# Important Notice

THE UNIVERSITY OF CALGARY

# Numerical Methods in Seismic Wave Propagation

by

MATT MCDONALD

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF M.Sc. IN APPLIED MATHEMATICS

DEPARTMENT OF MATHEMATICS AND STATISTICS

CALGARY, ALBERTA

September, 2012

# THE UNIVERSITY OF CALGARY

# FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "NUMERICAL METHODS IN SEISMIC WAVE PROPAGATION" submitted by MATT MCDONALD in partial fulfillment of the requirements for the degree of M.Sc. IN APPLIED MATHEMATICS.

Dr. Michael P. Lamoureux
Department of Mathematics and Statistics

Dr. Wenyuan Liao
Department of Mathematics and Statistics

Dr. Gary F. Margrave
Department of Geoscience

Dr. Mike Potter
Department of Electrical and Computer Engineering

Date

# Abstract

The numerical modelling of wave equations is a common theme in many seismic applications, and is an important tool in understanding how the physical systems of interest react in the process of a seismic experiment. In this thesis we apply state-of-the-art numerical methods based on domain-decomposition combined with local pseudospectral spatial discretization, to three physically realistic models of seismic waves, namely their propagation in acoustic, elastic, and viscoelastic media. The Galerkin formulation solves the weak form of the partial differential equation representing wave propagation and naturally includes boundary integral terms to represent free surface, rigid, and absorbing boundary effects. Stability, accuracy, and computation issues are discussed in this context along with direct comparison with finite difference methodologies.

This works is an effort to bridge the gap between the development of accurate physical models to represent the real world, as seen in seismic modelling, and the implementation of modern numerical techniques for the accurate solutions of partial differential equations.

# Acknowledgements

I would like to thank Michael Lamoureux for being not only an amazing supervisor, but also a great friend, my friends and colleagues in CREWES, our director Gary Margrave, the always helpful and supportive Laura Baird and everyone who I met and talked with along the way. Graduate school for me was a life-changing experience and the people I met while a graduate student will always be close to my heart. Thank you everyone.

I would also like to gratefully acknowledge the support of *mprime* through the POTSI research project and its industrial collaborators, the support of NSERC through the CREWES consortium and its industrial sponsors, and support of the Pacific Institute for the Mathematical Sciences.

Dedicated to my family: My Parents Brian and Janet, my brother John and everyone on my mother and father's sides of the family. Kirsten, Jacqueline, Cheryl, Denise, Bart, Ashley, Richard and the entire Dutchak family. Diane Pshyk, Steve, Erin and Taylor Kaye. Darius Bilokraly. Simon, Ava and Joshua Rainsbury. Kevin, Darlene, Sarah and Katie Scott.

My friends: Shane and Lukas Ash and Mika Seo. Rob and Bradley Pocsik and Amanda Chalmers. Kurt and Natalie Reid. Mark and Caleb Slade. Jonathan Murrin and Paula Kahr. Cody Painter and Shelley Blimke. Chelsie Cann. Jason and Kristin Demoskoff. Nicole Dekuysscher. All of my friends I met through music and school and adventure and all of the things that come in between in every iteration of life.

It's been a bumpy ride at times. You have all been there for me when I needed you most, and I couldn't have done it without you.

This is for you.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Background

The numerical modelling of seismic waves is an integral part of many seismic processing procedures. When attempting to image the subsurface of the earth it is sometimes necessary to iteratively update the current model based on the difference between the response of the modelled system, and the data recorded from the actual experiment. As such, it is important that both the numerical method, and the type of model used in the forward modelling are capable of accurately representing the physical experiment.

Here, an argument is made that the partial differential equations that accurately model the earth's properties require a specific type of numerical method. More precisely, because of the discontinuous nature of the earth's properties, the partial differential equations exhibit a low-order level of continuity that shows as a "kink" at the discontinuous interfaces. Approximation methods that assume a higher level of continuity, can cause the position of these kinks to show up at incorrect spatial locations, leading to improperly reconstructed earth models.

A simple example of this can be seen in the difference between the two equations

$$u_{tt} = c^2(x)u_{xx} \tag{1.1}$$

and

$$u_{tt} = \frac{\partial}{\partial x}c^2(x)\frac{\partial}{\partial x}u. \tag{1.2}$$

Often these equations are used almost interchangebly. In fact, as of September 25, 2012 none of the Wikipedia pages: "Acoustic wave equation", "Wave equation" or "Acoustics" contain any mention of the second equation whatsoever. This begs the question: what is being missed? If $c^2(x)$ is discontinuous, as is often the case for physical earth models the second equation requires a much different continuity condition than the first equation. The spatial part of the first equation requires the function $u$ to be twice differentiable, while the second equation requires $u$ to be only once differentiable, but also requires that $c^2(x)u_x$ be continuous. Figures 1.1 to 1.6 show the difference between equations 1.1 and 1.2. We can see, in figures 1.1 and 1.2, the initial wave propagating to the left and right until the left-going wave hits a wavespeed discontinuity at $x = 0$. Figures 1.3 to 1.6 then show the kinked nature of the second type of equation, as well as the fact that the polarities of the reflected wave are opposite and the amplitudes of the trasnmitted waves are different. A more in depth discussion of this type of behaviour from a variational standpoint is included in the chapter on Galerkin methods.

This thesis will explore a type of numerical method that is capable of properly

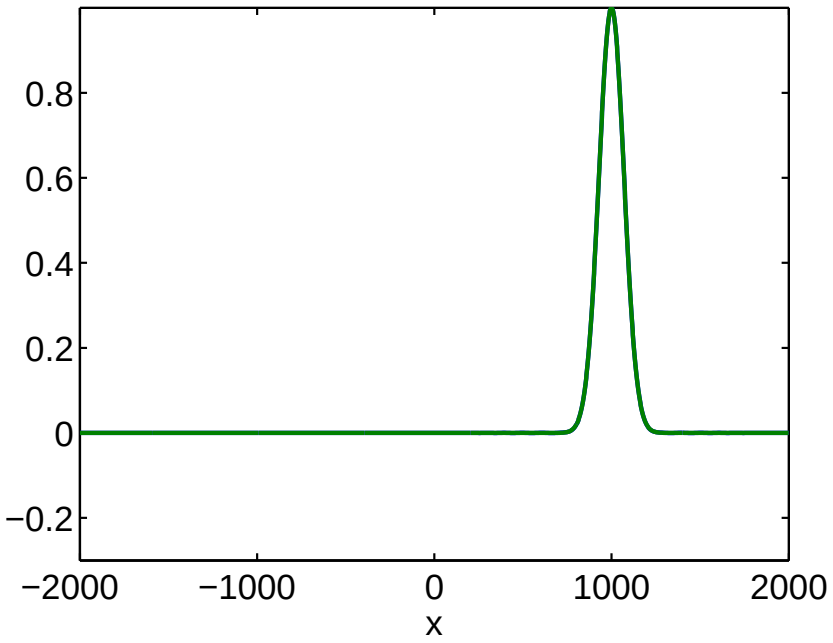



Figure 1.1: Comparison of solutions to equations 1.1 and 1.2

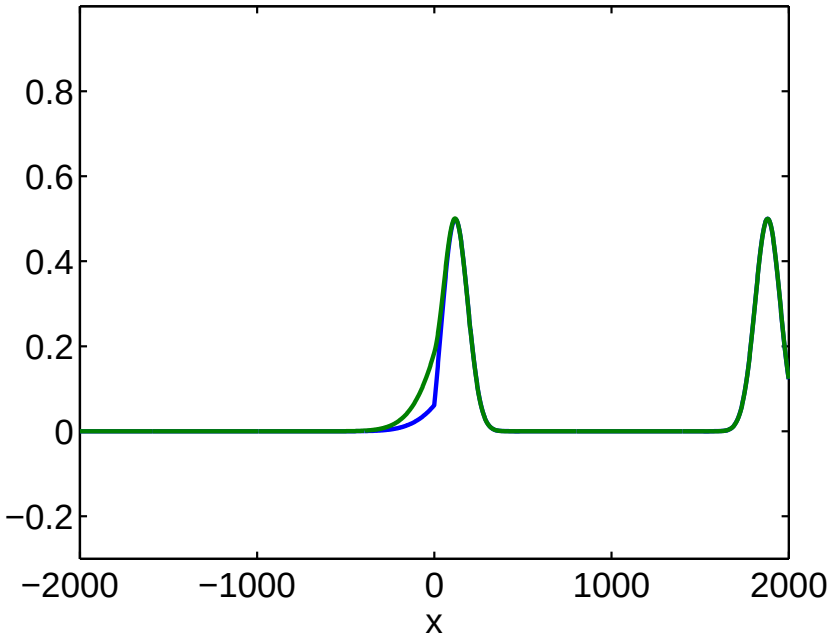



Figure 1.2: Comparison of solutions to equations 1.1 and 1.2

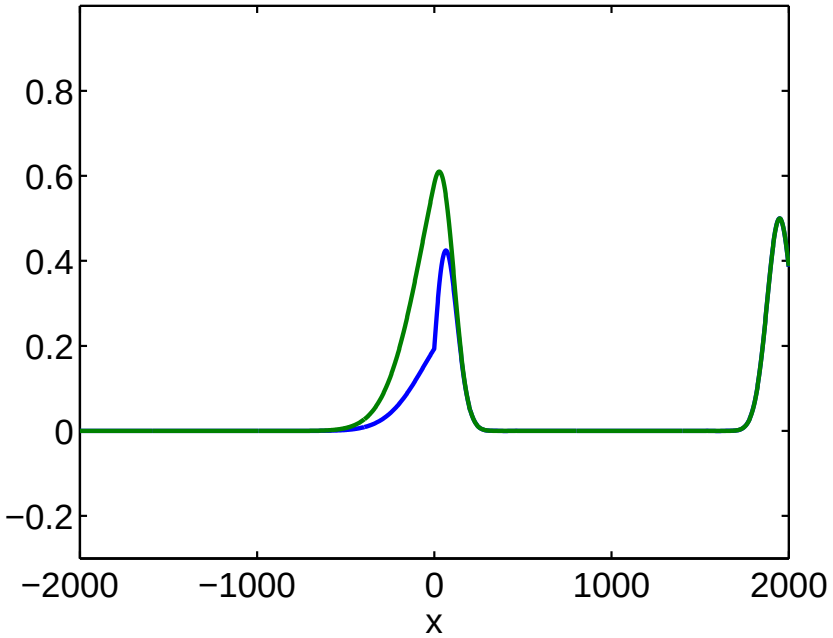



Figure 1.3: Comparison of solutions to equations 1.1 and 1.2

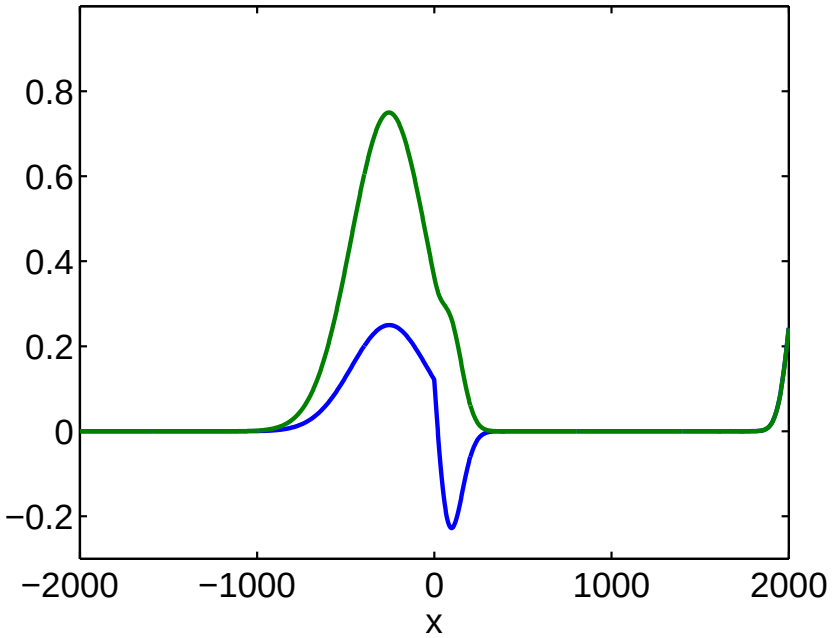



Figure 1.4: Comparison of solutions to equations 1.1 and 1.2

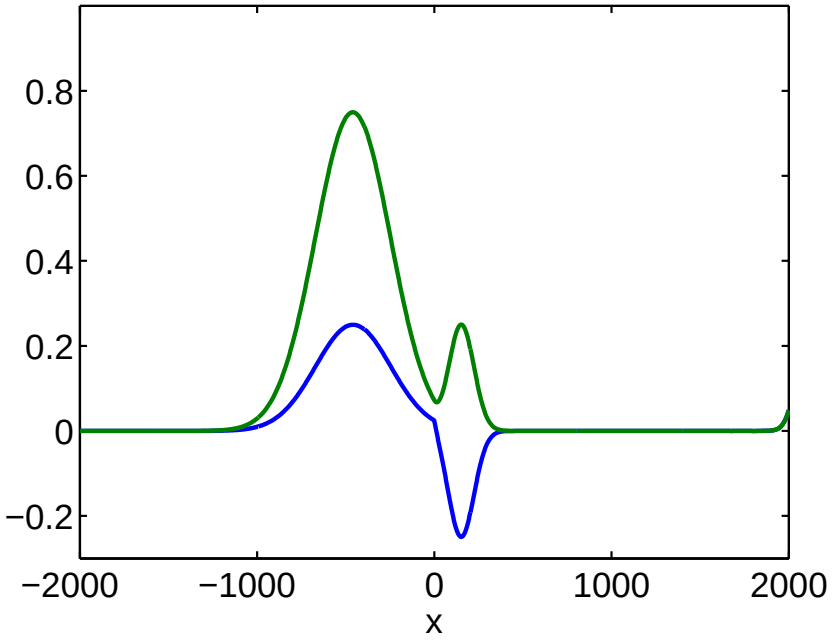



Figure 1.5: Comparison of solutions to equations 1.1 and 1.2

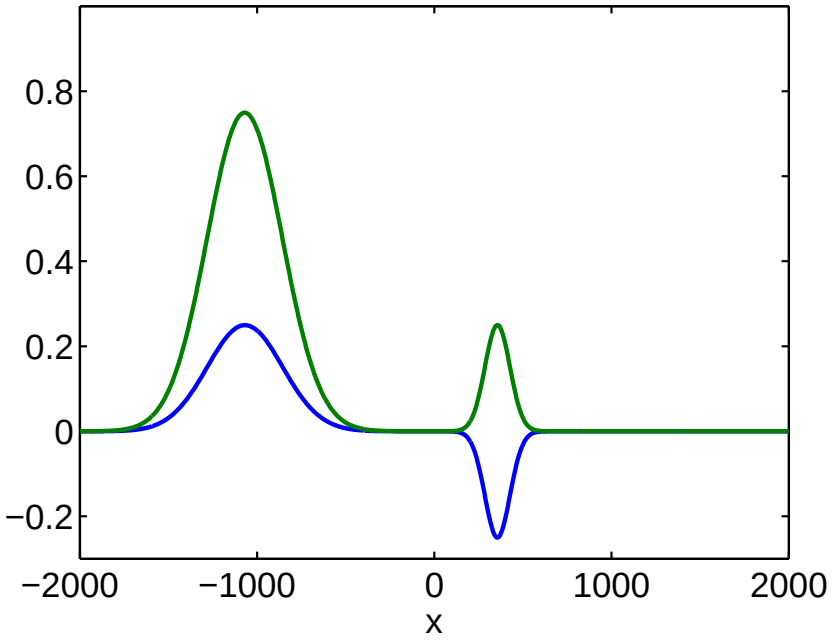



Figure 1.6: Comparison of solutions to equations 1.1 and 1.2

representing the second type of spatial differential operator by first converting the equation into the corresponding weak form, by a standard Galerkin procedure, and then implicitly enforcing boundary conditions at the material discontinuities. A discussion on the effects of applying global difference equations in place of the spatial derivatives is also included in the chapter on the elastic wave equation, which also includes a comparison between the numerical solutions obtained by finite-differences and those obtained by a Galerkin-type method.

## 1.2 Motivation

In conducting the research for this work, the main motivation came from two previous works. The main source, was that of E. Faccioli, F. Maggio, R. Paolucci and A. Quarteroni which culminated in the paper [8]. Their work laid the general framework for a high-order numerical method based on domain-decomposition combined with pseudospectral methods, and included a boundary treatment based on Lagrange multipliers that made it extremely efficient to model semi-infinite media by implicitly enforcing the boundary conditions as part of a damping term.

Several years later the work of Faccioli et al. was reproduced independently by D. Komatitsch and J.-P Vilotte in [16], using a different method for time-advancement. This became an issue of contention between the authors of the two papers, the public record of which is found in [10] and [25].

As is pointed out by Komatitsch, the method developed in both of the papers can be seen as an extension of the well-known h-p finite-element method developed by Babuska et al. in [2], wherein the classical finite-element method is extended to allow arbitrary order elements. That is, numerical accuracy can be increased by either increasing the number of elements, or the number of nodes that make up the element. This brings up questions of how to define the nodes on the element depending on the shape of the element and the type of nodes, both of which can be generalized to any number of methods that may, or may not, warrant a new classification.

Much of the work in the previous papers is done in an experimental setting and so contains little explanation of the actual implementation details and the problems that arise when attempting the implementation in practice. This work arose from trying to dig deeper into the actual implementation and construction of the numerical schemes for seismic wave propagation, including investigation into efficiency and feasibility on modern computer systems. Because much of the implementation is problem-dependent, due to the implicit enforcement of the absorbing boundary conditions, the chapters are organized in a manner such that the low-level nodal implementation are built-up and then referenced in the later chapters that take care of the analytic derivations for the acoustic, elastic and anelastic wave-equations, which themselves each depend on the previous chapter.

# Chapter 2

# Analytic Solutions

## 2.1 1D Scalar Waves

The simplest possible place to start studying wave equations is the constant coefficient equation

$$\begin{cases} u_{tt} = c^2 u_{xx}, & x \in \Omega,\ t > 0 \\ u(0,t) = u(L,t) = 0, & t \geq 0 \\ u(x, t=0) = u_0(x), & x \in \Omega \\ u_t(x, t=0) = u_1(x), & x \in \Omega \end{cases}$$

where the subscript denotes differentiation with respect to the specified variable, $c \in \mathbb{R}$ is the speed at which the wave travels, and $\Omega \in [0, L] \subseteq \mathbb{R}$ is domain of the problem being modelled. The solution $u(x,t)$ can be thought of as the vibration of a string with constant density, fixed at both ends and released from its initial position $u_0(x)$ with initial velocity $u_1(x) = 0$.

## 2.2 Separation of Variables

To go about solving the equation the method of separation of variables can be applied. That is, the solution is assumed to be of the form $u(x,t) = X(x)T(t)$. Substituting

this into the equation yields

$$T''X = c^2TX''$$

or

$$\frac{T''}{c^2T} = \frac{X''}{X}.$$

Because the left and right hand side are functions of different variables they must be equal to a constant $C$. This produces two ordinary differential equations. For the $x$-variable the result is the boundary value problem

$$\begin{cases} X''(x) = CX(x) \\ X(0) = X(L) = 0; \end{cases}$$

which gives 3 cases: $C < 0$, $C = 0$, and $C > 0$.

If $C = 0$ then

$$X(x) = c_1 + c_2x.$$

The left boundary condition is $X(0) = c_1 = 0$ and the right is $X(L) = c_2L = 0$ which implies $c_2 = 0$ and so the solution is identically zero.

If $C > 0$ then say $C = \lambda^2, \lambda \in \mathbb{R} \setminus \{0\}$ so $X'' = \lambda^2 X$. The general form of the solution to this equation is

$$X(x) = c_1e^{\lambda x} + c_2e^{-\lambda x}.$$

The left boundary condition then produces $X(0) = c_1 + c_2 = 0$ which implies that $c_1 = -c_2$. The right boundary condition then reads $X(L) = c_1(e^{\lambda L} - e^{-\lambda L}) = 0$, or that, $e^{\lambda L} = e^{-\lambda L}$ for all $\lambda$ which is a not possible.

