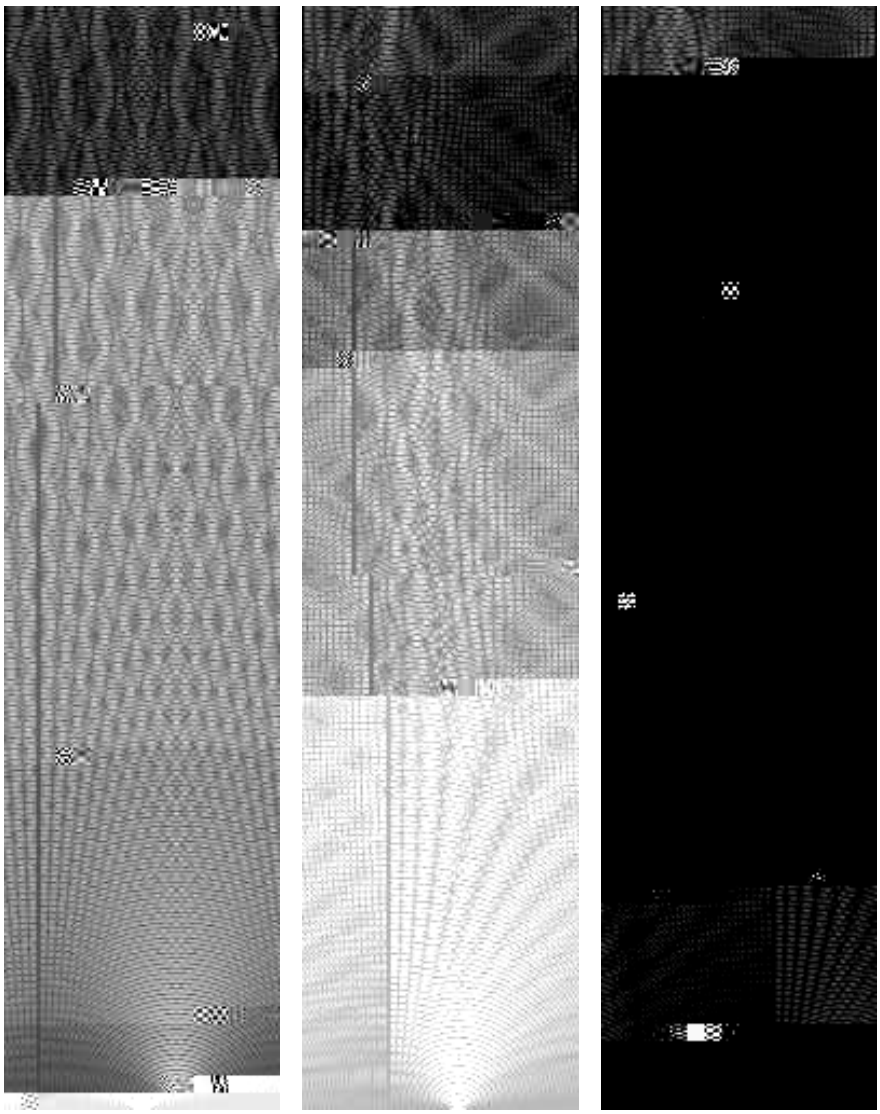


Figure 7.3: Absolute value of the wave field from the solution to (7.11) for the three compressibilities $_1(x)$ (top), $_2(x)$ (center), and $_3(x)$ (bottom).

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 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

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 di erent, 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 $t_1; :$ duced ecimen 4.4 Tl774

where w_l are scalar w

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 xing the image size and varying the number of projections.

References

[1] B. Alpert, G. Beylkin, D. Gines, and L. Vozovoi. Adaptive solution of partial di erential 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 re ection 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 di erential 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 Beylkin.

[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 elds. 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 rst course in the numerical analysis of di erential 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

Appendix A

Proof of Corollary 10

We observe that

$$\int_{-1}^{1} e^{ictx}\, dt$$

By introducing $w_k = v_k/c$ and $\tau_k = t_k/c$ we have that

$$\int_{-1}^{1} e^{ictx}\, dt - \sum_{k=1}^{N} w_k e^{ic\tau_k x} < \varepsilon/c.$$

We note that $|\tau_k| \leq c = \frac{1}{N}$ so by choosing N sufficiently large, $|\tau_k| < 1$.