Finally, the only possible case is when $C < 0$. Say $C = -\lambda^2, \lambda \in \mathbb{R} \setminus \{0\}$. Then the equation becomes $X'' + \lambda^2 X = 0$. The general solution is now

$$X(x) = c_1 \cos(\lambda x) + c_2 \sin(\lambda x)$$

Substituting in the left boundary conditions produces $X(0) = c_1 = 0$. The right boundary produces $X(L) = c_2 \sin(\lambda L) = 0$ which can only be true if $\lambda_k = k\pi/L$ for $k \in \mathbb{Z}$. The $\lambda_k$ are the eigenvalues of the system associated with the eigenfunctions $X_k(x) = \sin(\lambda_k x)$. For the $t$-variable the problem now reads $T'' = c^2 \lambda_k^2 T$. Like before, this has general solution

$$T(t) = a_k \cos(c\lambda_k t) + b_k \sin(c\lambda_k t).$$

Thus, the general solution to the constant coefficient wave equation in one spatial dimension is then a linear combination of the eigenfunctions multiplied by the time part of the solution. That is

$$u(x,t) = \sum_{k=1}^{\infty} \{a_k \cos(c\lambda_k t) + b_k \sin(c\lambda_k t)\} \sin(\lambda_k x).$$

All that is left is to apply the initial conditions. Setting $u(x, t = 0) = u_0(x)$ produces

$$\sum_{k=1}^{\infty} a_k \sin(\lambda_k x) = u_0(x).$$

Multiplying both sides by $\sin(\lambda_j x)$ and integrating from 0 to $L$ yields

$$a_k = \frac{2}{L} \int_0^L u_0(x) \sin(\lambda_k x) dx$$

Similarly, setting $u_t(x, t = 0) = u_1(x)$ produces

$$\sum_{k=1}^{\infty} c\lambda_k b_k \sin(\lambda_k x) = u_1(x).$$

Proceeding as in the case for the $a_k$'s it is found that

$$b_k = \frac{2}{c\lambda_k L} \int_0^L u_1(x) \sin(\lambda_k x) dx.$$

As an example, consider the problem

$$\begin{cases} u_{tt} = c^2 u_{xx}, & x \in [0, 1],\ t > 0 \\ u(0, t) = u(1, t) = 0, & t \geq 0 \\ u(x, t = 0) = u_0(x), & x \in [0, 1] \\ u_t(x, t = 0) = 0, & x \in [0, 1] \end{cases}$$

with

$$u_0(x) = \begin{cases} x, & x < 1/2 \\ 1 - x, & x > 1/2 \end{cases}$$

(ignore for the moment that $u_0$ is not actually differentiable at $x = 1/2$).

Clearly the $b_k$'s are all equal to zero. The $a_k$'s are found as

$$\begin{aligned} a_k &= 2 \left\{ \int_0^{1/2} x \sin(\lambda_k x) dx + \int_{1/2}^1 (1 - x) \sin(\lambda_k x) dx \right\} \\ &= \frac{4 \sin(k\pi/2)}{k^2 \pi^2} \end{aligned}$$

These are the Fourier sine coefficients of the hat function $u_0$. The final solution is then

$$u(x, t) = \sum_{k=1}^{\infty} \left\{ \frac{4 \sin(k\pi/2)}{k^2 \pi^2} \cos(c\lambda_k t) \right\} \sin(\lambda_k x).$$

Only the odd coefficients are non-zero, as is expected, since our initial condition $u_0$ is even about the midpoint of the interval. These correspond to the eigenfunctions

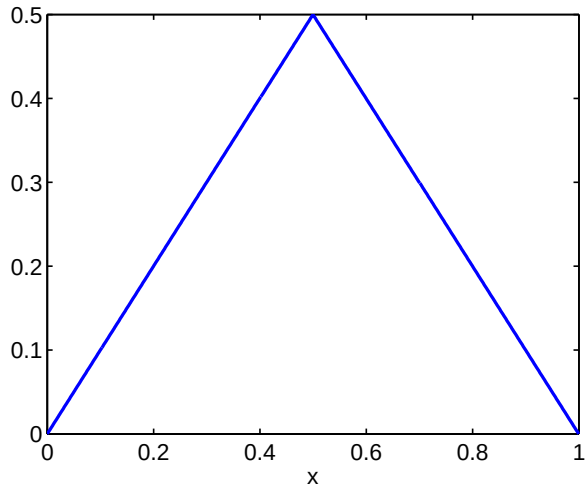



Figure 2.1: Hat function initial condition for example problem

that are also even about the midpoint. These modes have wavenumber equal to $k/2$ Hz. Figure 2.2 shows several of the contributing eigenfunctions.

## 2.3 d'Alembert's Formula

The example problem can also be solved by the formula of d'Alembert. To do so, the change of variables $\xi = x + ct$, $\eta = x - ct$ is applied, resulting in the new problem

$$\frac{\partial^2 u}{\partial \xi \partial \eta} = 0.$$

This has general solution $u(\xi, \eta) = \phi(\xi) + \psi(\eta)$. The initial conditions then read

$$u(x, t = 0) = u_0(x) = \phi(x) + \psi(x),$$
$$u_t(x, t = 0) = 0 = c(\phi(x) - \psi(x)).$$

Together these imply that $\phi(x) = \psi(x) = u_0(x)/2$ so the general solution is

$$u(x, t) = \frac{1}{2}(u_0(x + ct) + u_0(x - ct)).$$

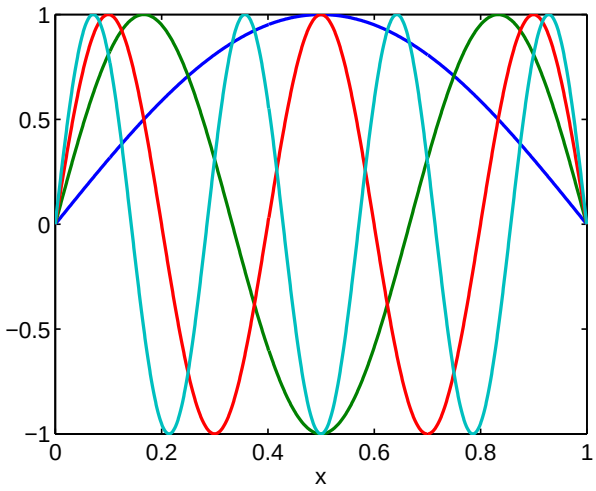



Figure 2.2: Contributing Fourier modes of the example problem.

This, however, does not satisfy the boundary conditions $u(0,t) = u(1,t) = 0$. To correct the situation write

$$u(x,t) = \frac{1}{2}(\hat{u}_0(x+ct) + \hat{u}_0(x-ct)).$$

where $\hat{u}_0$ is a periodic extension of $u_0$ that must satisfy

$$u(0,t) = \frac{1}{2}(\hat{u}_0(ct) + \hat{u}_0(-ct)) = 0,$$

and

$$u(1,t) = \frac{1}{2}(\hat{u}_0(ct) + \hat{u}_0(-ct)) = 0.$$

That is $\hat{u}_0(ct) = \hat{u}_0(-ct)$ and $\hat{u}_0(1+ct) = \hat{u}_0(1-ct)$, which implies that $\hat{u}_0$ is the even extension of $u_0$ with period 1.

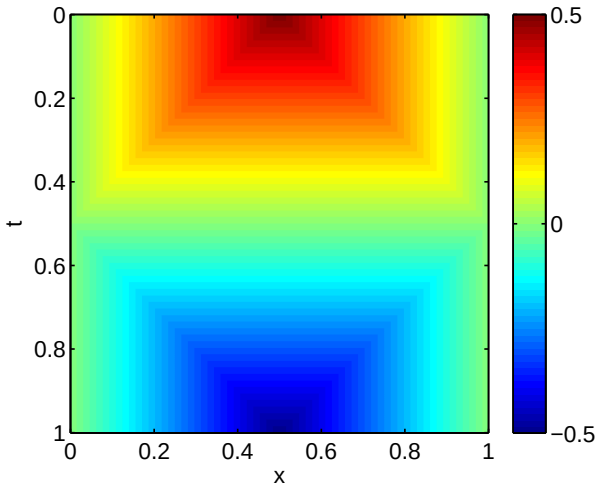



Figure 2.3: d'Alembert solution to example.

## 2.4 Error in Partial Sums

We now have a way of looking at the error of the partial sums of the first solution. Figure 2.3 shows the d'Alembert solution for all $x$ and all $t$. Notice that the lines along $t = 1/2 \pm x$ remain sharp as the solution propagates. This is what is expected to happen due to the inherent properties of the wave equation. Figures 2.4, 2.5 and 2.6 show the partial sum approximations for $N = 10, 100, 500$ and the absolute difference from the d'Alembert solution.

This illustrates the problem inherent to approximating kinked functions by smooth functions. Even though the solution appears to be well-approximated at $N = 100$, increasing the number of terms to 500 only increases the accuracy by a single order of magnitude. Figure 2.7 shows that increasing $N$ further to 1000 reduces the error even less than it did when increasing from 100 to 500. Careful examination of the

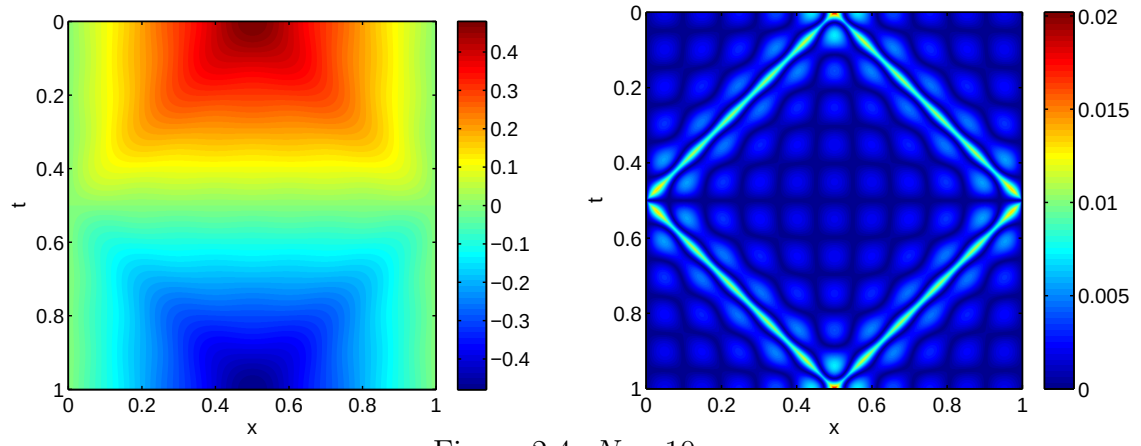



Figure 2.4: $N = 10$

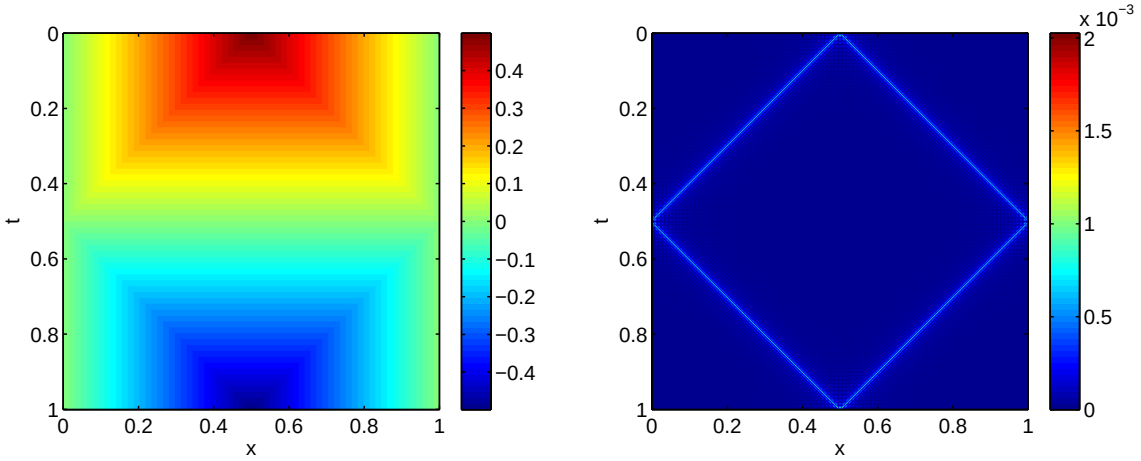



Figure 2.5: $N = 100$

error surfaces shows that the partial sums are converging as expected away from $t = 1/2 \pm x$ but along the kinks the convergence starts to stall. Figure 2.8 shows the first 5000 Fourier coefficients (only the 2500 non-zero odd coefficients are plotted). The relatively small increase in accuracy is because the amplitude of the coefficients is decreasing slower and slower as $N$ increases and so even an increase from 500 to 1000 coefficients contributes little to the overall approximation. This is visible in the plot of the amplitude of the first 5000 Fourier coefficients seen in figure 2.8 (only the

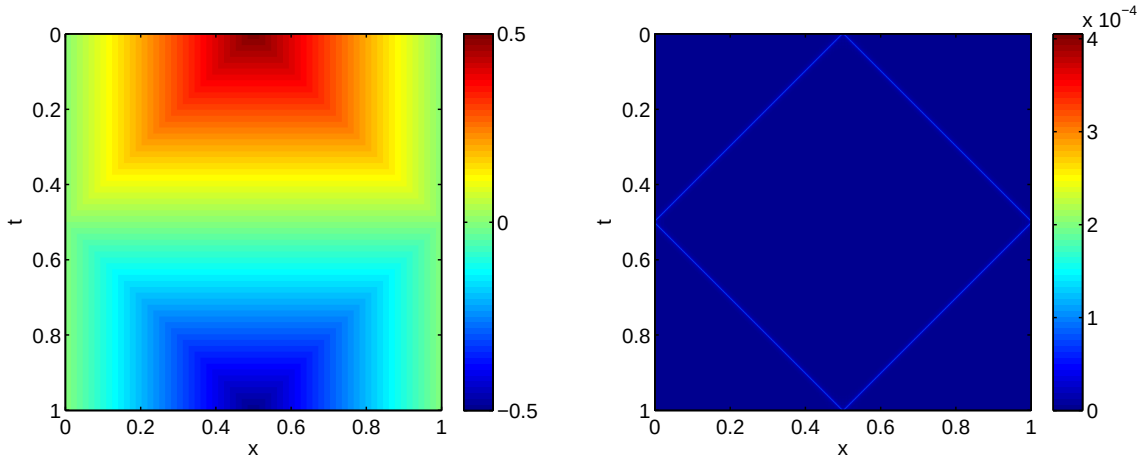



Figure 2.6: $N = 500$

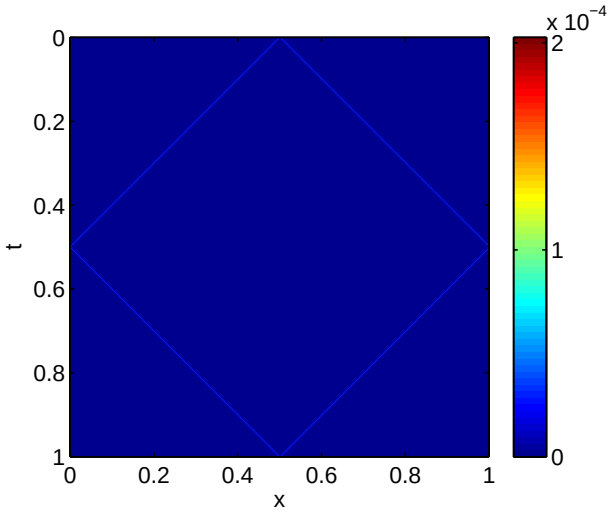



Figure 2.7: Error surface at $N = 1000$.

2500 non-zero odd coefficients are plotted).

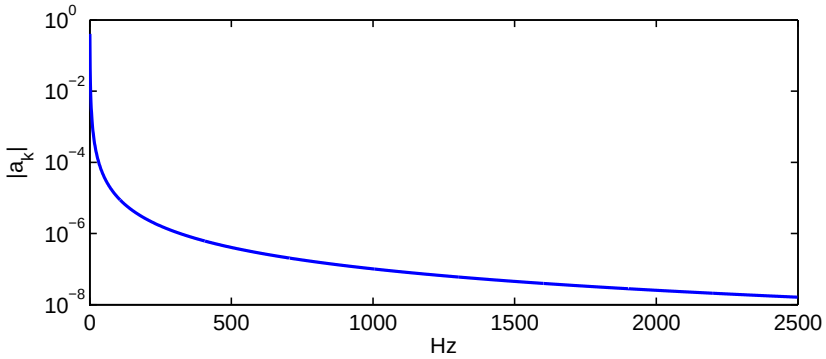



Figure 2.8: Amplitude of the non-zero $a_k$'s in the example problem.

# Chapter 3

# Nodal Approximations

## 3.1 Introduction

In the previous chapter we compared partial sums of series solutions to the d'Alembert solution to the constant coefficient wave equation in one spatial dimension. The method of partial sums, while an approximation, still relies on the use of analytic differentiation, and is therefore dependent entirely on manual symbolic analysis of the boundary and initial conditions.

Another method of approximation, not reliant on symbolic differentiation, is to replace the derivatives in the wave equation with approximations based on point-wise evaluations of the functions at hand, and then develop a scheme to further approximate how these point-wise values will evolve through time. This chapter will deal with this type of approximation, starting with small, local approximations to spatial derivatives and then moving through to global approximations, based on each point-wise function value.

The language we use is that of differentiation matrices. That is, the matrices that approximately represent continuous differential operators in the sense that, given a set of point-wise evaluations of a function, multiplication by the differentiation matrix

returns approximately the derivative at the same points.

## 3.2 Finite Differences

Suppose that we wish to approximate the derivative of a function at a set of points $\{x_i\}_{i=1}^{N_p}$ from a set of sampled values $\{f(x_i)\}_{i=1}^{N_p}$. A logical place to tackle such a problem is by using Taylor series to build an approximation. For instance, if the $x_i$ are evenly spaced, let $h = x_{i+1} - x_i$. Then we can write

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + O(h^3) \tag{3.1}$$

and

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) + O(h^3) \tag{3.2}$$

where $O(\cdot)$ is the Landau *big-O* notation for the error term understood to mean that, $f(x) = O(g(x))$ if, and only if, there exists a constant $M$ and a real number $x_0$ such that, for $x > x_0$, $|f(x)| \leq M|g(x)|$. Rearranging 3.1 and 3.2 we obtain the *forward-difference* and *backward-difference* formulas, respectively,

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h),$$

and

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + O(h).$$

Subtracting 3.2 from 3.1 cancels the even terms (including the zeroth) leaving

$$f(x+h) - f(x-h) = 2hf'(x) + O(h^3);$$

rearranging produces the *centered-difference* formula for the first derivative

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + O(h^2).$$

Note that we have now decreased the power in the error term. By choosing the grid small enough, so that $h << 1$, it is expected that the error will be small enough so that the results are meaningful. Similarly, adding 3.2 to 3.1 cancels the odd terms terms (not including the zeroth) and leaves

$$f(x+h) + f(x-h) = 2f(x) + h^2 f''(x) + O(h^4).$$

Rearranging we obtain the *centered-difference* formula for the second derivative

$$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} + O(h^2).$$

While this method may seem appealing for its rather elegant approach it becomes quite labourious once we try to increase the order in the error term further.

One such reason for doing this is to maintain a level of accuracy at the endpoints of the sampling interval. Suppose we wish to approximate the first derivative to second order at each of the $x_i$. The centered-difference formula will work for the $2^{nd}$ through $(N-1)^{st}$ points, but fails at the first and last points since we only have one-sided information about the function. We seek a formula for $f'(x_1)$ using information from $f$ at $x_1, x_2$ and $x_3$. The Taylor series for $f(x+2h)$ is

$$f(x+2h) = f(x) + 2hf'(x) + 2h^2 f''(x) + O(h^3). \tag{3.3}$$

Then 3.3 minus 4 times 3.1 yields

$$\begin{aligned} f(x+2h) - 4f(x+h) =& f(x) + 2hf'(x) + 2h^2 f''(x) + O(h^3) \\ & - 4f(x) - 4hf'(x) - 2h^2 f''(x) + O(h^3) \\ =& -3f(x) - 2hf'(x) + O(h^3) \end{aligned}$$

Rearranging produces the second order forward-difference formula

$$f'(x_i) = -\frac{3f(x_i) - 4f(x_{i+1}) + f(x_{i+2})}{2h} + O(h^2).$$

At the right end of the interval we simply substitute $-h$ to obtain the backward-difference formula

$$f'(x_i) = \frac{3f(x_i) - 4f(x_{i-1}) + f(x_{i-2})}{2h} + O(h^2).$$

The natural thing to do now is to look at the matrix that acts on the values $\{f(x_i)\}_{i=1}^N$ and returns approximately $\{f'(x_i)\}_{i=1}^N$ for a fixed error. For two points all we can do is first order since all we have is endpoints. This matrix is

$$\frac{1}{h}\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

For second-order error we can move up to three points, producing

$$\frac{1}{2h}\begin{bmatrix} -3 & 4 & -1 & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & 1 & -4 & 3 \end{bmatrix}.$$

Beyond this, we can no longer resort to strictly centered or one-sided approximations. The correct way to go about building higher-order differentiation matrices from Taylor series is to use as many of the left (right) endpoints as possible while

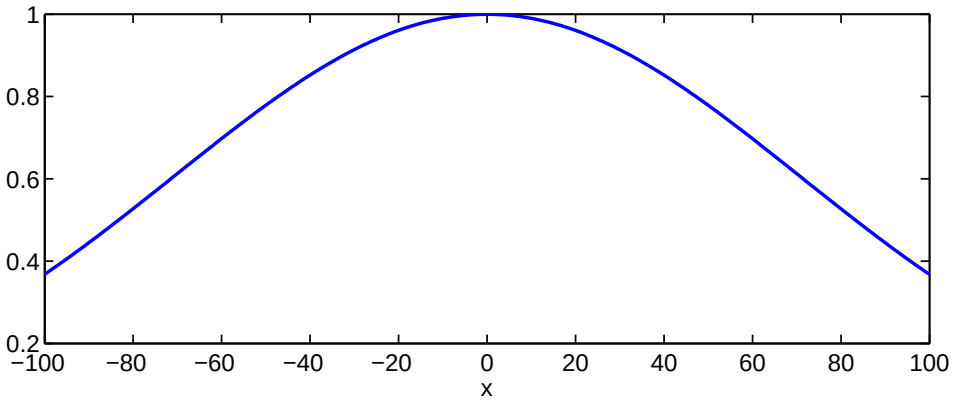
moving down (up) the rows until the centered formulas will fit. For the fourth order matrix this means that the second and second-to-last rows are neither forward nor centered and are instead *lop-sided*. The fourth-order differentiation matrix is then

$$\frac{1}{12h}\begin{bmatrix} -25 & 48 & -36 & 16 & -3 & & \\ -3 & -10 & 18 & -6 & 1 & & \\ 1 & -8 & 0 & 8 & -1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & 1 & -8 & 0 & 8 & -1 \\ & & -1 & 6 & -18 & 10 & 3 \\ & & 3 & -16 & 36 & -48 & 25 \end{bmatrix}.$$

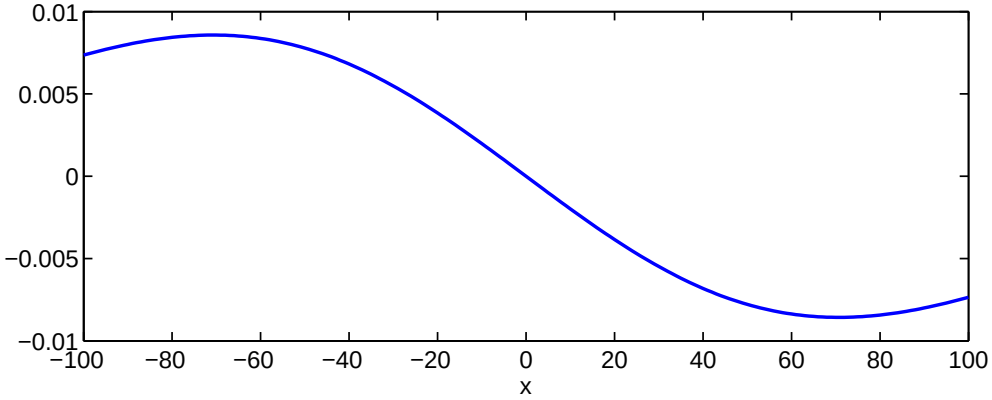
Careful observation will reveal that building differentiation matrices in this manner can eventually lead to problems due to the fact that the coefficients in the error terms for non-centered formulas grow extremely large for higher order formulas. Figure 3.1 shows this anomaly. Even though the error is on the order of $10^{-10}$ the saw-tooth regions near the endpoints can grow large enough to cause problems for numerical boundary conditions. As is suggested in [11], one strategy is to increase the order of approximations strictly at the non-centered rows to offset the larger coefficients in the error terms.

## 3.3 Pseudospectral Methods

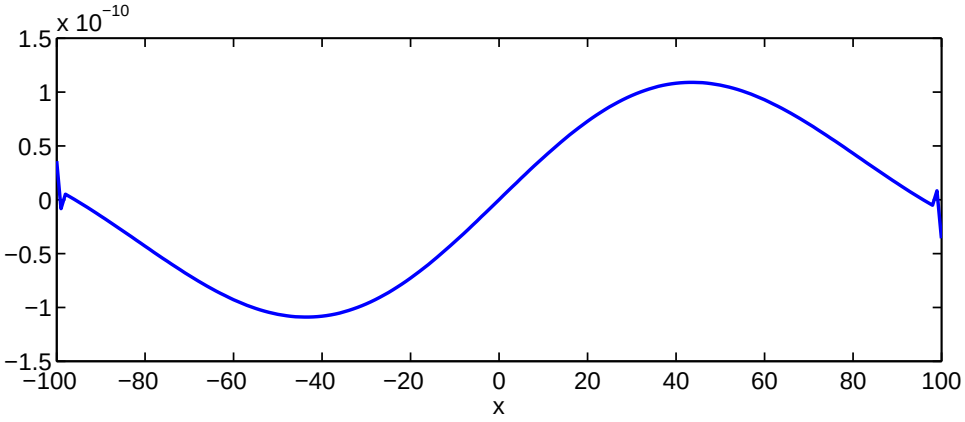
Another way to build differentiation matrices is to fit a polynomial about a number of points in a set of nodes and then differentiate the interpolating polynomial. These are termed pseudospectral-differentiation matrices. There's a fair bit of discrepancy




(a) The function.




(b) The approximate derivative.




(c) The error.

Figure 3.1: A good approximation with a bad error. Notice the sawtooth-like anomalies near the endpoints.

in the use of the term *pseudospectral*, but they generally fall into the category of solutions to partial differential equations, where it is understood to mean solutions derived from assuming an eigenfunction expansion of the form

$$u(\mathbf{x}) = \sum_{n=0}^{\infty} a_n \phi_n(\mathbf{x}). \tag{3.4}$$

For time-dependent problems the coefficients $a_n$ are generally assumed to be functions of $t$. When the functions $\phi_n(\mathbf{x})$ in 3.4 form an orthogonal basis the expansion is known as a *generalized-Fourier-series* in the sense that the basis functions are no longer strictly sines and cosines. Perhaps the most well-known form of the pseudospectral method in geophysical wave propagation stems from the choice of the standard Fourier basis for the $\phi_n$, and, in practice, this is probably the best place to start when seeking pseudospectral solutions due to the availability of the fast-Fourier-transform for computing numerical derivatives and convolutional sums, but is not the approach taken here. Another family of methods comes from choosing the $\phi_n$'s to be from a basis of orthogonal polynomials such as *Chebyshev* or *Legendre* polynomials. These are defined as the eigenfunctions of singular Sturm-Liouville differential equations but may be derived by many methods. All of the methods mentioned so-far are termed *modal* methods where the unknowns are the coefficients of the expansion and, thus, require transformations to and from sampled values of the functions we are interested in approximating and the appropriate coefficients.

The method used here is a so-called *nodal* method based on interpolation formulas

that make use of the Lagrange polynomials

$$l_j(x) = \prod_{i \neq j} \frac{(x - x_i)}{(x_j - x_i)}$$

for a set of nodes $\{x_i\}_{i=1}^{N}$.

These provide a very convenient way of generating interpolating polynomials due to their discrete-delta property, $l_i(x_j) = \delta_{ij}$. Given a set of nodes $\{x_i\}_{i=1}^{N_p}$ and functions values $\{f(x_i)\}_{i=1}^{N_p}$ the Lagrange polynomials allow us to write

$$f(x) \approx \sum_{i=1}^{N_p} f(x_i) l_i(x).$$

For brevity, we will derive only the three point formula about $(x_{i-1}, f(x_{i-1}))$,$(x_i, f(x_i))$, $(x_{i+1}, f(x_{i+1}))$, but a similar process may derive an approximation given any number of points. Neglecting the error term and substituting our three points into the above formula gives us the following approximation

$$f(x) \approx \frac{(x - x_i)(x - x_{i+1})}{(x_{i-1} - x_i)(x_{i-1} - x_{i+1})} f(x_{i-1}) + \frac{(x - x_{i-1})(x - x_{i+1})}{(x_i - x_{i-1})(x_i - x_{i+1})} f(x_i) + \frac{(x - x_{i-1})(x - x_i)}{(x_{i+1} - x_{i-1})(x_{i+1} - x_i)} f(x_{i+1}),$$

differentiating, yields the first derivative approximation

$$f'(x) \approx \frac{2x - x_i - x_{i+1}}{(x_{i-1} - x_i)(x_{i-1} - x_{i+1})} f(x_{i-1}) + \frac{2x - x_{i-1} - x_{i+1}}{(x_i - x_{i-1})(x_i - x_{i+1})} f(x_i) + \frac{2x - x_{i-1} - x_i}{(x_{i+1} - x_{i-1})(x_{i+1} - x_i)} f(x_{i+1})$$

If we evaluate at $x_i$ and choose a uniform grid spacing $(x_{i+1} - x_i) = h$ then the above equation simplifies into the familiar form,

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$$

Differentiating once more produces

$$f''(x_i) \approx \frac{2f(x_{i-1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})} + \frac{2f(x_i)}{(x_i-x_{i-1})(x_i-x_{i+1})} + \frac{2f(x_{i-1})}{(x_{i+1}-x_i)(x_{i+1}-x_{i-1})},$$

or, after again making the substitution $(x_{i+1} - x_i) = h$,

$$f''(x_i) \approx \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2}.$$

We now have the tools to make fully populated differentiation matrices defined using Lagrange polynomials. That is, the approximate derivative at each point will consist of information from all other points in the set of nodes. Since the Lagrange polynomials are unique and form a basis for all polynomials of degree $N_p - 1$ the error associated with these differentiation matrices is the error associated with interpolating $f$ by a degree $N_p - 1$ polynomial. In fact, the $(N_p - 1)$ by $(N_p - 1)$ pseudospecral differention matrix differentiates the sampled values of an $(N_p - 1)^{th}$ degree polynomial to within machine-precision. Figure 3.2 shows the convergence to machine-precision of the 2-norm of the difference between a pseudocpectral approximation to the derivative of a $10^{th}$-degree polynomial and the analytic derivative as the dimension of the differentiation matrix increases. Notice that when the dimension reaches the degree of the polynomial, the error is immediately on the order of machine-precision.

According to the Weierstrass approximation theorem, any continuous function on an interval can be uniformly approximated by polynomial to within any error bound.

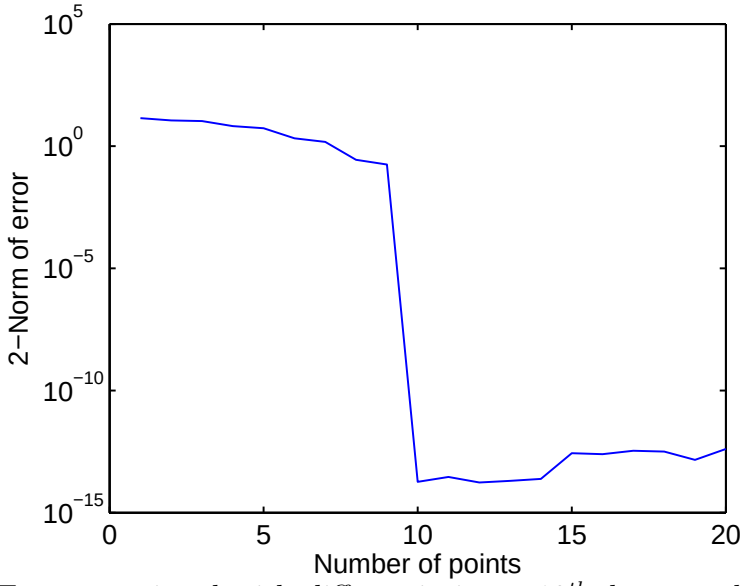



Figure 3.2: Error associated with differentiating a $10^{th}$-degree polynomial with increasing dimension of pseudospectral differentiation matrix.

The error is closely related to the choice of the interpolation nodes and the resulting interpolating polynomial basis functions. For example, consider a set of equispaced points. As we can see in figure 3.3, increasing the number of nodes quickly forces the polynomials to act very badly near the ends of the interval in order to compensate for the increase in order.

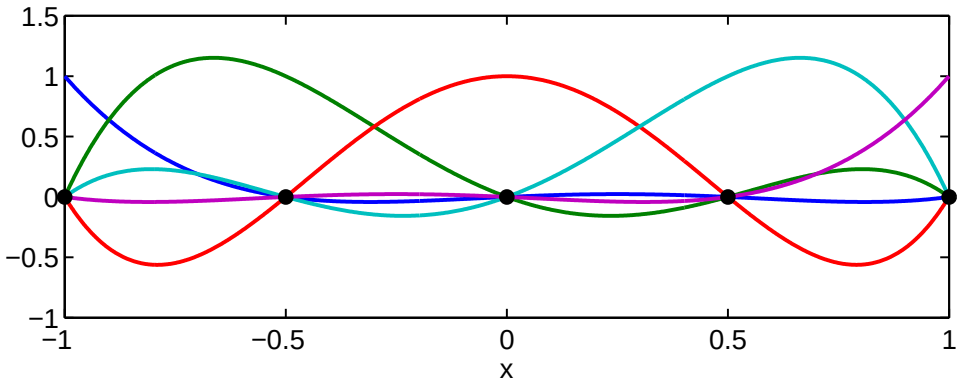
The behaviour of the interpolating polynomials at the endpoints of an equispaced grid is known as *Runge's phenomenon* and stems from the error term

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i),$$

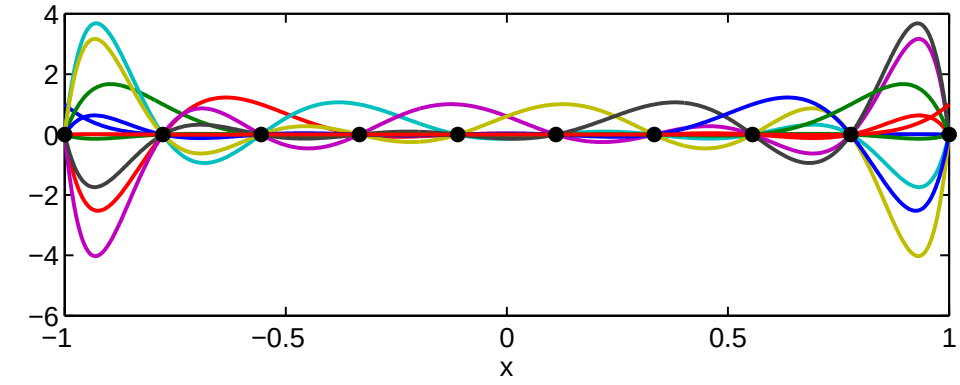
for some $\xi \in \{\min_i\{x, x_i\}, \max_i\{x, x_i\}\}$.

Runge's classical example is the function

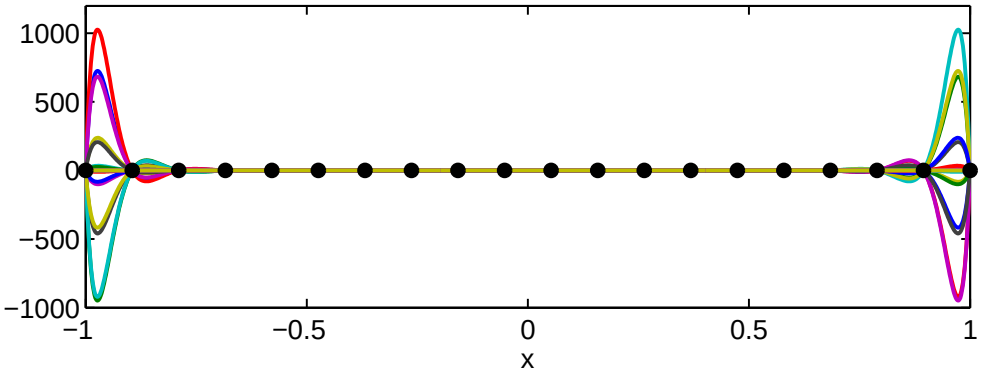
$$f(x) = \frac{1}{1 + 25x^2}, \quad x \in [-1, 1].$$




(a) 5 Lagrange polynomials




(b) 10 Lagrange polynomials




(c) 20 Lagrange polynomials

Figure 3.3: Families of Lagrange interpolating polynomials for increasing number of zeros.

The upper bound on the error goes to $\infty$ as $n \to \infty$. Figure 3.4 shows the increased ringing near the end of the interval as the number of nodes is increased.

The choice of nodes from which to sample our functions is clearly significant. Choosing the nodes to minimize the presence of Runge's phenomenon leads to the zeros of $(1-x^2)P_n'$, where $P_n$ is the $n^{th}$ Chebyshev or Legendre polynomial defined as the eigenfunctions of the singular Sturm-Liouville problem

$$\frac{d}{dx}(1-x^2)w(x)\frac{d}{dx}P_n^{(\alpha,\beta)}(x) + n(n+\alpha+\beta+1)w(x)P_n^{(\alpha,\beta)}(x) = 0, x \in [-1,1],$$

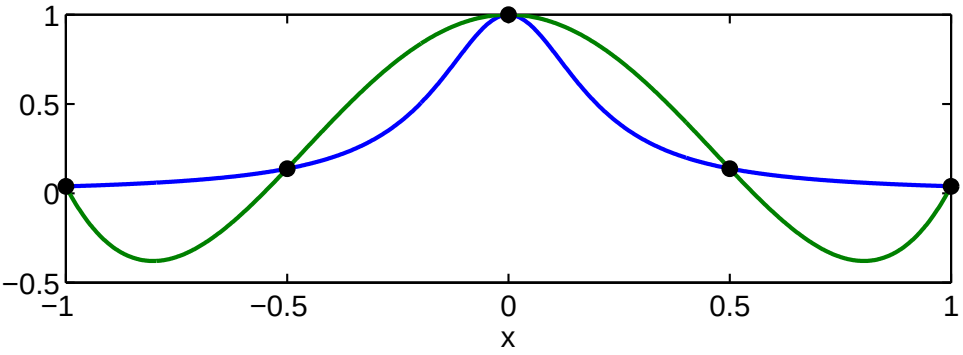
$$w(x) = (1-x)^\alpha(1+x)^\beta.$$

chosen by $\alpha = \beta = -1/2$ or $\alpha = \beta = 0$, for the Chebyshev or Legendre polynomials, respectively. These points are known as the Chebyshev or Legendre Gauss-Lobatto nodes [12].

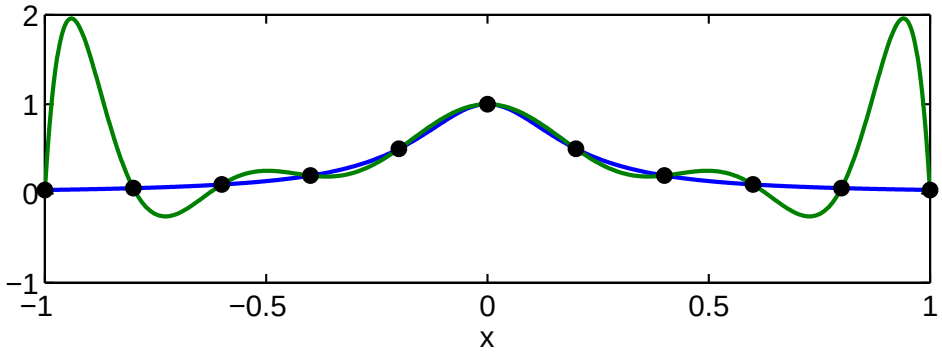
Let us first provide some motivation by constructing an appropriate set of interpolation nodes for the interval $[-1,1]$.

First of all, note that the $n^{th}$ degree polynomial that interpolates $n+1$ nodes is unique. Call this polynomial $p(x)$. If it was not unique then there would exist another we will call $q(x)$. Form the polynomial $r(x) = p(x) - q(x)$. This polynomial is of degree $n$ since $p$ and $q$ are of degree $n$, but at the $n+1$ nodes $p(x) = q(x)$ and so $r(x) = 0$ implying $r(x)$ has $n+1$ roots, a contradiction. Thus we can assume that our polynomial is of the form

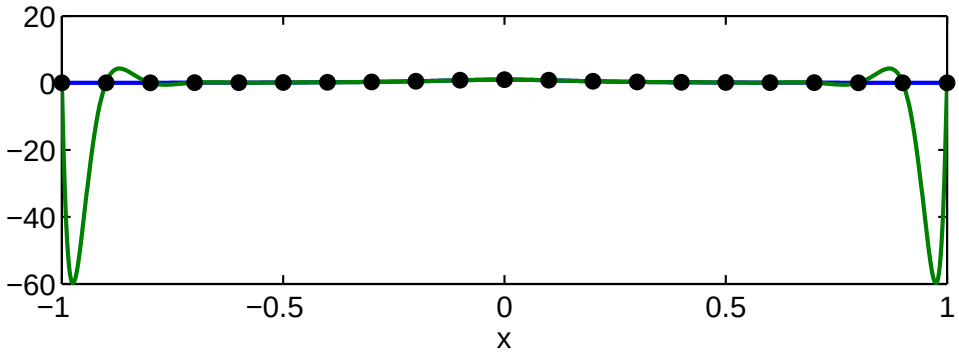
$$p(x) = \sum_{i=0}^{n} a_n x^n$$




(a) 5 Nodes




(b) 11 Nodes




(c) 21 Nodes

Figure 3.4: Runge phenomenon for increasing order of interpolation. The blue line is the function, and the green line is the interpolating polynomial.

Then $p(x)$ interpolates $f(x)$ at the $n+1$ nodes $\{x_i\}_{i=0}^n$ so $p(x_i) = f(x_i)$. That is

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix}$$

The matrix on the left is the *generalized Vandermonde matrix*. Since $x_i \in [-1, 1]$ the matrix becomes increasingly ill-conditioned as $n$ increases. The determinant of the matrix is

$$\prod_{i \neq j} (x_j - x_i).$$

The fact that we are trying to solve a linear system suggests that the best set of nodes would be those that maximize the determinant [13].

This is done approximately by the $N+1$ Chebyshev-Gauss-Lobatto (CGL) nodes, which can be computed analytically in Matlab using Code 1.

```
x = -cos(pi*(0:N)/N);
```

Code 1: Matlab function for computing the Chebyshev-Gauss-Lobatto nodes.