According to Theorem 9, the error for approximating $v(x)$ is bounded by

$$\int_{-\infty}^{\infty} \phi(t) e^{-ity}\, dt - \sum_{k=1}^{N} v_k e^{-i\tau_k y} <$$

$$2k\,k_1\,3\left(\frac{c}{N}\right)^{2m} + \frac{2}{2 + (2+\sqrt{3})^N + (2-\sqrt{3})^N} + \frac{2d_m}{1-e^{-\gamma_m}} e^{-\gamma_m(1-\delta)N}\Bigg)!\,.$$

We conclude the proof by observing that $k\,k_1 = 2c$.

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 $\sum_i \lambda_i e_i f_i$ 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

$$L = \sum_{k=1}^{K} s_k e_k f_k$$

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 $\mathbb{C}^N$. 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

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 su cient to reduce the rank

A gorith R n red ction of sep r ted represent tion

1. **Normalize $\{e_k g_k\}$ and $\{f_k g_k\}$ and adjust $\{s_k g_k\}$ accordingly. Set $i = 1$.**
2. **Pivot (put the term with the largest s_i rst)**
3. hi e **$i < r$**

$f_i = s_i f_i$
For **$j = i + 1 : r$**
$a_j = < e_i, e_j >$
$e_j = e_j - a_j e_i$ (This means that $< e_j, e_i > = 0$.)
$b_j = \|e_j\|_2$
$f_i = f_i + s_j a_j f_j$
end
For **$j = i + 1 : r$**
$s = s_j b_j$
f **$s >$**
$e_j = \frac{e_j}{b_j}$
$s_j = s$
e se
$r = r - 1$
end
$s_i = \|f_i\|$
f **$s_i >$**
$f_i = \frac{f_i}{s_i}$
e se
$r = r - 1$
end
end

Pivot (put the term with the largest s_i rst)

i=i+1

end

B.2 PLR matrix-vector multiplication

A gorith M tri ector tip ic tion for tri gi en y PLR represen t tion

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 o -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,

$$D_3^2 = \begin{array}{c|c} D_5 & U_3^3 \\ \hline L_3^3 & \end{array}$$

14 -0.360 0 1 -7 43999 Td (l)Tj /R36 11.9552 Tf 9.72 3.12 Td (as3u)Tj /R36 11.9isTd (e)T75/R36 11.9aTd (F)8j /R39 11.9

L^3

We use the corresponding notation for the matrices B and C, but mark blocks from these matrices with a tilde (~), and a 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 $i = k \quad 1 : 1$

For $j = 1 : 2^{i\ 1}$

$F_j^i = U_j^i$ d T 002 T+97011 Tf 9.1302 4.3d (i)Tj -R39)Tj /R19 7.97011 Tf 6.599dC52 Tf -1.32 -3

multiplication in Appendix B.2 on the left factors in the separated representation of U_l^k. We can compute the product $U_l^k D_{2k}^k$ in a similar way. Finally, we can compute the product $U_j^i L_l^k$ 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

4. For $i = m \quad 1 : 1$ (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

$$\left(\frac{\partial}{\partial x} \otimes I_y \right) u(x;y) = \sum_{l=1}^{M} \sum_{n=1}^{N} s_{nl} \left(\sum_{k=1}^{M} \sum_{m=1}^{N} s_{mk} \, \phi'_{mk}(x) \right) \phi_{nl}(y)$$
$$g(x) \sum_{l=1}^{M} \sum_{n=1}^{N} s_{nl} \, \phi_{nl}(y).$$

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, $\frac{\partial}{\partial x} \otimes I_y \, u(x;y)$ is continuous across the vertical interfaces for each y.

C.2 Construction of derivativ

A gorith Constr ction of **D** nd $\mathbf{D_0}$ for sing to so e the co stic eq tion

1. **Construct N quadrature nodes and weights according to De nition 11.**
2. **Construct the matrix S according to (2.27), the matrix K according to (4.1), and the matrices E, F, and G de ned in De nition 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 De nition 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 De nition 20, and for multiple intervals by using (3.32) and Tables 3.1-3.3.**
5. **Construct**

$$\mathbf{L} = \begin{matrix} 0 & \mathbf{D_0} \\ \mathbf{D} & 0 \end{matrix} :$$

6. **Project**

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 $A = S\,\mathrm{diag}(\lambda_k)S^{-1}$ where the columns of S are right eigenvectors e_k of A and the columns of $(S^{-1})^T$ are the left eigenvectors f_k. By scaling the eigenvectors such that $f_k^T e_k = 1$, we have that

$$f_k^T e_l = \delta_{kl}: \tag{D.1}$$

(1) This follows from the definition of P_k and the scaling $f_k^T e_k = 1$.

(2) By using (D.1) we have that

$$P_k P_l = e_k f_k^T e_l f_l^T = e_k(\delta_{kl}) f_l^T = \delta_{kl} P_k:$$

(3) Since A is diagonalizable, the eigenvectors form a complete basis and, hence, if ξ is an arbitrary vector then

$$\xi = \sum_k x_k e_k$$

for some set of coefficients x_k. Using (D.1) it follows that

$$\sum_l P_l \xi = \sum_l P_l \sum_k x_k e_k = \sum_l \sum_k x_k e_l f_l^T e_k$$

$$= \sum_l x_l P_l e_l = \sum_l x_l e_l = \xi$$

and, since ξ is arbitrary, $I = \sum_k P_k$.

(4) Let be an arbitrary vector. Then

$$A = A \sum_k x_k e_k = \sum_k x_k A e_k = \sum_k {}_k x_k e_k = \sum_k {}_k P_k$$

and, since is arbitrary, $A = \sum_k {}_k P_k$.

(5) Using (D.1) and that the eigenvectors form a complete basis, we have that

$$P_k = P_k \sum_l x_l e_l = x_k e_k :$$

D.2 The inverse problem for acoustics in the time domain

The equation de ning 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. Consider

Appendix E

A fast reconstruction algorithm for electron microscopy [6]

e o y Bey n
Department of Applied Mathematics, University of Colorado at Boulder
. d N M.3r on. de
Boulder Laboratory for 3-D Electron Microscopy of Cells,
Department of Molecular, Cellular, and Developmental Biology, University of Colorado at Boulder
3r . n . nd e
Department of Applied Mathematics, University of Colorado at Boulder

A str ct **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 ltered backprojection. For two-dimensional images, the new algorithm scales data**

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 $O(N\ M \log M) + O(M\ N \log N)$ operations, where N is the number of projection angles and $M\ \ N$ is the size of the reconstructed image. This should be compared to computing the standard $\ltered$ backprojection using direct summation in the space domain which scales as $O(N\ M\ N)$. 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 $\ltered$ backprojection for tomographic reconstruction sums $\ltered$ 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.

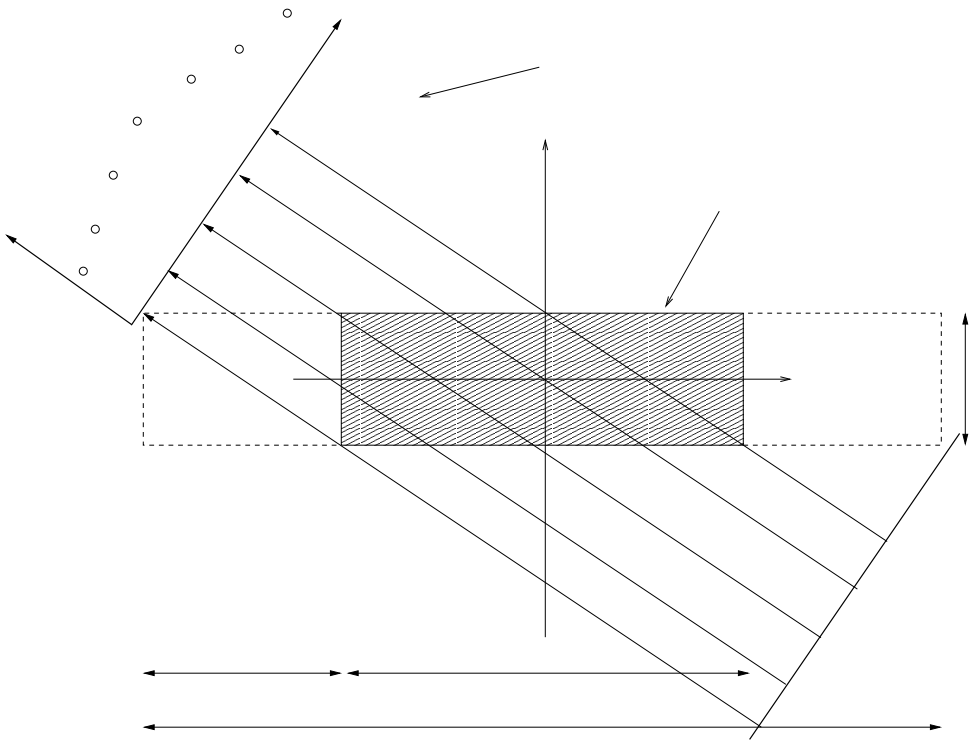
Reconstruction algorithms for electron microscopy imaging of biological specimens have been described by DeRosier and Klug [19] via a Fourier based method, and by Gilbert [28] via direct summation. For a review, see Frank [22].