The Legendre-Gauss-Lobatto (LGL) nodes are those that exactly maximize the determinant, but unfortunately have no closed form and so must be computed by a numerical root-finding method.

Code 2 computes the LGL nodes by a Newton method using the asymptotic relation in [17] as a starting point. At the same time, it computes the Legendre

polynomials using the recursion relation

$$\begin{cases} P_0(x) = 1, \\ P_1(x) = x, \\ (n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x). \end{cases}$$

obtained by applying a Gram-Schmidt procedure to the monomial basis.

```
function x = LGLNodes(N)
x = -cos(((0:N/2)+.25)*pi/N - 3./(8*N*pi*((0:N/2)+.25)));
xold = 0;
V = zeros(N+1,length(x));
while max(abs(x-xold))>eps
   V(1,:) = 1;
   V(2,:) = x;
   for n = 3:N+1
      V(n,:) =((2*n-3)*x.*V(n-1,:)-(n-2)*V(n-2,:))/(n-1);
   end
   xold = x;
   x=xold-( x.*V(N+1,:)-V(N,:) )./( (N+1)*V(N+1,:) );
end
x = [x,-x(ceil(N/2):-1:1)];
```

Code 2: Matlab function for computing the Legendre-Gauss-Lobatto nodes.

Figure 3.5 shows the result of interpolating a compactly-supported function using different choices of clustered nodes and the mitigated Runge's phenomenon.

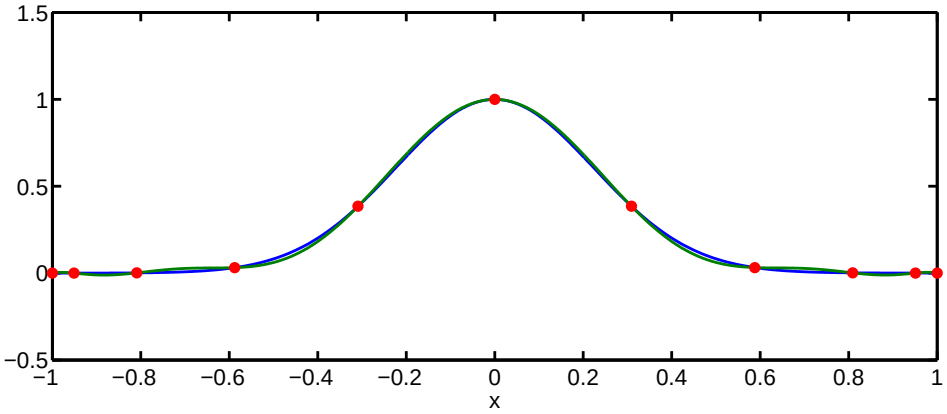



Figure 3.5: Interpolation of the function $exp(-10x^2)$ on clustered nodes.

Returning to the definition of the differentiation matrices, consider the interpolary expansion

$$u(x) = \sum_{n=0}^{\infty} a_n \phi_n(x) \quad \forall x_i, \quad i = 0, ..., N,$$

We can write this in matrix form

$$\begin{pmatrix} u(x_0) \\ u(x_1) \\ \vdots \\ u(x_N) \end{pmatrix} = \begin{pmatrix} \phi_0(x_0) & \cdots & \phi_N(x_0) \\ \phi_0(x_1) & \cdots & \phi_N(x_1) \\ \vdots & \ddots & \vdots \\ \phi_0(x_N) & \cdots & \phi_N(x_N) \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_N \end{pmatrix}$$

so then

$$\begin{pmatrix} a_0 \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} \phi_0(x_0) & \cdots & \phi_N(x_0) \\ \phi_0(x_1) & \cdots & \phi_N(x_1) \\ \vdots & \ddots & \vdots \\ \phi_0(x_N) & \cdots & \phi_N(x_N) \end{pmatrix}^{-1} \begin{pmatrix} u(x_0) \\ \vdots \\ u(x_N) \end{pmatrix}$$

The matrix equation for the nodal values of the derivative is then

$$\begin{pmatrix} u'(x_0) \\ \vdots \\ u'(x_N) \end{pmatrix} = \begin{pmatrix} \phi_0'(x_0) & \cdots & \phi_N'(x_0) \\ \phi_0'(x_1) & \cdots & \phi_N'(x_1) \\ \vdots & \ddots & \vdots \\ \phi_0'(x_N) & \cdots & \phi_N'(x_N) \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_N \end{pmatrix}$$

$$= \begin{pmatrix} \phi_0'(x_0) & \cdots & \phi_N'(x_0) \\ \phi_0'(x_1) & \cdots & \phi_N'(x_1) \\ \vdots & \ddots & \vdots \\ \phi_0'(x_N) & \cdots & \phi_N'(x_N) \end{pmatrix} \begin{pmatrix} \phi_0(x_0) & \cdots & \phi_N(x_0) \\ \phi_0(x_1) & \cdots & \phi_N(x_1) \\ \vdots & \ddots & \vdots \\ \phi_0(x_N) & \cdots & \phi_N(x_N) \end{pmatrix}^{-1} \begin{pmatrix} u(x_0) \\ \vdots \\ u(x_N) \end{pmatrix}$$

Choosing a basis $\{\phi_n\}_{n=0}^N$ (such as the Legendre or Chebyshev polynomials) and set of points $\{x_n\}_{n=0}^N$ (such as the LGL or CGL nodes) fully defines the pseudospectral differentiation matrix

$$D = \begin{pmatrix} \phi_0'(x_0) & \phi_1'(x_0) & \cdots & \phi_N'(x_0) \\ \phi_0'(x_1) & \phi_1'(x_1) & \cdots & \phi_N'(x_1) \\ \vdots & \ddots & \ddots & \vdots \\ \phi_0'(x_N) & \phi_1'(x_N) & \cdots & \phi_N'(x_N) \end{pmatrix} \begin{pmatrix} \phi_0(x_0) & \phi_1(x_0) & \cdots & \phi_N(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_N(x_1) \\ \vdots & \ddots & \ddots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_N(x_N) \end{pmatrix}^{-1}$$

We compute this matrix for the Legendre polynomials using two functions in Matlab in Code 3 and 4.

```
function [V Vx] = legVVx(x)
N = length(x);
V = zeros(N);
Vx = zeros(N);
Vxx = zeros(N);
V(:,1) = 1;
V(:,2) = x;
for n = 3:N
    V(:,n) =((2*n-3)*x'.*V(:,n-1)-(n-2)*V(:,n-2))/(n-1);
end
Vx(:,1) = 0;
Vx(:,2) = 1;
```

```
for n = 2:N-1;
    Vx(:,n+1) = (2*n-1)*V(:,n) + Vx(:,n-1);
end
```

Code 3: Compute the matrices of nodal values of the Legendre polynomials and their first derivatives.

```
function Dx = legDMat(x)
[V Vx] = legVVx(x);
Dx = Vx/V;
```

Code 4: Compute the pseudospectral differentiation matrix.

To compare the accuracies of pesudospectral differentiation matrices with that of various orders of finite-difference matrices, consider the function in equation 3.5, shown in figure 3.6 alongside its derivative.

$$f(x) = x(1 + \sin(10\pi \exp(-10x^2))) \tag{3.5}$$

Figure 3.7 shows the 2-norm of the difference between the approximate derivative, obtained by applying the various pseudospectral and finite-difference matrices, and the analytic derivative. Figure 3.8 shows the function 3.5 and the points used in the approximations, the computation times and the final errors for Chebyshev and $8^{th}$ order finite differences. We can see that if the concern of the model is accuracy and not speed, the two methods are fairly similar, but the pseudospectral method requires

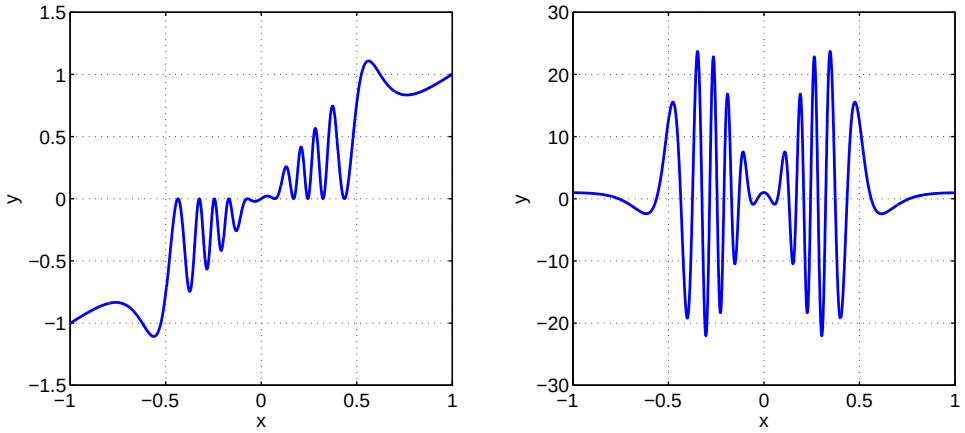



Figure 3.6: The function in equation 3.5 and its derivative

much fewer points. In terms of gridpoints-per-wavelength, Chebyshev points require at least $\pi$ gridpoints-per-wavelength, but are $\pi/2$ times less dense near the endpoints than they are in the middle of the interval [23]. Thus, they require 2 grid-points-wavelength in the center of the interval as is expected from the Nyquist-Shannon sampling theorem.

As a final note it should be pointed out that while it is possible to define higher-order differentiation matrices $D^{(n)}$ that compute the $n^{th}$ nodal derivative by taking powers of the first derivative matrix, this is generally a bad idea for a large number of nodes. Instead the matrices should be computed either using the same procedure we used to derive the first-order differentiation matrix or via special recursion formulas specific to the choice of basis functions [27].

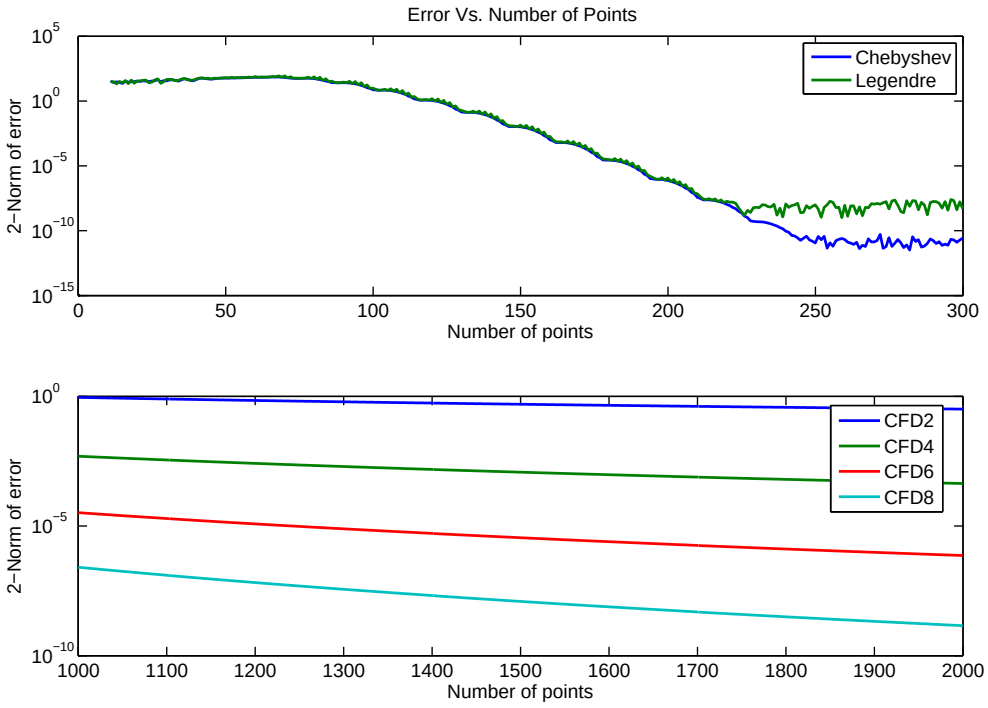



Figure 3.7: Various derivative approximations

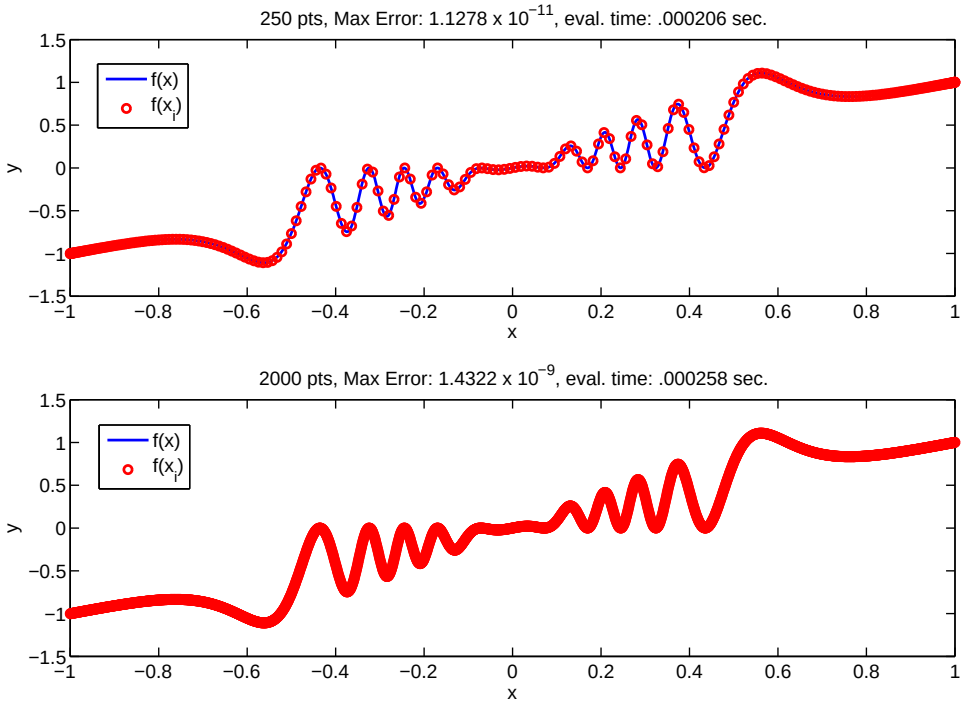



Figure 3.8: Comparison of pseudospectral differentiation to $8^{th}$-order finite-differences.

## 3.4 Integration Weights

Associated with a set of Gauss-Lobatto nodes $\{x_i\}_{i=0}^{N}$ is a set of numerical integration weights $\{w_i\}_{i=0}^{N}$. The weights are computed analytically in both the Chebyshev and Legendre cases. For the CGL nodes the weights are such that

$$\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}} dx = \frac{\pi}{N} \left\{ \sum_{i=1}^{N-1} f(x_i) w_i + \frac{f(x_1)w_1 + f(x_N)w_N}{2} \right\}$$

is exact when $f$ is a polynomial of degree less then or equal to $2N-1$. For a general function $f$ the weights would then be $(\sqrt{1-x^2})\pi/N$, where $\sqrt{1-x^2}$ is the weight function associated with the Sturm-Liouville equation for the Chebyshev functions. The integration weights are computed in Code 5.

```
function x = CGLNodesAndWeights(N)
x = CGLNodes;
w = sqrt(1-x.^2)*pi/N;
```

Code 5: Matlab function for computing the Chebyshev-Gauss-Lobatto nodes and weights.

For the Legendre polynomials the Sturm-Liouville weight function is equal to 1 and so the quadrature formula is

$$\int_{-1}^{1} f(x) dx = \sum_{i=0}^{N} f(x_i) w_i.$$

This is, again, exact for polynomials of degree less than or equal to $2N-1$. The integration weights are computed from the values of the $N^{th}$ Legendre polynomial evaluated at the LGL nodes in Code 6.

```
function [x w] = LGLNodesAndWeights(N)
x = LGLNodes(N);
V = zeros(N+1,length(x));
V(1,:) = 1;
V(2,:) = x;
for n = 3:N+1
  V(n,:) =((2*n-3)*x.*V(n-1,:)-(n-2)*V(n-2,:))/(n-1);
end
w = 2./(N*(N+1)*V(N+1,:).^2);
```

Code 6: Matlab function for computing the Legendre-Gauss-Lobatto nodes and weights.

## 3.5 Higher Dimensions

The 2D versions of the pseudospectral differentiation matrices and integration weights are obtained by defining their 1D counter-parts along each dimension and then taking Kronecker tensor products. This can be done in Matlab in several ways. For the integration weights, assume we have two column vectors $w_x$ and $w_z$ containing the weights in the $x$ and $z$ directions, respectively, associated with the vectors $x = LGLNodes(Nx)$ and $z = LGLNodes(Nz)$ of dimension $Nx$ and $Nz$. Then the 2D integration weights can be computed as $W = w_z * w_x.'$ giving an $Nz$-by-$Nx$ matrix. So, if $U$ is the column-major-storage of the matrix of nodal values of a function $u(x, z)$ we can perform integration by taking the dot product with the column-major-storage-vector version of $W$.

Computing the differentiation matrices is a little different. Suppose again that we are working with the vector $U$ and wish to compute the matrices $Dx$ and $Dz$ that discretely compute $\partial_x$ and $\partial_z$, respectively. We first compute the 1D

differentiation matrices $Dx1D$ and $Dz1d$. The matrix $Dx$ can be computed as $Dx = kron(Dx1d, eye(Nz))$ and $Dz = kron(eye(Nx), Dz1d)$.

# Chapter 4

# Galerkin Methods

## 4.1 The Calculus of Variations

The Galerkin method is an analytic procedure for transforming the strong form of a (partial) differential equation into the weak form, which stems from the fundamental lemma of the calculus of variations. The lemma states that if a function $u$ is k-times differentiable on an interval $[a, b]$ and

$$\int_a^b u(x)v(x)dx = 0, \quad \forall v \in C_0^k([a, b]),$$

where, $C_0^k([a, b])$ is the space of k-times differentiable functions vanishing on the boundary of $[a, b]$, then $u \equiv 0$ on $[a, b]$.

Let us employ the lemma to derive the Euler-Lagrange equations from an action functional for a function $u(x, t)$. We make use of the notation

$$\frac{\partial u}{\partial x} = u_x$$

and

$$\frac{\partial u}{\partial t} = u_t$$

The action functional is

$$S[u] = \int_{t_0}^{t_1} \int_{x_0}^{x_1} \mathcal{L}\left(x, t, u, u_x, u_t\right) dxdt,$$

where $\mathcal{L}$ is a Lagrangian function. The idea is to find the extrema of $S$ by fixing the endpoints of $u$ and varying it slightly inside the interval $\Omega = [x_0, x_1] \times [t_0, t_1]$ by adding small variations $\varepsilon v(x,t)$ that disappear on the boundary $\partial\Omega$. So, if $u$ is an extrema of the action functional, then

$$\frac{dS}{d\varepsilon}[u + \epsilon v]\Big|_{\varepsilon=0} = 0.$$

That is,

$$\int_\Omega \frac{d}{d\varepsilon}\mathcal{L}(x, t, u + \varepsilon v, u_x + \varepsilon v_x, u_t + \varepsilon v_t)d\Omega\Big|_{\varepsilon=0} = 0$$

or

$$\int_\Omega \left(\frac{\partial \mathcal{L}}{\partial u}v + \frac{\partial \mathcal{L}}{\partial u_x}v_x + \frac{\partial \mathcal{L}}{\partial u_t}v_t\right) d\Omega = 0$$

This is known as the weak form of the problem. Note that the integral can be written as

$$\int_\Omega \left(\frac{\partial \mathcal{L}}{\partial u}v + \left(\frac{\partial \mathcal{L}}{\partial u_x}, \frac{\partial \mathcal{L}}{\partial u_t}\right) \cdot \nabla v\right) d\Omega.$$

Where

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial t}\right)$$

Green's theorem states that

$$\int_\Omega \left(\frac{\partial \mathcal{L}}{\partial u_x}, \frac{\partial \mathcal{L}}{\partial u_t}\right) \cdot \nabla v d\Omega$$

$$= \oint_{\partial\Omega} \left(\frac{\partial \mathcal{L}}{\partial u_x}, \frac{\partial \mathcal{L}}{\partial u_t}\right) \cdot \mathbf{n} v dS - \int_\Omega \nabla \cdot \left(\frac{\partial \mathcal{L}}{\partial u_x}, \frac{\partial \mathcal{L}}{\partial u_t}\right) v d\Omega$$

where $\mathbf{n}$ is the outward pointing unit normal vector. Since $v$ vanishes on the boundary the surface integral term disappears and we are left with

$$\int_\Omega \left(\frac{\partial \mathcal{L}}{\partial u} - \frac{\partial}{\partial x}\frac{\partial \mathcal{L}}{\partial u_x} - \frac{\partial}{\partial t}\frac{\partial \mathcal{L}}{\partial u_t}\right) v d\Omega = 0$$

Since $v$ is arbitrary, applying the fundamental lemma of the calculus of variations yields the Euler-Lagrange equations

$$\frac{\partial \mathcal{L}}{\partial u} - \frac{\partial}{\partial x}\frac{\partial \mathcal{L}}{\partial u_x} - \frac{\partial}{\partial t}\frac{\partial \mathcal{L}}{\partial u_t} = 0.$$

which is known as the strong-form of the problem.