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 $\tech$nique $\rst$ 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 $\exibility$ for choosing interpolation schemes and incorporating $\lters$ 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 $\e$ectively sums over the angles in the Fourier domain. We discretize the inversion formula in

Section E.4



by $I(t) = I_0 e^{-R_\theta(t)}$, 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 $I(t)$. On taking $R(t) = -\ln I(t)$, our measurements provide us with $R_{\theta_l}(t_k)$, where θ_l and t_k are discretizations of θ and t respectively. Sampling $R(t)$, 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 $l = 1;2;\ldots;N$. 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 $R(t)$ in (E.2) via the integral

$$g(x;z) = \int_0^{\pi} (\mathcal{H} R_\theta)(t(x;z))\, d\theta; \tag{E.4}$$

where $\mathcal{H}$ 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

$$\hat{\mathcal{H}}(\omega) = |\omega|. \tag{E.5}$$

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

$$g(x_m; z_n) = \sum_{l=1}^{N_\theta} w_l \, [\mathcal{R}_l](t(x_m; z_n)) \qquad \text{(E.6)}$$

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 $\mathcal{R}_l(t(x_m; z_n))$ are estimated by some interp

Consider the sum used for ltered backprojection (E.6),

$$g$$

where x_s is a shift parameter which depends on the selection of the coordinate system in x. It is easily veri ed that the function $R_l(t)$ is continuous with respect to t. Using (E.13) we have

$$\begin{aligned}\hat{R}_l(\omega) &= \sum_{k=0}^{M-1} r_{kl} \int_{-\infty}^{\infty} (t - k + x_s) e^{-2\pi i t\omega}\, dt \\ &= \sum_{k=0}^{M-1} r_{kl} e^{-2\pi i(k - x_s)\omega} \int_{-\infty}^{\infty} \phi(s) e^{-2\pi i s\omega}\, ds \\ &= e^{-2\pi i x_s \omega} \hat{\phi}(\omega) \sum_{k=0}^{M-1} r_{kl} e^{-2\pi i k\omega}.\end{aligned} \tag{E.14}$$

Combining (E.11) and (E.14) yields

$$\begin{aligned}v_l(\omega) &= \frac{w_l}{\cos\theta_l} e^{-2\pi i x_s \frac{\omega}{\cos\theta_l}} \hat{\phi}\left(\frac{\omega}{\cos\theta_l}\right) \hat{\phi}\left(\frac{\omega}{\cos\theta_l}\right) \sum_{k=0}^{M-1} r_{kl} e^{-2\pi i k \frac{\omega}{\cos\theta_l}} \\ &= F_l(\omega) \hat{\phi}_l\left(\frac{\omega}{\cos\theta_l}\right);\end{aligned} \tag{E.15}$$

where

$$F_l(\omega) = \frac{w_l}{\cos\theta_l} e^{-2\pi i x_s \frac{\omega}{\cos\theta_l}} \hat{\phi}\left(\frac{\omega}{\cos\theta_l}\right) \hat{\phi}\left(\frac{\omega}{\cos\theta_l}\right); \tag{E.16}$$

and

$$\hat{\phi}_l\left(\frac{\omega}{\cos\theta_l}\right) = \sum_{k=0}^{M-1} r_{kl} e^{-2\pi i k \frac{\omega}{\cos\theta_l}}. \tag{E.17}$$

Note that the factor $F_l(\omega)$ is independent of the data once the angles θ_l are known.

The nal step is computing $g_n(x)$ from $\hat{g}_n(\omega)$. By taking the inverse Fourier transform of (E.8) we arrive at

$$g_n(x) = \int_{-\infty}^{\infty} \sum_{l=1}^{N_\theta} v_l(\omega) e^{-2\pi i \tau_l(\omega) z_n} e^{-2\pi i \omega x}\, d\omega; \tag{E.18}$$

where $\tau_l(\omega) = \omega \tan\theta_l$ and $v_l(\omega)$ are de ned in (E.15). We also observe

$$\begin{aligned}g_n(x) &= \int_{-\infty}^{\infty} \sum_{l=1}^{N_\theta} v_l(\omega) e^{-2\pi i \tau_l(\omega) z_n} e^{-2\pi i \omega x}\, d\omega \\ &= \sum_{j \in \mathbb{Z}} \int_{-\frac{1}{2}}^{\frac{1}{2}} \sum_{l=1}^{N_\theta} v_l(\omega + j) e^{-2\pi i \tau_l(\omega + j) z_n} e^{-2\pi i(\omega + j)x}\, d\omega;\end{aligned} \tag{E.19}$$

where we recall

$$v_l(\omega + j) = \frac{w_l e^{2\pi i x_s \frac{\omega}{\cos\theta_l}} \hat{\psi}\left(\frac{\omega + j}{\cos\theta_l}\right) \hat{\psi}\left(\frac{\omega + j}{\cos\theta_l}\right) \hat{\psi}_l\left(\frac{\omega + j}{\cos\theta_l}\right)}{\cos\theta_l}.$$

Let us consider a bandlimited filter such that the support of $\hat{\psi}$ is contained in $[-\frac{1}{2}, \frac{1}{2}]$. As an important example consider

$$\hat{\psi}(\omega) = \begin{cases} 1 - |\omega|, & |\omega| \leq \frac{1}{2} \\ 0, & |\omega| > \frac{1}{2} \end{cases}.$$

For this example we observe that $v_l(\omega + j) = 0$ for all $j \neq 0$. Hence (E.19) reduces to

$$g_n(x) = \int_{-\frac{1}{2}}^{\frac{1}{2}} \sum_{l=1}^{N_\theta} v_l(\omega) e^{2\pi i \tau_l(\omega) z_n} e^{2\pi i \omega x} d\omega. \qquad \text{(E.20)}$$

Remark Note that equation (E.20) is equivalent to the sum (E.6) used in the direct summation algorithm, where 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

$$x_m = x_s + m, \quad m$$

we approximate (E.20) by

$$g_n(x_m) \approx \frac{1}{M_f} \sum_{k=-\frac{M_f}{2}+1}^{\frac{M_f}{2}} \left(\sum_{l=1}^{N_\theta} v_l\left(\frac{k}{M_f}\right) e^{-2\pi i l \left(\frac{k}{M_f}\right) z_n} \right) e^{2\pi i \frac{k}{M_f} x_m}$$

$$= \frac{1}{M_f} \sum_{k=-M}^{\frac{M_f}{2}}$$

Since our algorithm uses the sum (E.22), we need a fast algorithm to compute the sums

$$\hat{u}_n = \sum_{k=1}^{M} u_k e^{2\pi i k n}, \quad n =$$

where $_{max} = \max\{|_l|\}_{l=1}^{N_\theta}$. 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 $M_{ext} = M + 2$ 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

$$M_f \quad M + N \tan _{max}: \tag{E.26}$$

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

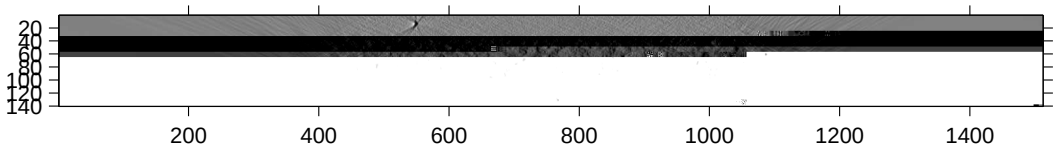


Figure E.2: For this reconstruction M = 572, N = 140, and $_{max}$ = 73:31°. This gives3114071 l S 4973.17