As an aside, if $\Omega$ is the union of two regions, $\Omega = \Omega_1 \cup \Omega_2$, then the surface integral does not disappear on the interior boundary. Assume that $\Omega_1 = [-L, 0] \times [0, T]$ and $\Omega_2 = [0, L] \times [0, T]$. Then the normal vector along the boundary $x = 0$ is equal to $\mathbf{n} = [-1, 0]$, in $\Omega_1$ and $\mathbf{n} = [1, 0]$, in $\Omega_2$. So the surface integral becomes the sum of the contributions on either side. That is

$$\begin{aligned}
&\oint_{\partial\Omega} \left( \frac{\partial \mathcal{L}}{\partial u_x}, \frac{\partial \mathcal{L}}{\partial u_t} \right) \cdot \mathbf{n} v dS \\
&= -\int_0^T \lim_{x \to 0^-} \frac{\partial \mathcal{L}}{\partial u_x} v dt + \int_0^T \lim_{x \to 0^+} \frac{\partial \mathcal{L}}{\partial u_x} v dt \\
&= \int_0^T \left( \lim_{x \to 0^+} \frac{\partial \mathcal{L}}{\partial u_x} - \lim_{x \to 0^-} \frac{\partial \mathcal{L}}{\partial u_x} \right) v dt.
\end{aligned}$$

As before, since $v$ is arbitrary, this implies

$$\lim_{x \to 0^+} \frac{\partial \mathcal{L}}{\partial u_x} = \lim_{x \to 0^-} \frac{\partial \mathcal{L}}{\partial u_x},$$

which are known as the interface conditions.

Let's now derive the wave equation for a vibrating string using the Euler-Lagrange equations. The Lagrangian function $\mathcal{L}$ of a dynamic system is defined to be the total kinetic energy $T$ minus the total potential energy $V$. So the action integral is

$$\int_\Omega (T - V)\, d\Omega.$$

Let the length of the string be $L$ and let the displacement from equilibrium at some point $x \in [0, L]$ at some time $t \in [0, T]$ be a function $u(x, t)$. Let the mass of an infinitesimal length of string be $\rho(x)$ and the modulus of elasticity be $k(x)$. Then, the total kinetic energy of the string is

$$T = \frac{1}{2} \int_0^L \rho(x) u_t^2 dx.$$

The potential energy is equal to

$$V = \frac{1}{2} \int_0^L k(x) \left(1 + u_x^2\right) dx.$$

So the action integral is

$$S[u] = \frac{1}{2} \int_0^T \int_0^L \left\{ \rho(x) \left( \frac{\partial u}{\partial t} \right)^2 - k(x) \left( 1 + \left( \frac{\partial u}{\partial x} \right)^2 \right) \right\} dx dt$$

The Lagrangian is

$$\frac{1}{2} \left\{ \rho(x) \left( \frac{\partial u}{\partial t} \right)^2 - k(x) \left( 1 + \left( \frac{\partial u}{\partial x} \right)^2 \right) \right\}.$$

So

$$\frac{\partial \mathcal{L}}{\partial u} = 0,$$

$$\frac{\partial \mathcal{L}}{\partial u_x} = -k(x) u_x$$

and

$$\frac{\partial \mathcal{L}}{\partial u_t} = \rho(x) u_t.$$

Substituting these into the Euler-Lagrange equations produces

$$-\frac{d}{dx}(-k(x) u_x) - \frac{d}{dt} \rho(x) u_t = 0$$

or since the density does not depend on time

$$\rho(x)u_{tt} = \frac{\partial}{\partial x}(k(x)u_x).$$

If the density varies, but the bulk modulus is constant then this can be written as

$$u_{tt} = c^2(x)u_{xx}$$

where

$$c(x) = \sqrt{\frac{k}{\rho(x)}}$$

is the propagation speed of the waves in the string.

If the region $\Omega$ is split into two regions as before, the interface conditions would then be

$$\lim_{x\to 0^-} k(x)u_x = \lim_{x\to 0^+} k(x)u_x$$

in the case of varying bulk modulus, or

$$\lim_{x\to 0^-} u_x = \lim_{x\to 0^+} u_x$$

if it is constant.

To see the difference in the two cases consider the two problems

$$\begin{cases} u_{tt} = c^2(x)u_{xx} & , \ x \in \mathbb{R} \\ u(x, t = 0) = u_0(x) & , \ x \in \mathbb{R} \\ u_t(x, t = 0) = 0 & , \ x \in \mathbb{R} \end{cases} \tag{4.1}$$

and

$$\begin{cases} u_{tt} = \frac{\partial}{\partial x}c^2(x)\frac{\partial u}{\partial x} & , \ x \in \mathbb{R} \\ u(x, t = 0) = u_0(x) & , \ x \in \mathbb{R} \\ u_t(x, t = 0) = 0 & , \ x \in \mathbb{R} \end{cases} \tag{4.2}$$

where

$$c^2(x) = \begin{cases} c_1^2 & , x < 0 \\ c_2^2 & , x > 0 \end{cases}$$

Note that $c_1$ and $c_2$ arise from different choices of $\rho(x)$ and $k(x)$ as defined before, but for our purposes we want them to be the same. Technically 4.1 is the wave equation with piecewise continuous density and constant bulk modulus so

$$c_1^2 = \frac{k_0}{\rho_1}, \quad c_2^2 = \frac{k_0}{\rho_2}$$

while 4.2 is the equation for constant density and piecewise continuous bulk modulus, implying

$$c_1^2 = \frac{k_1}{\rho_0}, \quad c_2^2 = \frac{k_2}{\rho_0}.$$

Together this yields

$$k_1\rho_1 = k_2\rho_2 = k_0\rho_0.$$

We can write either problem as

$$\begin{cases} u_{tt} = c_1^2 u_{xx}, & x < 0 \\ u_{tt} = c_2^2 u_{xx}, & x > 0 \end{cases}$$

where the difference between $c_1$ and $c_2$ may come from either varying density or bulk modulus.

As in chapter 2, assume the solution in either region, $x < 0$ or $x > 0$, is of the form

$$u(x,t) = \Phi(x)\Gamma(t)$$

This yields

$$\frac{c_k^2\Phi''}{\Phi} = \frac{\Gamma''}{\Gamma} = \lambda^2, \quad k = 1, 2,$$

and so,

$$\begin{cases} \Phi'' = -(\lambda/c_k)^2\Phi, \\ \Gamma'' = -\lambda^2\Gamma. \end{cases}$$

For simplicity, assume the solutions are constant multiples of

$$\Gamma(t) = \exp(\pm i\lambda t)$$

$$\Phi(x) = \exp(\pm i\lambda s_k x)$$

where

$$s_k = \frac{1}{c_k}, \quad k = 1, 2.$$

Thus,

$$u(x,t) = \begin{cases} A_1 \exp(i\lambda(t \pm s_1 x)), & x < 0 \\ A_2 \exp(i\lambda(t \pm s_2 x)), & x > 0, \end{cases}$$

where $A_1, A_2$ are in $\mathbb{R}$. If we assume that solution is initially waves travelling in the positive $x$ direction with unit amplitude, then the full solution will be of the form

$$u(x,t) = \begin{cases} \exp(i\lambda(t - s_1 x)) + R\exp(i\lambda(t + s_1 x)), & x < 0 \\ T\exp(i\lambda(t - s_2 x)), & x > 0, \end{cases}$$

where $R$ is the coefficient of the reflected wave, and $T$ is the coefficient of the trasmitted wave.

For both the case of discontinuous density, and discontinuous bulk modulus, we require the solution to be continuous at the interface $x = 0$. That is

$$(1 + R)\exp(i\lambda t) = T\exp(i\lambda t)$$

or $1 + R = T$.

For discontinuous density, we require that the first spatial derivative be continuous at $x = 0$. That is,

$$-i\lambda s_1 \exp(i\lambda t) + i\lambda s_1 R \exp(i\lambda t) = -i\lambda s_2 T \exp(i\lambda t)$$

or

$$-i\lambda s_1 (1 - R) \exp(i\lambda t) = -i\lambda s_2 T \exp(i\lambda t).$$

Making the substitution $T = 1 + R$ produces

$$s_1(1 - R) = s_2(1 + R).$$

Solving for $R$ and then, again, using $T = 1 + R$ yields

$$R = \frac{s_1 - s_2}{s_1 + s_2}, \quad T = \frac{2s_1}{s_1 + s_2}$$

or,

$$R = \frac{c_2 - c_1}{c_1 + c_2}, \quad T = \frac{2c_2}{c_1 + c_2}.$$

For discontinuous bulk modulus, we require that the bulk modulus times the first spatial derivative at $x = 0$ be continuous. It follows from above that the second condition required to solve for the transmission and reflection coefficients is

$$-i\lambda s_1 k_1 (1 - R) \exp(i\lambda t) = -i\lambda s_2 k_2 T \exp(i\lambda t)$$

or, after some cancellations and the fact that $k_i = \rho_i c_i^2$,

$$c_1(1 - R) = c_2 T.$$

Solving the system of equations for $R$ and $T$ yields,

$$R = \frac{c_1 - c_2}{c_1 + c_2}, \quad T = \frac{2c_1}{c_1 + c_2}.$$

So, we can see that, regardless of whether $c_1$ or $c_2$ is larger, the sign of the reflection coefficient will be the opposite in the case of varying bulk modulus than it will be for varying density. Further, the solution in the case of varying bulk modulus only exhibits $C^0$ continuity, that is, it contains functions that are "kinked", and as we have already seen, may cause problems for numerical methods assuming a higher level of continuity than is actually present.

## 4.2 Weak Forms of Differential Equations

The Galerkin method can be thought of as the Calculus of Variations performed backwards. That is, instead of solving the strong form of the problem, the Galerkin method seeks to find the weak form, and solve that instead.

To apply the method to a differential equation of the form $\mathcal{L}[u] = f$, defined on a region $\Omega$ where $\mathcal{L}$ is a linear spatial differential operator, a space of functions $V$ is chosen in which $u$ and $v$ will reside. $u$ is then written as a linear combination of the basis functions of the space,

$$u = \sum_{i=1}^{\infty} a_i \phi_i,$$

and $v$ is chosen from amongst the basis functions.

The measure of the residual $R[u] = \mathcal{L}[u] - f$ should then theoretically be zero. That is,

$$\int_{\Omega} R[u] v dx = 0, \quad \forall v \in C_0^1(\Omega),$$

or,

$$\sum_{i=1}^{\infty} a_i \int_\Omega \mathcal{L}[\phi_i]\phi_j d\Omega = \int_\Omega f\phi_j d\Omega, \quad \forall j.$$

For certain problems, where the strong form corresponds to the Euler-Lagrange equations of minimum potential energy, the Galerkin method is equivalent to a Rayleigh-Ritz minimization technique [3].

To make this a numerical method, the infinite sums must be truncated at some large $N$, the integrals evaluated, and re-written as a large $N$-dimensional system of equations to be solved for the unknown $a_i$'s,

$$\begin{pmatrix} \int_\Omega \mathcal{L}[\phi_1]\phi_1 d\Omega \cdots \int_\Omega \mathcal{L}[\phi_N]\phi_1 d\Omega \\ \vdots \quad \ddots \quad \vdots \\ \int_\Omega \mathcal{L}[\phi_1]\phi_N d\Omega \cdots \int_\Omega \mathcal{L}[\phi_N]\phi_N d\Omega \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} \int_\Omega f\phi_1 d\Omega \\ \vdots \\ \int_\Omega f\phi_N d\Omega \end{pmatrix}.$$

or,

$$K\mathbf{a} = \mathbf{f}.$$

Clearly, the method is greatly dependent on how the choice of basis function affects the solvability of the resulting matrix equation. Choosing the basis functions to be sines or cosines (depending on the boundary conditions) would make the matrix $K$ diagonal, so long as the differential operator is of the form

$$\mathcal{L}[u] = c_0 u + c_1 u' + ... + c_n u^{(n)}$$

where the $c_i's$ are all constants.

Another method arises from choosing the basis functions to be compactly supported piecewise polynomials designed to control the bandwidth of the matrix. These

are termed basis-splines or b-splines. The functions themselves are considered global, but are defined by a small set of nodes corresponding to the order of the basis function. For example, with three nodes $\{x_{i-1}, x_i, x_{i+1}\}$, define a basis function

$$\phi_i(x) = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}}, & x \in [x_{i-1}, x_i) \\ \frac{x_{i+1}-x}{x_{i+1}-x_i}, & x \in [x_i, x_{i+1}]. \end{cases}$$

The function $\phi_i$, a first-order b-spline, only overlaps with $\phi_{i-1}$ and $\phi_{i+1}$ and so the matrix $K$ will be tridiagonal.

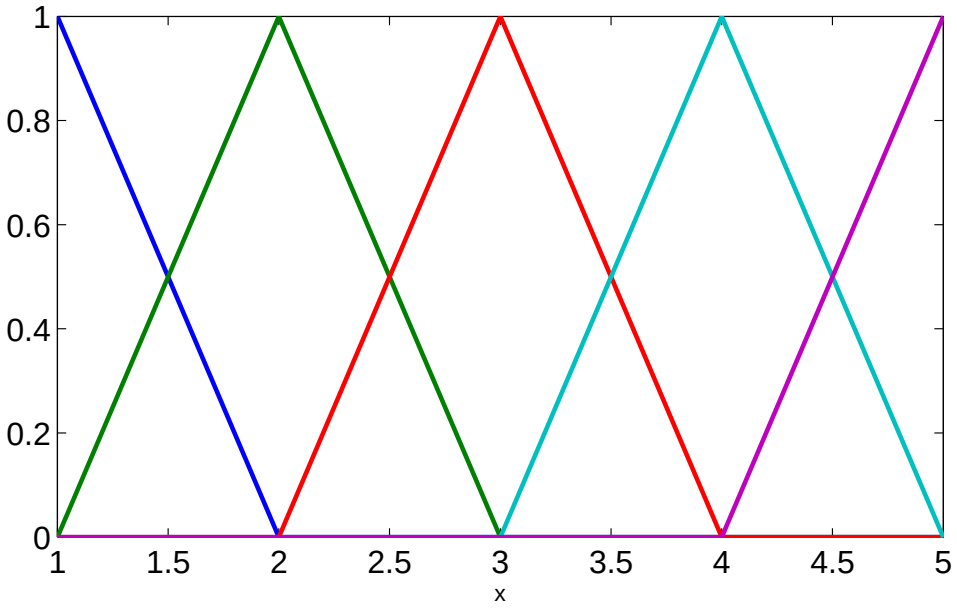



Figure 4.1: Linear b-splines.

It's fairly easy to see that low-order b-splines can cause problems when the individual pieces of the function are low-enough order that they do not have enough derivatives to be non-zero when applying the differential operator. Increasing the order of b-splines, however, requires more nodes in their definition, and so will in-

crease the bandwidth of the matrix $K$. There is way around this, as we can see in the following example.

Consider the problem

$$\begin{cases} -u''(x) = f, x \in (0, L), \\ u(0) = u(L) = 0, \end{cases} \tag{4.3}$$

where $f \in \mathbb{R}$ is a constant. We will employ first-order b-splines defined on equally spaced nodes, $x_{i+1} - x_i = \Delta x$, for all $i$, as our basis functions. The boundary conditions can be enforced implicitly by removing the first and last b-splines from the functions considered. The elements of the matrix $K$ are then

$$K_{j,i} = -\int_0^L \phi_i'' \phi_j dx$$

Differentiating $\phi_i$ (in the distributional sense) gives

$$\phi_i'(x) = \begin{cases} \frac{1}{x_i - x_{i-1}}, & x \in (x_{i-1}, x_i) \\ \frac{-1}{x_{i+1} - x_i}, & x \in (x_i, x_{i+1}) \end{cases}$$

$$= \frac{1}{\Delta x}\{H(x - x_{i-1}) - 2H(x - x_i) + H(x - x_{i+1})\}$$

where $H(x)$ is the Heaviside step function

$$H(x) = \begin{cases} 1, & x > 0 \\ 0, & x < 0. \end{cases}$$

Differentiating again, yields

$$\phi''(x) = \frac{1}{\Delta x}\{\delta(x - x_{i-1}) - 2\delta(x - x_i) + \delta(x - x_{i+1}\},$$

where $\delta(x - x_i)$ is the Dirac delta function defined by

$$\int \delta(x - x_i) g(x) dx = g(x_i).$$

The integrals inside the matrix $K$ then become

$$-\int_0^L \phi_i'' \phi_j dx = -\frac{1}{\Delta x}\int_0^L \{\delta(x-x_{i-1}) - 2\delta(x-x_i) + \delta(x-x_{i+1}\}\phi_j dx.$$

The only time the $\phi_j' s$ contribute to this integral is when $j$ is equal to either $i-1$, $i$ or $i+1$, resulting in

$$-\int_0^L \phi_i'' \phi_j dx = \begin{cases} \frac{-1}{\Delta x}, & j = i-1 \\ \frac{2}{\Delta x}, & j = i \\ \frac{-1}{\Delta x}, & j = i+1 \end{cases}.$$

We're not done, however. The right hand side of equation 4.3 is equal to

$$\int_0^L f\phi_j dx = f\int_{x_{j-1}}^{x_{j+1}} \phi_j dx,$$

which is just the area of a triangle with base-length $2\Delta x$ and height 1. Thus, the integral is equal to $f\Delta x$. Dividing both sides by $\Delta x$ gives

$$\frac{1}{\Delta x^2}\begin{pmatrix} 2 & -1 & 0 & \cdots & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & \cdots & \cdots & 0 & -1 & 2 \end{pmatrix}\begin{pmatrix} a_1 \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} f \\ \vdots \\ f \end{pmatrix}.$$

Note that since the basis functions are essentially discrete-delta functions on the defining nodes, $a_i = u(x_i)$ and the final equation is equivalent to a standard second-order finite difference method. We could have reached the same conclusion by performing the integration by parts

$$-\int_0^L \phi_i'' \phi_j dx = \int_0^L \phi_i' \phi_j' dx$$

$$= \begin{cases} \frac{-1}{\Delta x}, & j = i-1 \\ \frac{2}{\Delta x}, & j = i \\ \frac{-1}{\Delta x}, & j = i+1 \end{cases},$$

which is exactly how the Dirac delta function is equal to the derivative of the Heaviside step function anyways.

As we have seen, the method is extremely general and various choices of basis function can lead to any number of different numerical schemes such as spectral methods, pseudospectral methods, finite-element methods and even finite-difference methods. To make the procedure even more general the functions $v$ can be chosen to be different from the $\phi_j$'s resulting in a Petrov-Galerkin method [19].

## 4.3 Nodal Galerkin Methods

The nodal Galerkin method is a variant of the pseudospectral method where-in the basis functions are defined nodally, that is, as functions defined on a set of points with the discrete-delta property that they vanish at all but a single node. A simple way of defining nodal basis functions is via Lagrange polynomials defined by

$$l_i(x) = \prod_{j \neq i}^{N_p} \frac{x - x_j}{x_i - x_j}.$$

Figure 4.2 shows six one-dimensional Lagrange polynomials defined on a set of Legendre-Gauss-Lobatto (LGL) nodes.

The $d$-dimensional version of the Gauss-Lobatto points on a rectangular domain

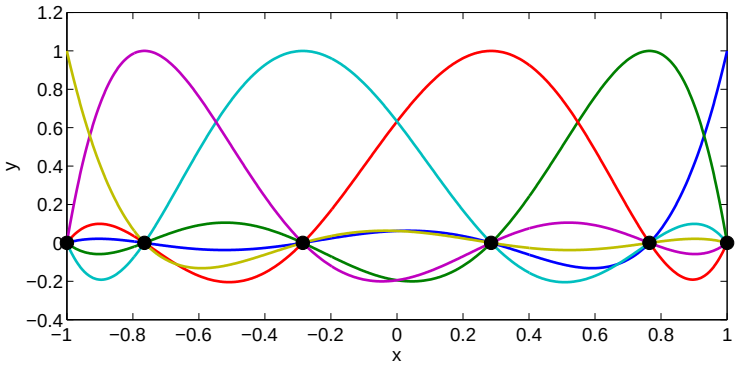



Figure 4.2: Six Lagrange polynomials defined on a set of LGL nodes

are defined by the one-dimensional nodes in each direction. This is not possible on a triangle and so more complicated constructions involving warping function must be employed [26]. The Lagrange polynomials in multiple dimensions have no known simple closed form for an arbitrary set of points, but for tensor-product grids, are defined by taking the product of one-dimensional Lagrange polynomials defined along each dimension. These are shown in figure 4.3 for a rectangular domain.