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

$$\hat{\psi}(\omega_k) = \begin{cases} |\omega_k|, & \omega_k \le 1/2 \\ 0, & \omega_k > 1/2 \end{cases}$$

$$\hat{\varphi}(\omega_k) = \left(\frac{\sin(\omega_k)}{\omega_k} \right)^2 .$$

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 angles, we wish to reconstruct a sequence of N_i images at M 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: $O(N\ M_f) + O(N\ M \log M)$.

   (b) For each frequency ω_k, compute the sum in (E.22) using the USFFT. See the Appendix for details. Computational cost: $O(M_f N) + O(M_f N \log N)$.

(c) Compute the sum in (E.21) using the FFT. Computational cost: $O(N M_f \log M_f)$

The steps are illustrated in Figure E.4.

A Fast Fourier Summation algorithm is important when reconstructing a large number of two-dimensional slices. In most applications, the angles $_l$ and lters 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 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 $M_f \quad 2M$. Therefore, the total computational cost of the full three-dimensional reconstruction algorithm is given by $O(N_i N\ M \log M) + O(N_i N\ M_f \log M_f)$. This should be compared to the cost $O(N_i M N N\ )$ for the traditional method of direct summation. Actual speed comparisons are given in Section E.5.2.

Re r The sum $\sum_{k=0}^{M\ 1} r_{kl} e^{2\ ik\frac{\omega}{\cos\theta_l}}$ computed in Step 2 of the algorithm is similar to the sums that can be computed with the FFT. For each xed angle $_l$, the

Thus,

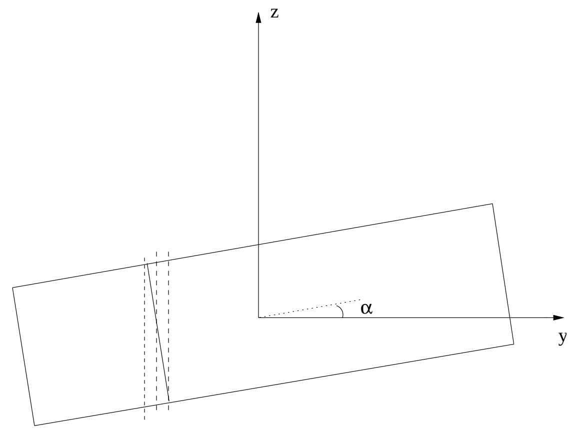


Figure E.5: Side view of section to be reconstructed where the section is tilted by the angle around the x-axis. Dashed lines are slices computed by the algorithm and the solid line is the output slice interpolated from vertical slices.

E.5.2 Tests

Acc r cy

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 lm 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 lm 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 lm as ducial markers for alignment. The di[illegible] between the

Figure E.6: (a): A test data set computed with the Fast Fourier Summation algorithm. (b): The di erence 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 ampli ed 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