In pseudospectral methods the nodes are typically chosen to be the zeros of a family of Jacobi polynomials such as the Legendre or Chebyshev polynomials. Associated with these nodes is a set of Gauss-Lobatto integration weights, $\mathbf{w} = \{w_1, .., w_n\}$, and a pseudospectral differentiation matrix, $D$, defined by $D_{ij} = l'_j(x_i)$. The details of the construction of which we have already seen. Where the pseudospectral methods and nodal Galerkin methods differ is that the pseudospectral method deals with the strong form of a differential equation while the nodal Galerkin method deals with the integral form. Both methods however proceed by replacing functions with vectors

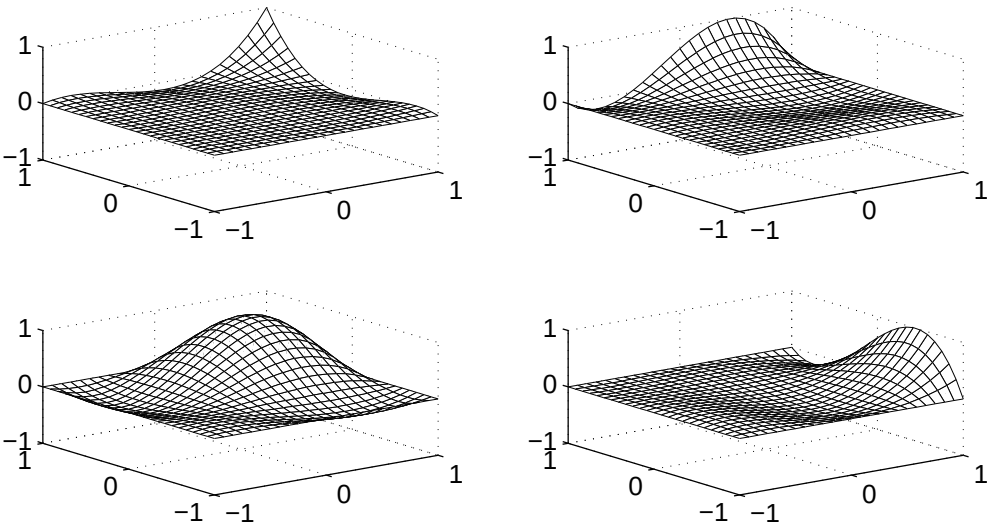


Figure 4.3: Four 2D Lagrange polynomials defined on a set of tensor-product LGL nodes

of nodal values, and differential operators with matrices. Where they differ is in the nodal Galerkin methods ability to implicitly enforce boundary conditions by modifying the surface integrals that arise from applying Green's theorem to the spatial part of the integral form of the equation.

For example, consider the 1D equation for waves propagating in either direction with unit speed

$$\begin{cases} u_{tt} = u_{xx}, & x \in (-1,1),\ t > 0 \\ u(x,0) = u_0(x), & x \in (-1,1) \\ u_t(x,0) = u_1(x), & x \in (-1,1) \\ u(-1,t) = u(1,t) = 0, & t \geq 0 \end{cases}$$

where the subscript denotes differentiation with respect to the respective variable, both methods start by choosing a set of nodes, $\{x_1, ..., x_n\}$, and defining $\mathbf{u}(t) =$

$[u(x_1,t),...,u(x_n,t)]^T$. The pseudospectral method then proceeds by solving the system of ordinary differential equations

$$\begin{cases} \ddot{\mathbf{u}}(t) = D^2\mathbf{u}(t), \quad t > 0, \\ \mathbf{u}(0) = \mathbf{u}_0, \\ \dot{\mathbf{u}}(0) = \mathbf{u}_1 \end{cases}$$

where the first and last rows and columns of the matrix $D^2$ must be set to zero to enforce the boundary conditions.

The nodal Galerkin method requires a few more steps. First $u(x,t)$ is written as

$$u(x,t) = \sum_{i=1}^{n} u(x_i,t) l_i(x).$$

Then, the weak form is derived by integrating by parts

$$\begin{aligned} \sum_{i=1}^{n} \ddot{u}(x_i,t) \int_{-1}^{1} l_i(x) l_j(x) dx &= \sum_{i=1}^{n} u(x_i,t) \int_{-1}^{1} l_i''(x) l_j(x) dx \\ &= \sum_{i=1}^{n} u(x_i,t) \left\{ l_i'(x) l_j(x) \big|_{-1}^{1} - \int_{-1}^{1} l_i'(x) l_j'(x) dx \right\}. \end{aligned}$$

For the left hand side

$$\begin{aligned} & \textstyle\sum_{i=1}^{n} \ddot{u}(x_i,t) \int_{-1}^{1} l_i(x) l_j(x) dx \\ = \; & \textstyle\sum_{i=1}^{n} \ddot{u}(x_i,t) \sum_{k=1}^{n} l_i(x_k) l_j(x_k) w_k && \leftarrow \text{LGL quadrature} \\ = \; & \textstyle\sum_{i=1}^{n} \ddot{u}(x_i,t) \sum_{k=1}^{n} \delta_{ik} \delta_{jk} w_k && \leftarrow l_i(x_k) = \delta_{ik} \\ = \; & \textstyle\sum_{i=1}^{n} \ddot{u}(x_i,t) \delta_{ij} w_i \\ = \; & \ddot{u}(x_j,t) w_j. \end{aligned}$$

For the right hand side,

$$\begin{aligned} & \textstyle\sum_{i=1}^{n} u(x_i,t) \left\{ l_i'(x) l_j(x) \big|_{-1}^{1} - \int_{-1}^{1} l_i'(x) l_j'(x) dx \right\} \\ = - \; & \textstyle\sum_{i=2}^{n-1} u(x_i,t) \left\{ \int_{-1}^{1} l_i'(x) l_j'(x) dx \right\} && \leftarrow \text{Boundary conditions} \\ = - \; & \textstyle\sum_{i=2}^{n-1} u(x_i,t) \sum_{k=2}^{n-1} l_i'(x_k) l_j'(x_k) w_k && \leftarrow \text{LGL quadrature} \\ = - \; & \textstyle\sum_{i=2}^{n-1} u(x_i,t) \sum_{k=2}^{n-1} D_{ki} D_{kj} w_k && \leftarrow l_n'(x_m) = D_{mn} \end{aligned}$$

The boundary conditions are enforced by setting $u(x_1, t) = u(x_n, t) = 0$. Again, this may be done by setting the first and last rows and columns of the differentiation matrix $D$ to zero. The problem then becomes solving the system of ordinary differential equations

$$\begin{cases} M\ddot{\mathbf{u}} + K\mathbf{u} = 0, \quad t > 0, \\ \mathbf{u}(0) = \mathbf{u}_0, \\ \dot{\mathbf{u}}(0) = \mathbf{u}_1 \end{cases}$$

where $M = diag(\mathbf{w})$ is the *mass matrix* and $K = D^T M$ is the *stiffness matrix*. Notice that if the boundary conditions had not been enforced explicitly, the matrix $K$ would have been defined as $\hat{D} - D^T M$ where

$$\hat{D} = \begin{bmatrix} D_{11} & \cdots & D_{1n} \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \\ D_{n1} & \cdots & D_{nn} \end{bmatrix}$$

Multiplying by $M^{-1}$ gives exactly the matrix $M^{-1}(\hat{D} - D^T M) = D^2$, so the nodal Galerkin and pseudospectral methods are essentially equivalent for certain types of problems. The difference between the two methods is the flexibility of the nodal Galerkin method to deal with more complex types of boundary conditions by making suitable substitutions into the weak form. For example, if the boundary terms $l_i'(x) l_j(x)|_{-1}^{1}$ were set to zero and the matrix $K$ left as $\hat{D} - D^T M$, the conditions enforced would be $u'(x_1, t) = u'(x_n, t) = 0$ which corresponds to a stress-free boundary condition, useful in modelling free-surfaces. This could have been done in the pseudospectral method by suitably altering the matrix $D^2$, but would have been a

more complicated procedure. The gap in ease of implementation is exacerbated in multiple dimensions.

To define a spectral element method (SEM) in one-dimension, an arbitrary interval $[a, b]$ is split up into $N$ smaller subintervals (cells) with endpoints $\{x_0, x_1, ..., x_N\}$. Each of these cells is mapped to the reference domain $[-1, 1]$ by the function

$$\xi : [x_i, x_{i+1}] \to [-1, 1] : x \mapsto \left( \frac{2}{x_{i+1} - x_i} \right) x - 1$$

and back by

$$x : [-1, 1] \to [x_i, x_{i+1}] : \xi \mapsto \left( \frac{x_{i+1} - x_i}{2} \right) \xi + \frac{x_i + x_{i+1}}{2}.$$

The spectral element method basis functions are defined by enforcing $C^0$ continuity across the cell interfaces. This is done by taking the Lagrange polynomials in neighbouring cells that are non-zero at the same node on the cell interface and piecing them together to form a single piecewise-continuous polynomial. Interior to each cell the basis functions are unaltered.

In practice this means that the contributions to the global problem from each cell can be computed individually and then summed at the positions corresponding to the interface nodes. This is done by defining a *connectivity matrix* $C$ with columns defined as the global nodes that appear in each element. For example, in the case of 2 cells with 3 nodes in each cell, for a total of 5 global nodes as seen in figure 4.6, the connectivity matrix would be

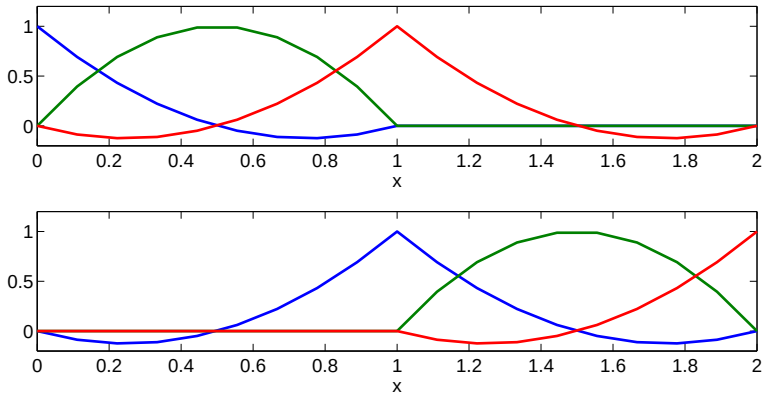



Figure 4.4: 1D SEM basis functions defined on 2 cells

$$C = \begin{bmatrix} 1 & 3 \\ 2 & 4 \\ 3 & 5 \end{bmatrix}$$

This matrix can then be used to define the global mass and stiffness matrices $M$ and $K$ from the element mass and stiffness matrices $M^i$ and $K^i$ as

**for** $i$ **from** 1 **to** # *of elements*

**for** $j$ **from** 1 **to** # *of nodes*

$$M_{C_{ji}C_{ji}} = M_{C_{ji}C_{ji}} + M^i_{jj}$$

**for** $k$ **from** 1 **to** # *of nodes*

$$K_{C_{ki},C_{ji}} = K_{C_{ki}C_{ji}} + K^i_{kj}$$

**end**

**end**

**end**

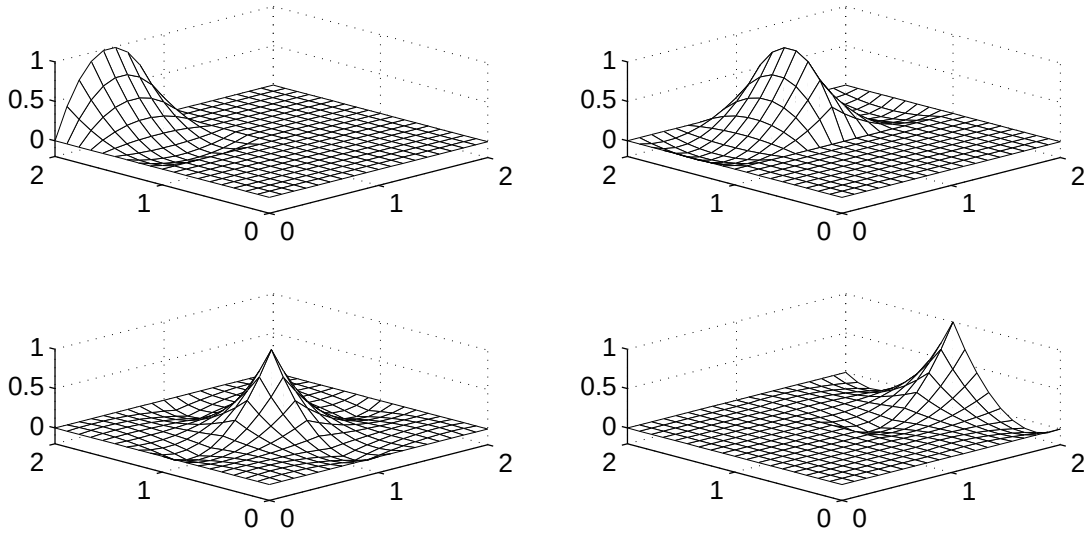



Figure 4.5: 2D SEM basis functions defined on 4 cells

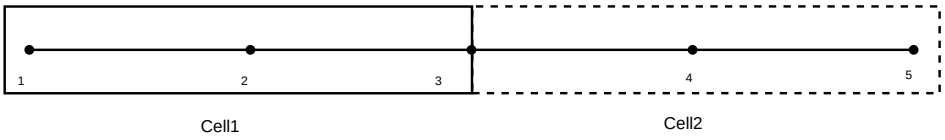



Figure 4.6: 2 cells each with 3 nodes

When computing the element mass and stiffness matrices, the basis functions on each cell are then the Lagrange polynomials mapped from the reference domain. Thus,

$$l_j'(x_i) = \frac{1}{\mathcal{J}} l_j'(\xi)$$

and

$$\int_{x_i}^{x_{i+1}} l_i(x) l_j(x) dx = \int_{-1}^{1} l_i(\xi) l_j(\xi) \mathcal{J} d\xi$$

where $\mathcal{J} = \frac{dx}{d\xi} = (x_{i+1} - x_i)/(2)$ is the Jacobian of the map from $[x_i, x_{i+1}]$ to $[-1, 1]$.

The integration weights and differentiation matrix on $[x_i, x_{i+1}]$ are then $\mathcal{J}\mathbf{w}$ and $\frac{1}{\mathcal{J}}D$. In multiple spatial dimensions the weights are, again, tensor-products of the one-dimensional versions. The two-dimensional differentiation matrices are defined by taking left or right tensor-products of the differentiation matrix defined along one dimension with the identity matrix of size equal to the number of points in the opposite dimension. For example, in two dimensions $(x, z) \in \mathbb{R}^2$, if $D$ is the 1D differentiation matrix along the $x$ direction, the 2D version in the same direction is $D \otimes I$. Since the 1D matrix $D$ is fully populated, the number of non-zero entries in the 2D version will be $1/Np$, where $Np$ is the number of points in the opposite direction that $D$ is defined along.

# Chapter 5

# Acoustic Waves

## 5.1 The Acoustic Wave Equation

We now have the tools necessary to start modelling more complicated wave equations. The first type we will derive the method for is the acoustic wave equation for the propagation of pressure (P) waves.

The full acoustic wave equation for pressure is written as

$$\ddot{u}(\mathbf{x},t) = K(\mathbf{x})\nabla \cdot \left(\frac{1}{\rho(\mathbf{x})}\nabla u(\mathbf{x},t)\right) + f(\mathbf{x},t), \quad \mathbf{x} \in \Omega, t > 0, \tag{5.1}$$

where $K$ is the bulk modulus, $\rho$ is the density of the media and $f$ is the forcing term. $\Omega$ is the domain, which is assumed to be discontinuous globally, but made up of smaller continuous sections where waves will have constant parameters [9]. $x \in \mathbb{R}^d$, $d = 1, .., 3$ is the spatial dimension, dot denotes the time derivative and $\nabla$ is the gradient operator. This simulates a pure pressure wave propagating through the region $\Omega$ and has only a single component $u$ representing the pressure in the medium at a point $\mathbf{x}$ at time $t$. While this is less realistic in terms of modelling seismic waves than the elastic wave equation, it is still very useful since the bulk of recorded data is due to the pressure component of the wave. Because of this, many iterative imaging algorithms employ the acoustic wave equations since it is much less computationally

expensive to model, yet still exhibits useful responses.

## 5.2 Derivation of the Weak Problem

We begin by choosing a suitable space $V$ of trial and test functions, and applying the Galerkin method to (5.1). This yields

$$\int_\Omega \ddot{u} v d\Omega = \int_\Omega K \nabla \cdot \left( \frac{1}{\rho} \nabla u \right) v d\Omega + \int_\Omega f v d\Omega, \quad \forall v \in V$$

or, after applying Green's theorem with $\Gamma = \partial\Omega$

$$\int_\Omega \ddot{u} v d\Omega + \int_\Omega \frac{1}{\rho} \nabla u \cdot \nabla (Kv) d\Omega = \oint_\Gamma \frac{K}{\rho} \nabla u \cdot \mathbf{n} v d\Gamma + \int_\Omega f v d\Omega,$$

where $\mathbf{n}$ is the unit outward pointing normal vector.

The region $\Omega$ is then split into a union of smaller subregions $\Omega^k$ in such a way that in each of the subregions the parameters $K$ and $\rho$ are constant and will be denoted by $K_k$, and $\rho_k$. The speed at which the waves propagate in each subregion is $Vp_k = \sqrt{K_k/\rho_k}$, so the last equation may be written as

$$\sum_{k=1}^{N_{cells}} \left\{ \int_{\Omega^k} \ddot{u} v d\Omega^k + V p_k^2 \int_{\Omega^k} \nabla u \cdot \nabla v d\Omega^k \right\} = \int_\Omega f v d\Omega + \sum_{k=1}^{N_{edges}} V p_k^2 \oint_{\Gamma^k} \nabla u \cdot \mathbf{n} v d\Gamma^k. \tag{5.2}$$

The boundary integral has been split into $N_{edges}$ subregions corresponding to the boundary of the $\Omega_k$ that lie along the outer boundary, $\Gamma$, of the entire region. Rigorously, this means that the space of trial and test functions defined along the boundary does not belong to the same space of functions as those defined on the entire domain.

This, however happens automatically in practice as the space of functions chosen to define the problem on the interior is made up of products of functions defined along the boundary.

The points chosen here are the Legendre-Gauss-Lobatto (LGL) points mapped to the subdomain $\Omega^k$. Globally, each of these subdomains share edge nodes and so the points are chosen constructively instead of explicitly. For simplicity, the cells chosen here are all the same size with the same number of nodes in each direction. More complicated procedures are possible for construction of the computational domain via more elaborate mappings from the reference domain to the global subdomains, but generally require substantial user input and, thus, are not substantive when employed in iterative imaging algorithms.

Further derivation in limited to the 2D case with $x$ being the horizontal variable and $z$ the vertical variable taken positive downward. The choice of nodes is denoted $\mathbf{x}$. The weights in 2D, $\mathbf{w}$. The integration weights along the boundary will be denoted $\mathbf{w}^N$, $\mathbf{w}^S$, $\mathbf{w}^E$, $\mathbf{w}^W$ corresponding to the north (up), south, east and west boundaries, respectively. The differentiation matrices are denoted $D^x$ and $D^z$. Because the 2D grid is constructed is a simplified manner the Jacobian on the $k^{th}$ cell, $\mathcal{J}^k$, in two dimensions will be the product of the one dimensional Jacobians $\mathcal{J}_x^k$, $\mathcal{J}_z^k$.

Proceeding as in the previous chapter, the solution at some time $t > 0$ is assumed to be superpositions of previous times and so is written as a linear combination of

tensor-product Lagrange polynomials defined by the nodes $\mathbf{x}$ where the coefficients are the values at a single point at some time $t$. That is,

$$u(x,z,t) = \sum_{i=1}^{N_p^x} \sum_{j=1}^{N_p^z} u(x_i, z_j, t) l_i(x) l_j(z) := \sum_{k=1}^{N_p} u(\mathbf{x}_k, t) l_k(\mathbf{x})$$

where the assignment has been made in an attempt to simplify future notation. The notation $l_k(\mathbf{x})$ is slightly vague as the function is technically the product of two functions each with its own index, but it will make things much less cluttered. $N_p^x$ and $N_p^z$ denote the number of horizontal, vertical nodes, respectively, while $N_p$ is the total number of nodes.

Equation (5.2) must then be transformed into an $N_p$-dimension system of ordinary differential equations for the nodal values. As before, the test functions $v$ are chosen to be the same Lagrange polynomials used to define $u$. On each element the mass matrix is then calculated by

$$\begin{aligned}
\int_{\Omega^k} \ddot{u} v d\Omega^k &= \sum_{i=1}^{N_p} \ddot{u}(\mathbf{x}_i, t) \int_{\Omega^k} l_i(\mathbf{x}) l_j(\mathbf{x}) d\Omega^k \\
&= \sum_{i=1}^{N_p} \ddot{u}(\mathbf{x}_i, t) \sum_{m=1}^{N_p} l_i(\mathbf{x}_m) l_j(\mathbf{x}_m) \mathbf{w}_m \mathcal{J}^k \\
&= \sum_{i=1}^{N_p} \ddot{u}(\mathbf{x}_i, t) \sum_{m=1}^{N_p} \delta_{im} \delta_{jm} \mathbf{w}_m \mathcal{J}^k \\
&= \sum_{i=1}^{N_p} \ddot{u}(\mathbf{x}_i, t) \delta_{ij} \mathbf{w}_i \mathcal{J}^k \\
&= \ddot{u}(\mathbf{x}_j, t) \mathbf{w}_j \mathcal{J}^k
\end{aligned}$$

Since equation (5.2) is summed over all cells, at a cell interface node $\mathbf{x}_i$, the result is

$$\sum_{k=1}^{N_{cells}} \ddot{u}(\mathbf{x}_i, t)\mathbf{w}_i\mathcal{J}^k$$

So, for the $k^{th}$ element, $M^k = \mathcal{J}^k diag(\mathbf{w})$ and the global mass matrix is obtained by summing over the connected elements.