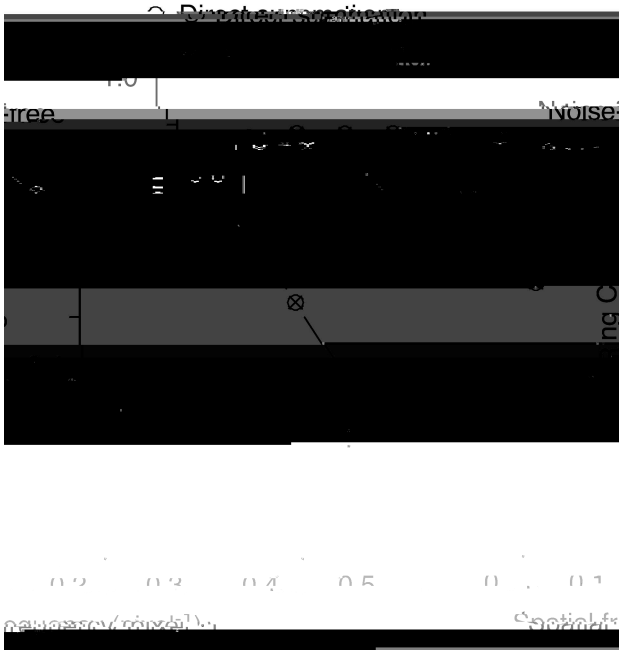


Figure E.7: The graph shows the correlation between the Fourier transforms of the reconstruction and those of the test volume, as a function of spatial frequency, averaged over 190 slices of each reconstruction.

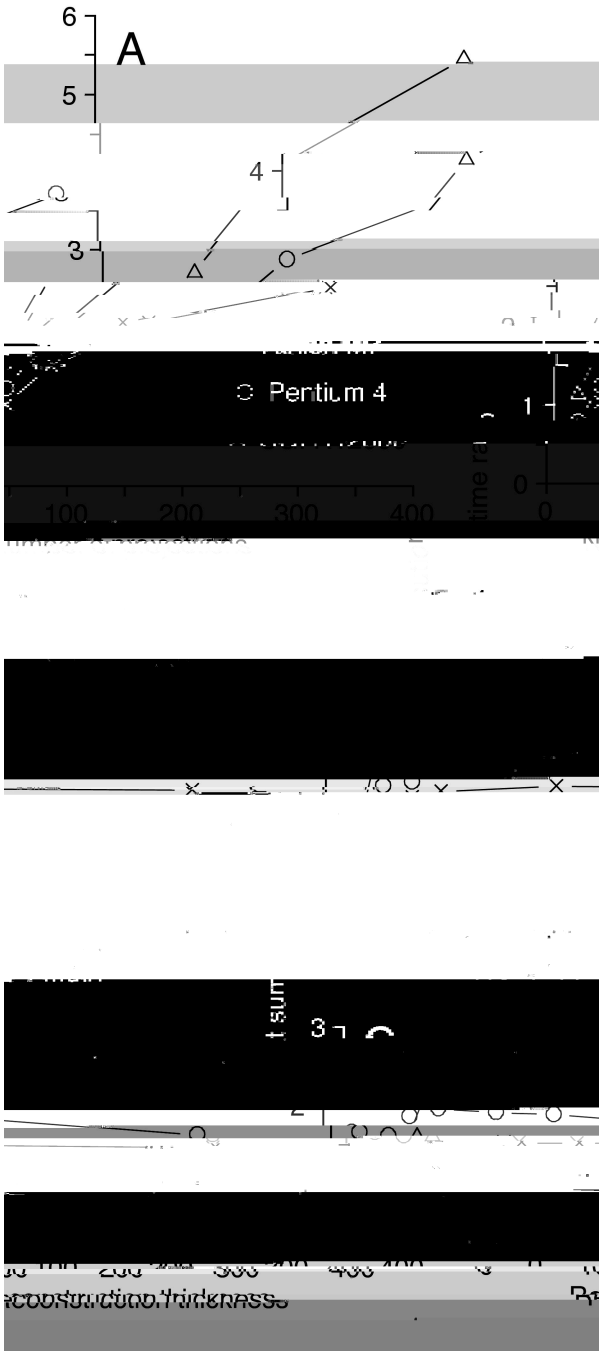


Figure E.8: (Direct summation)/(Fast Fourier Summation) execution time ratio for three computer architectures. (a): Dependence on number of projections with width 1024 and thickness 200. (b): Dependence on reconstruction width with 80 projections and thickness 200. (c): Dependence on thickness with 80 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 bene t climbs 5-fold with an increase from 20 to 320 projections. For the other dimensions, the bene t from Fast Fourier Summation tends to rise with initial increases then atten out. This initial rise was most abrupt and pronounced with Athlon processors (e.g., Fig. E.8c) and it re ects 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 bene t from Fast Fourier Summation is about 1.5 - 2.5 fold, with greater bene ts 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 projections falls below a speci ed limit for the given computer architecture.