The stiffness matrix is defined by

$$\begin{aligned}
(V_p^k)^2 \int_{\Omega^k} \nabla u \cdot \nabla v d\Omega^k &= (V_p^k)^2 \sum_{i=1}^{N_p} u(\mathbf{x}_i, t) \int_{\Omega^k} \left\{ \frac{\partial l_i(\mathbf{x})}{\partial_x}\frac{\partial l_j(\mathbf{x})}{\partial_x} + \frac{\partial l_i(\mathbf{x})}{\partial_z}\frac{\partial l_j(\mathbf{x})}{\partial_z} \right\} d\Omega^k \\
&= (V_p^k)^2 \sum_{i=1}^{N_p} u(\mathbf{x}_i, t) \sum_{m=1}^{N_p} \left\{ \frac{\partial l_i(\mathbf{x}_m)}{\partial_x}\frac{\partial l_j(\mathbf{x}_m)}{\partial_x} + \frac{\partial l_i(\mathbf{x}_m)}{\partial_z}\frac{\partial l_j(\mathbf{x}_m)}{\partial_z} \right\} \mathbf{w}_m\mathcal{J}^k \\
&= (V_p^k)^2 \sum_{i=1}^{N_p} u(\mathbf{x}_i, t) \sum_{m=1}^{N_p} \left\{ \frac{1}{(\mathcal{J}_x^k)^2} D_{mi}^x D_{mj}^x + \frac{1}{(\mathcal{J}_z^k)^2} D_{mi}^z D_{mj}^z \right\} \mathbf{w}_m\mathcal{J}^k
\end{aligned}$$

So, for the $k^{th}$ element,

$$K^k = \frac{\mathcal{J}_z^k}{\mathcal{J}_x^k}(D^x)^T diag(\mathbf{w}) D^x + \frac{\mathcal{J}_x^k}{\mathcal{J}_z^k}(D^z)^T diag(\mathbf{w}) D^z,$$

again, the entries that act on nodes shared between elements are summed.

For the forcing term, $f$ will be assumed to be of the form $f_1(t)f_2(\mathbf{x})$. Then

$$\begin{aligned}
\int_\Omega f v d\Omega = f_1(t) \int_\Omega f_2(\mathbf{x}) l_j(\mathbf{x}) d\Omega & \\
&= f_1(t) \sum_{i=1}^{N_p} f_2(\mathbf{x}_i) l_j(\mathbf{x}_i) \mathbf{w}_i \mathcal{J}^k \\
&= f_1(t) \sum_{i=1}^{N_p} f_2(\mathbf{x}_i) \delta_{ij} \mathbf{w}_i \mathcal{J}^k \\
&= f_1(t) f_2(\mathbf{x}_j) \mathbf{w}_j \mathcal{J}^k.
\end{aligned}$$

So the time-dependent vector of nodal forces is $M\mathbf{F}(t) = M[f_2(\mathbf{x}_1), ..., f_2(\mathbf{x}_{N_p})]^T f_1(t)$, where $M$ is the global mass matrix.

## 5.3 Absorbing Boundaries

In order to make the boundaries simulate an infinite medium, we aim to implement a type of boundary condition that will allow most of the energy of the waves travelling in the modelled region to escape when reaching the edge, without causing too much of a reflection. Two methods are considered here. The first is to add a damping term to the problem in a portion of the region near the boundary. This is called *Rayleigh damping*. For now, assume that $\nabla u \cdot \mathbf{n} = 0$ so that the boundary term

$$\sum_{k=1}^{N_{edges}} V p_k^2 \oint_{\Gamma^k} \nabla u \cdot \mathbf{n} v d\Gamma^k$$

disappears and the problem is now to numerically solve the system

$$M\ddot{\mathbf{u}}(t) + K\mathbf{u}(t) = M\mathbf{F}(t)$$

where $\mathbf{u}(t)$ is the vector of nodal displacements at some time $t$.

The damped problem is

$$M\ddot{\mathbf{u}}(t) + A\dot{\mathbf{u}}(t) + K\mathbf{u}(t) = M\mathbf{F}(t). \tag{5.3}$$

Denote the damping region by $\Omega_d$. The matrix in the damping term is defined in terms of the mass and stiffness matrices as

$$\begin{cases} A_{ij} = \alpha M_{ij} + \beta K_{ij}, & \mathbf{x}_i \; and \; \mathbf{x}_j \in \Omega_d, \\ A_{ij} = 0, & elsewhere \end{cases}$$

with $\alpha + \beta\omega_i^2 = 2\omega_i\zeta$, where the $\omega_i$ are the characteristic frequencies of the system and $\zeta$ is the damping parameter. To define $\alpha$ and $\beta$ two frequencies and a damping

parameter need to be chosen and then the resulting system of equations solved. [20] showed that choosing $\zeta = 0.3$ and $\omega_1, \omega_2$ to be the peak and upper half power frequencies of the source wavelet $f_1(t)$ proved to be sufficient in damping enough of the wave to be useful in seismic experiments.

The second method considered here is to introduce boundary conditions that simulate one-way wave equations along the region $\Gamma$ [6]. In one spatial dimension these are exact as all waves are either travelling left or right. In more than 1 spatial dimension waves can travel an infinite number of directions and so paraxial approximations must be made to account for this.

For a wave travelling at speed $c^2$ the general form of the absorbing boundary condition at the west boundary ($x = 0$) is

$$\left\{ \prod_{i=1}^{N} \cos(\theta_i) \frac{\partial}{\partial_t} - c \frac{\partial}{\partial_x} \right\} u(\mathbf{x}, t) = 0,$$

where $|\theta_i| < \pi/2$ for all $i$ [14]. First-order conditions can be generated by considering only $\theta = 0$ producing $\dot{u} + c\nabla u \cdot \mathbf{n} = 0$. In 1D this produces exactly the equations for left or right travelling waves $\dot{u} + cu_x = 0$ and $\dot{u} = cu_x$. In 2D the conditions are exact for wavefronts arriving parallel to the boundary and are increasingly reflected as $\theta \to \pm\pi/2$. Increasing the number of terms produces combined space/time differential operators that are much more difficult to implement.

Substituting the first order conditions into the boundary term for an element $k$

produces

$$\begin{aligned} Vp_k^2 \oint_{\Gamma^k} \nabla u \cdot \mathbf{n} v d\Gamma^k &= -Vp_k \oint_{\Gamma^k} \dot{u} v d\Gamma^k \\ &= -Vp_k \sum_{i=1}^{N_p} \dot{u}(\mathbf{x}_i, t) \oint_{\Gamma^k} l_i(\mathbf{x}) l_j(\mathbf{x}) d\Gamma^k \\ &= -Vp_k \sum_{i=1}^{N_p} \dot{u}(\mathbf{x}_i, t) \sum m = 1^{N_p} l_i(\mathbf{x}_m) l_j(\mathbf{x}_m) \mathbf{w}_m|^k \\ &= -Vp_k \sum_{i=1}^{N_p} \dot{u}(\mathbf{x}_i, t) \delta_{ij} \mathbf{w}_i \mathcal{J}^k \\ &= -Vp_k \dot{u}(\mathbf{x}_j, t) \hat{\mathbf{w}}_j \hat{\mathcal{J}}^k \end{aligned}$$

where $\hat{\mathbf{w}}$ and $\hat{\mathcal{J}}$ depend on if the integration is being performed over the $N, S, E$ or $W$ boundary. This again results in a damped system of ordinary differential equations (5.3). $A$ is now, however, much sparser than in the case of Rayleigh damping.

## 5.4 Time Stepping and Stability

To deal with the time-dependent system a numerical procedure must be implemented that is capable of handling the first and second order derivatives in equation (5.3). A second order in time scheme can be constructed by replacing the derivatives with second-order central difference approximations

$$\ddot{\mathbf{u}}(t_j) = \frac{\mathbf{u}(t_{j+1}) - 2\mathbf{u}(t_j) + \mathbf{u}(t_{j-1})}{\Delta t^2} + \mathcal{O}(\Delta t^2), \quad \dot{\mathbf{u}}(t_j) = \frac{\mathbf{u}(t_{j+1}) - \mathbf{u}(t_{j-1})}{2\Delta t} + \mathcal{O}(\Delta t^2).$$

After dropping the error term, (5.3) then becomes

$$M\left(\frac{\mathbf{u}(t_{j+1}) - 2\mathbf{u}(t_j) + \mathbf{u}(t_{j-1})}{\Delta t^2}\right) + A\left(\frac{\mathbf{u}(t_{j+1}) - \mathbf{u}(t_{j-1})}{2\Delta t}\right) + K\mathbf{u}(t_j) = M\mathbf{F}(t_j)$$

or in terms of the $t_j$'s

$$\left[M + \frac{\Delta t}{2}A\right]\mathbf{u}(t_{j+1}) + \left[\Delta t^2 K - 2M\right]\mathbf{u}(t_j) + \left[M - \frac{\Delta t}{2}A\right]\mathbf{u}(t_{j-1}) = \Delta t^2 M\mathbf{F}(t_j) \tag{5.4}$$

The matrix $\left[M + \frac{\Delta t}{2}A\right]$ must now be inverted in order to step forward in time. This is where the two types of boundary treatments lead to very different problems. In both cases the mass matrix is diagonal and so its inverse is trivial. In the case of absorbing boundary conditions the matrix $A$ is also diagonal and in fact much of its main diagonal is zero so $\left[M + \frac{\Delta t}{2}A\right]^{-1}$ is defined purely by it main diagonal which is just the reciprocal of the diagonal of $\left[M + \frac{\Delta t}{2}A\right]$. In the case of Rayleigh damping the matrix $A$ is not diagonal and so at each time step a large system of equations must be solved.

Another way to time-step the problem would be to re-write it as a first order system by making the substitution $\mathbf{v} = \dot{\mathbf{u}}$. Then (5.3) can be rewritten as

$$\begin{bmatrix}\dot{\mathbf{u}}\\ \dot{\mathbf{v}}\end{bmatrix} + \begin{bmatrix}0 & I\\ M^{-1}K & M^{-1}A\end{bmatrix}\begin{bmatrix}\mathbf{u}\\ \mathbf{v}\end{bmatrix} = \begin{bmatrix}0\\ \mathbf{F}\end{bmatrix} \tag{5.5}$$

and solved by an appropriate method method for first-order systems. Careful choice of the method can avoid the need to invert the matrix $A$ but at the cost of doubling the size of the system. In practice this is less ideal that it may appear as the size of the system has now doubled and may not fit into memory. Also, many numerical methods that avoid inverting $A$ result in *much* smaller time-steps than are required to solve the second-order system via central finite differences.

The time stepping method 5.4 is the same as that employed in [8], in which they state that the method must satisfy a CFL condition

$$\Delta t \leq \zeta \frac{\Delta x_{min}}{max\{Vp\}}$$

where $\zeta$ is strictly less than 1. Further, because $\Delta x_{min} \approx N^{-2}$ where $N$ is the maximum number of nodes in each direction inside the cells, this can be rewritten as

$$\Delta t \leq \frac{L}{N^2} \frac{\bar{\zeta}}{max\{Vp\}}$$

where L is the spatial dimension of the problem. They state that for the case when the number of nodes in each cell is 5-by-5 the constant $\bar{\zeta}$ required for stability is approximately $4.3\zeta$.

In testing the largest stable time-step for the second-order central finite-difference scheme was

$$\Delta t \approx \frac{1}{2} \frac{min\{\Delta x, \Delta z\}}{max\{Vp\}}.$$

For comparison, when the first-order system was advanced using a first-order-in-time Euler method the required time-step was

$$\Delta t \approx \frac{1}{125} \frac{min\{\Delta x, \Delta z\}}{max\{Vp\}}.$$

More concrete stability conditions for time-stepping systems of equations involving pseudospectral differentiation matrices are not as readily available as those for finite-differences. This is due to the non-normality of the matrices used in the construction. In a sense stability is an almost open question for methods involving

pseudospectral differentiation matrices. An extensive treatment of the topic is contained in [23] using the language of $\varepsilon - pseudospectra$ defined as the set of points. For a matrix $E$, the $\varepsilon - pseudospectra$ are defined as the set of points

$$\sigma_\epsilon(E) = \{z \in \mathbb{C} | \|(z - E)^{-1}\| > 1/\epsilon\}$$

The standard method of deriving stability conditions is that to evolve a difference equation

$$\mathbf{u}(t_{j+1}) = E\mathbf{u}(t_j)$$

the eigenvalues of $E$ must be contained within the unit circle so that powers of $E$ will remain small in norm. For spatial operators built out of pseudospectral differentiation matrices this fails in general. The problem is that small perturbations in the matrix $E$ can cause huge changes in its eigenvalues. Thus a more robust stability condition, as suggested in [23] is that the $\varepsilon - pseudospectra$ must be $well - behaved$. That is, small perturbations should not cause large changes in the eigenvalues.

To perform this analysis for equation 5.4 we rewrite is as

$$\begin{bmatrix} \mathbf{U}^{n+1} \\ \mathbf{U}^n \end{bmatrix} = \begin{bmatrix} P & Q \\ I & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U}^n \\ \mathbf{U}^{n-1} \end{bmatrix}$$

where,

$$P = \left[M + \frac{dt}{2}A\right]^{-1} \left[2M - dt^2 K\right]$$
$$Q = \left[M + \frac{dt}{2}A\right]^{-1} \left[M - \frac{dt}{2}A\right].$$

Then

$$E = \begin{bmatrix} P & Q \\ I & 0 \end{bmatrix}.$$

Figure 5.1 show the $\varepsilon - pseudospectra$ for a global method based on $\Delta t = \frac{3N^{-2}}{\max_j c(\mathbf{x}_j)}$. $Np = 10$ points, $\max_j c(\mathbf{x}_j) = 3$, while figure 5.2 shows the norm of powers of $E$ as time increases.

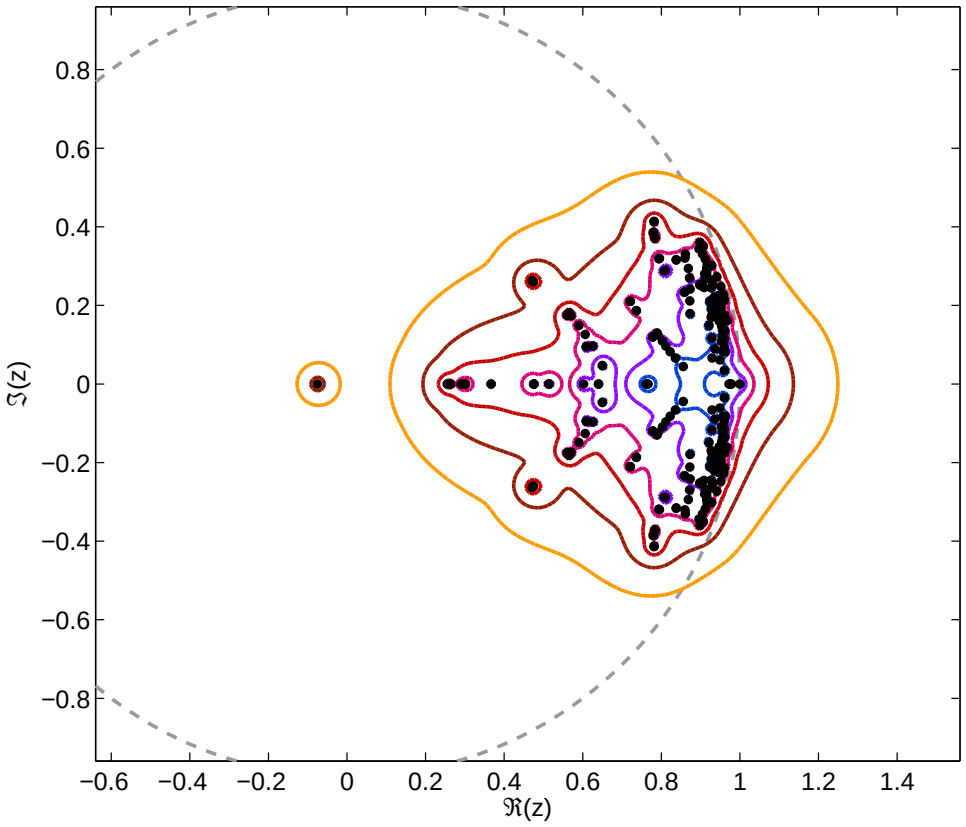



Figure 5.1: $\epsilon - pseudospectra$ for evolution operator $E$

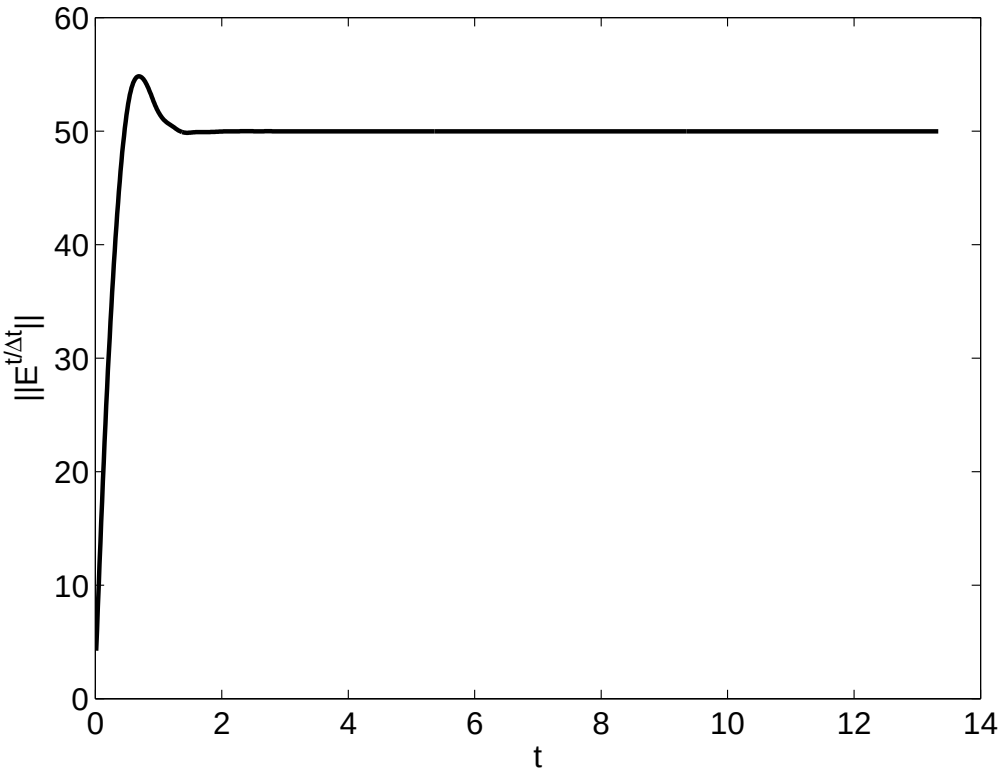



Figure 5.2: 2-norm of powers of the evolution operator $E$.

Figure 5.3 shows the $\varepsilon - pseudospectra$ for the same operator built using a domain-decomposition method with $Np = 4$ points, $N = 4$ cells, $\Delta t = \frac{min(\Delta x)}{2 \max_j c(\mathbf{x}_j)}$, $\max_j c(\mathbf{x}_j) = 2000$, while figure 5.4 shows the norm of powers of $E$. We have written the stability condition in terms of the minimum grid spacing instead of $N^{-}2$ since there could be a different number of nodes in each cell.

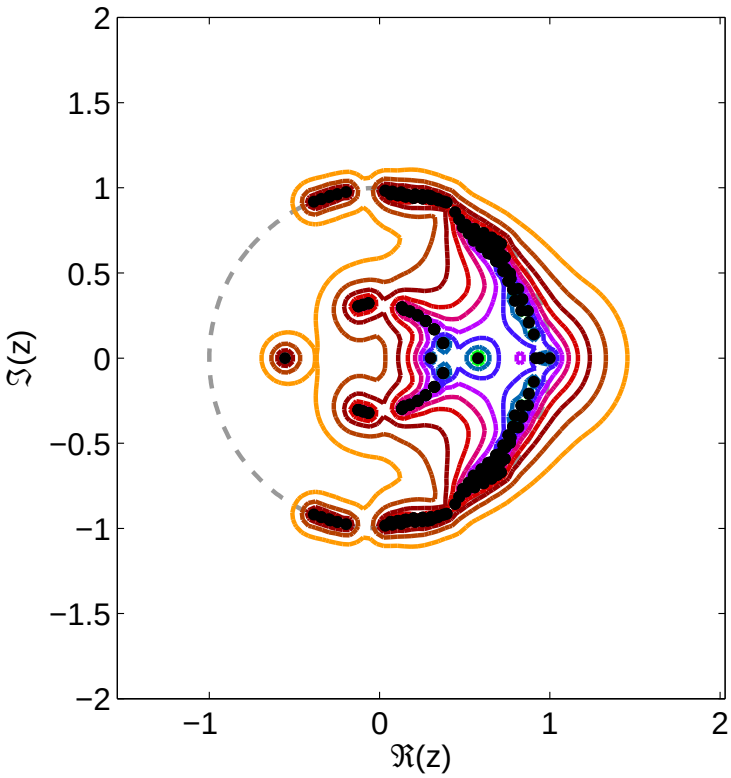



Figure 5.3: $\epsilon - pseudospectra$ for evolution operator $E$ for

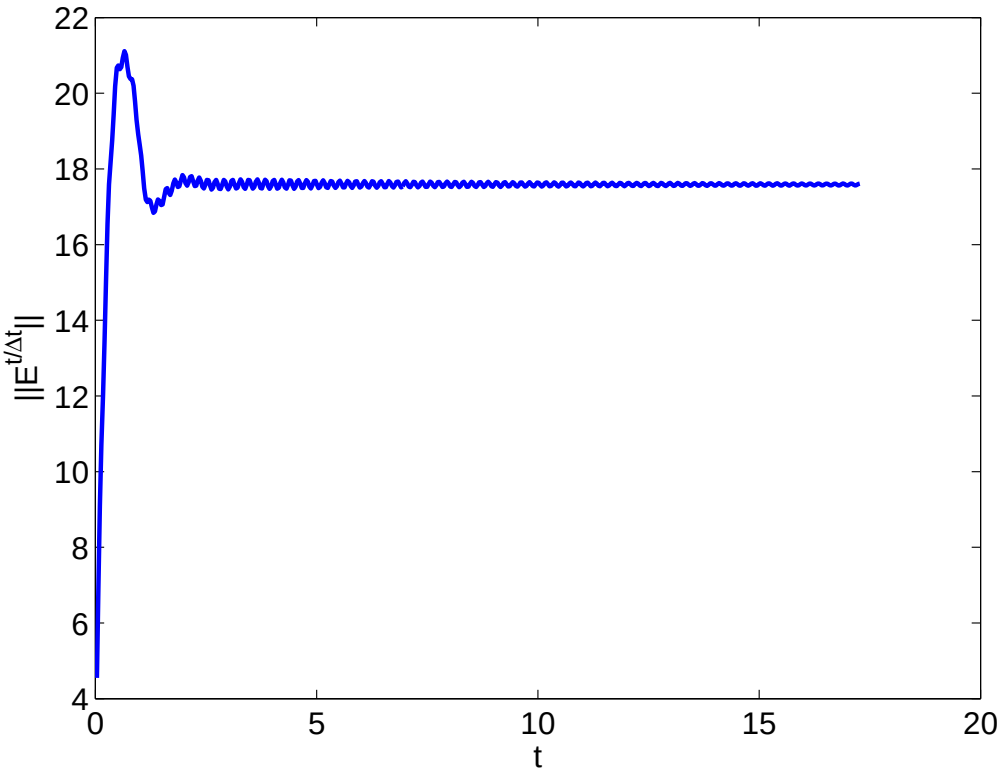



Figure 5.4: 2-norm of powers of the evolution operator $E$

It's clear that domain-decomposition has the desirable effect of spreading the eigenvalues out around the edge of the unit circle even for an increased number of total nodes. However, both the global and the domain-decomposed operators have the undesirable "bump" in the $\varepsilon-pseudospectra$ near the point $(1, 0)$. This can (and does) cause problems in practice so for long time evolution, even smaller time-step requirements may be necessary.

## 5.5 Comparison of Absorbing Boundaries

To test compare the boundary treatment a source was placed in the middle of a 2000 by 2000 meter square homogeneous medium with $Vp = 1000m/s$ and the simulation run for $1.5s$. The source had a small Gaussian of the form

$$f_2(\mathbf{x}) = e^{-||\mathbf{x}-\mathbf{x}_0||^2}$$

centered at $\mathbf{x}_0$ for a spatial component and a 20Hz Ricker wavelet, defined as the second-derivative of a gaussian and seen in figure 5.5, for a time component. The model was discretized into 300 by 300 cells each made up of 3 by 3 local nodes for a total of 601 by 601, or 361,201 global nodes.

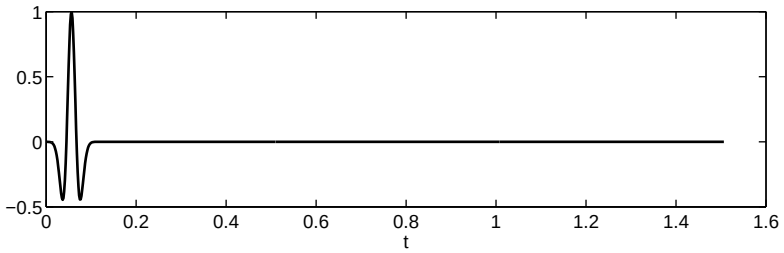



Figure 5.5: Ricker wavelet

Figures 5.6, 5.7 and 5.8 show the reflections from the boundaries for $\nabla \mathbf{u} \cdot \mathbf{n} = 0$, Rayleigh damping and first-order Higdon absorbing boundary conditions. As expected the boundary reflections from the absorbing boundary conditions are zero along horizontal and vertical lines away from the source and increase as the angle of incidence increases. The Rayleigh damping has the same reflections regardless of angle of incidence but the reflected wave has only about 10% of the energy of the

incident wave. This is promising, however the increased computational cost may not be feasible for larger models.

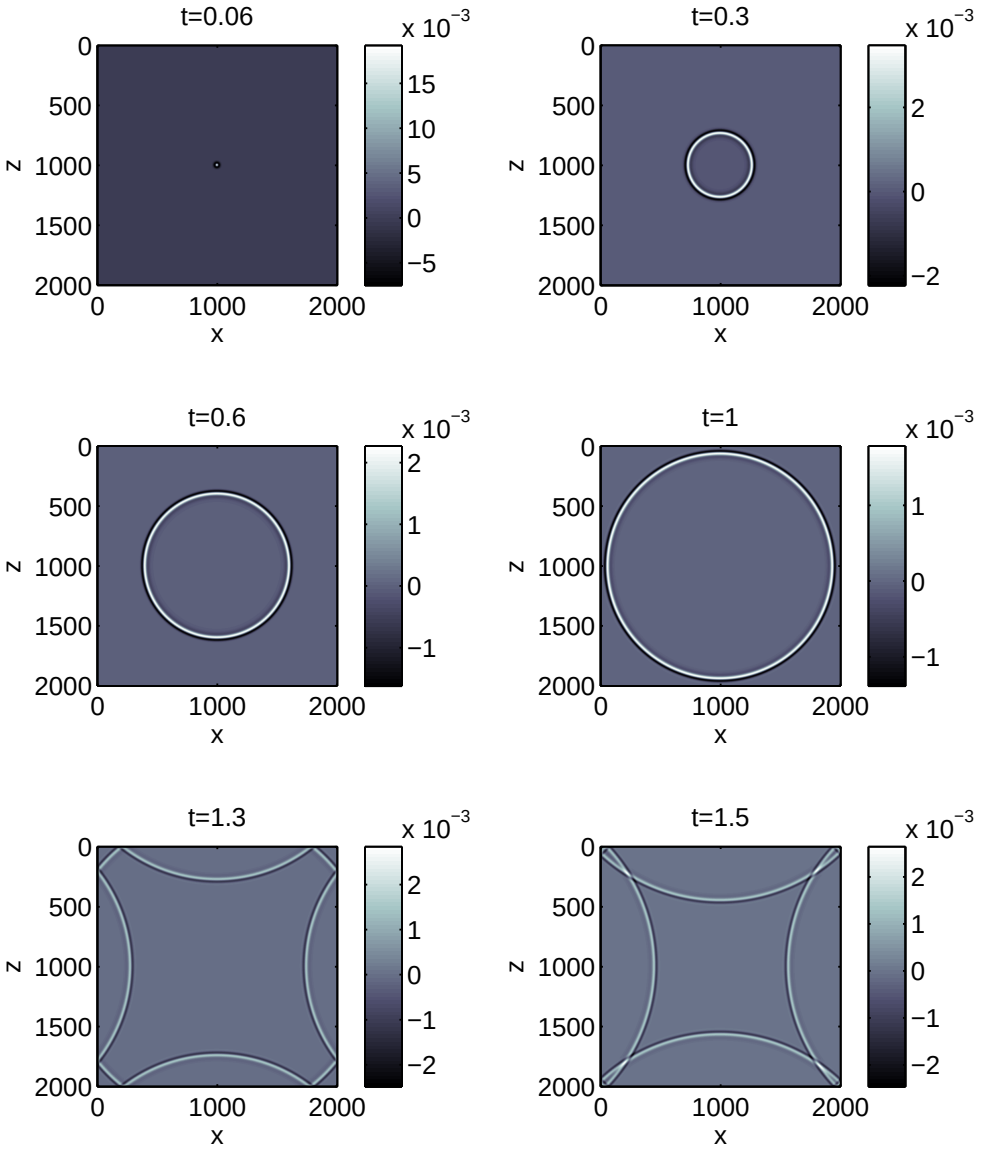



Figure 5.6: Full boundary reflection

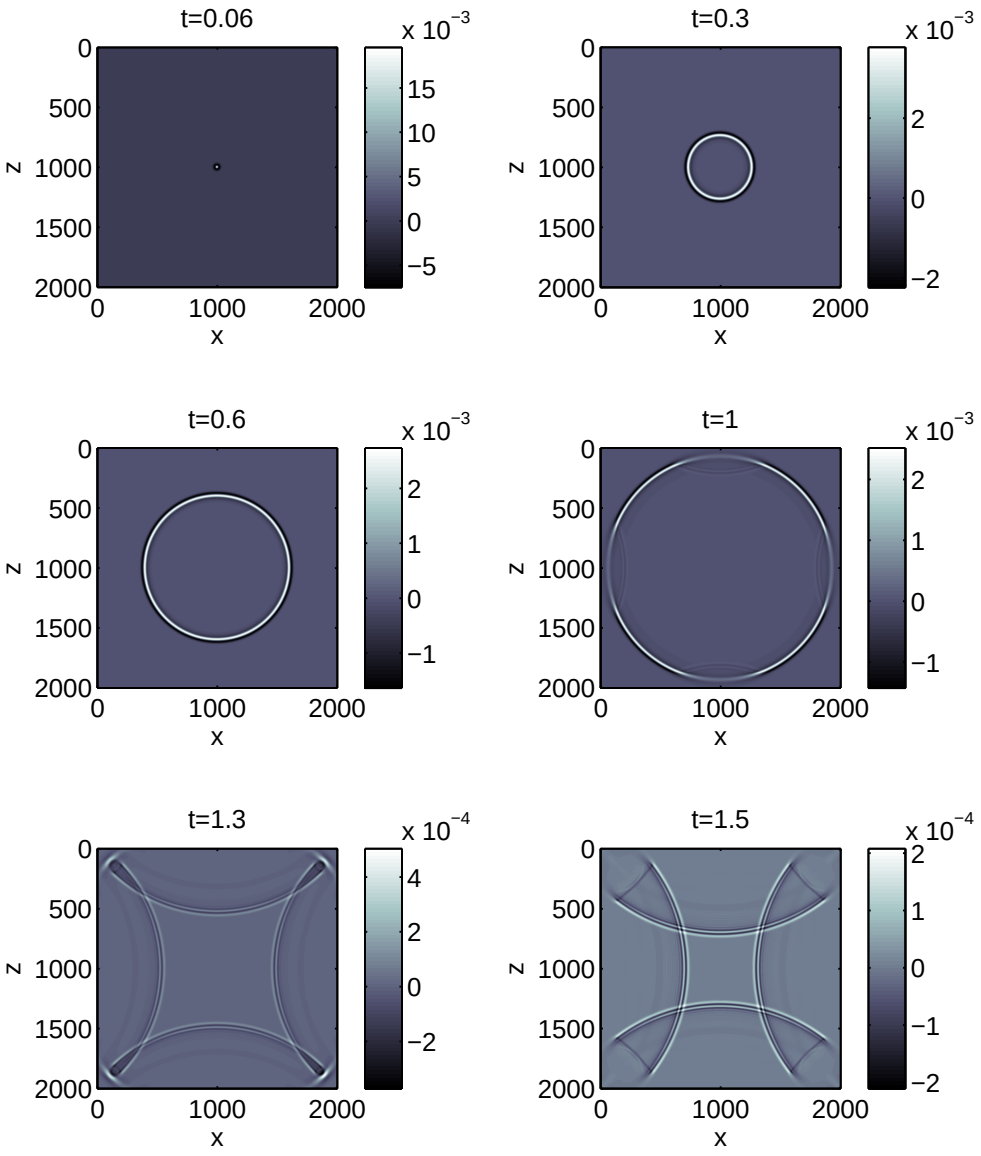



Figure 5.7: Rayleigh boundary reflection. Note the scale change.

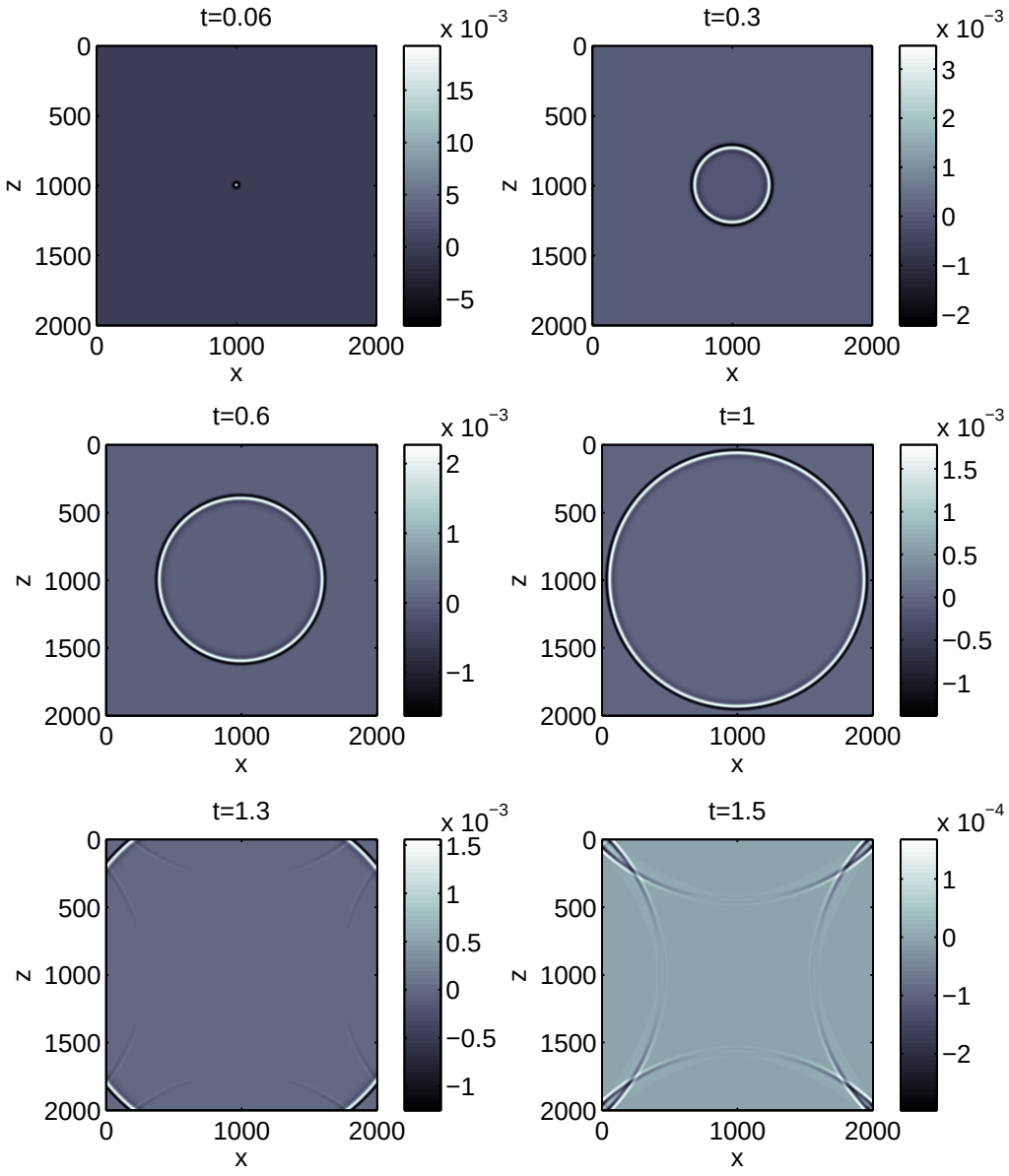



Figure 5.8: Absorbing boundary reflection. Note the scale change.

To test the method on a discontinuous velocity model a portion of the Marmousi

model was interpolated onto 400 by 400 cells as shown in figure 5.9.

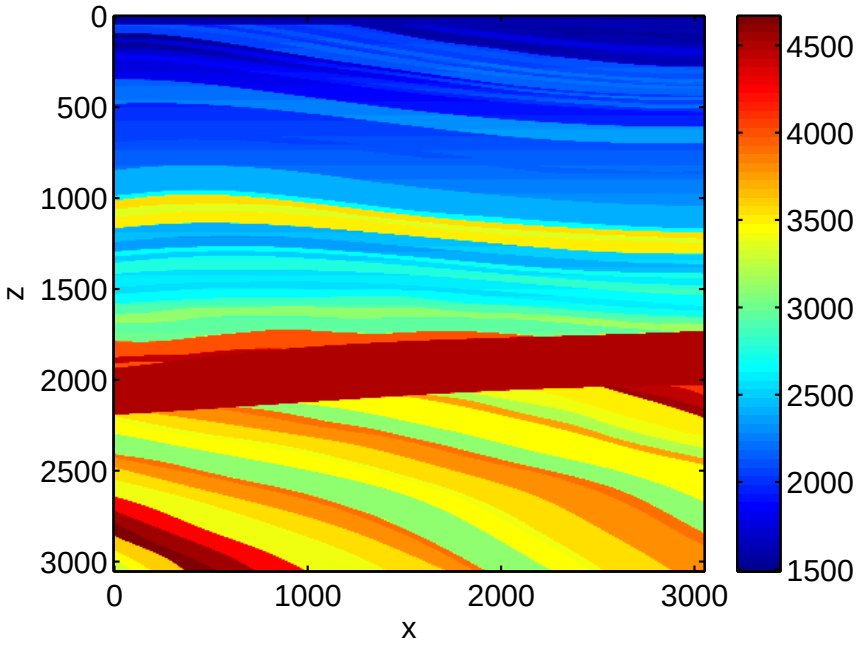



Figure 5.9: A portion of the Marmousi Velocity model interpolated onto 400 by 400 cells

Each cell was was made up of 3 by 3 nodes for a total of 641,601 global nodes. A source of the same type as above but with a 30Hz time component was placed at the surface and the simulation run for $2.5s$. A free surface boundary condition was simulated at the surface and absorbing boundary conditions were imposed on the the $S, E$ and $W$ boundaries. Figure 5.11 shows the resulting wavefield. Where the absorbing boundaries fail is when the reflected waves from the subsurface structure return to the surface near the vertical boundaries. These waves arrive at almost 90° to the vertical boundary and so fail to be absorbed as can be seen in figure 5.10.

Figure 5.12 shows the wavefield at the surface ($z = 0$) for all $t$. The image has been scaled so that the reflected waves can be seen clearly.

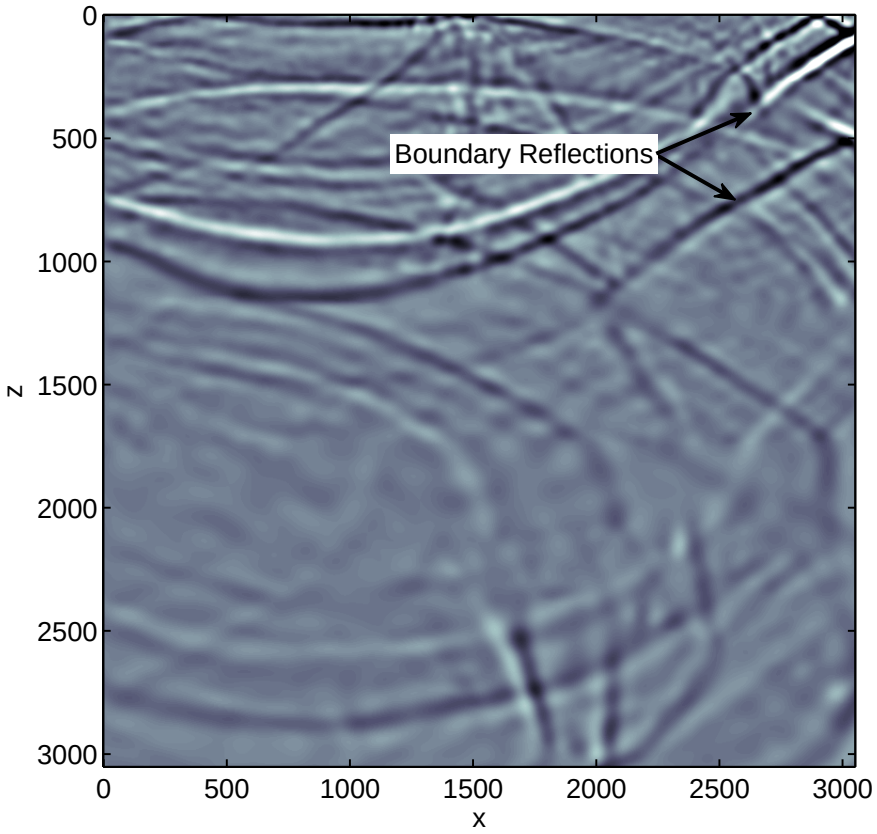



Figure 5.10: Reflections from absorbing boundary conditions

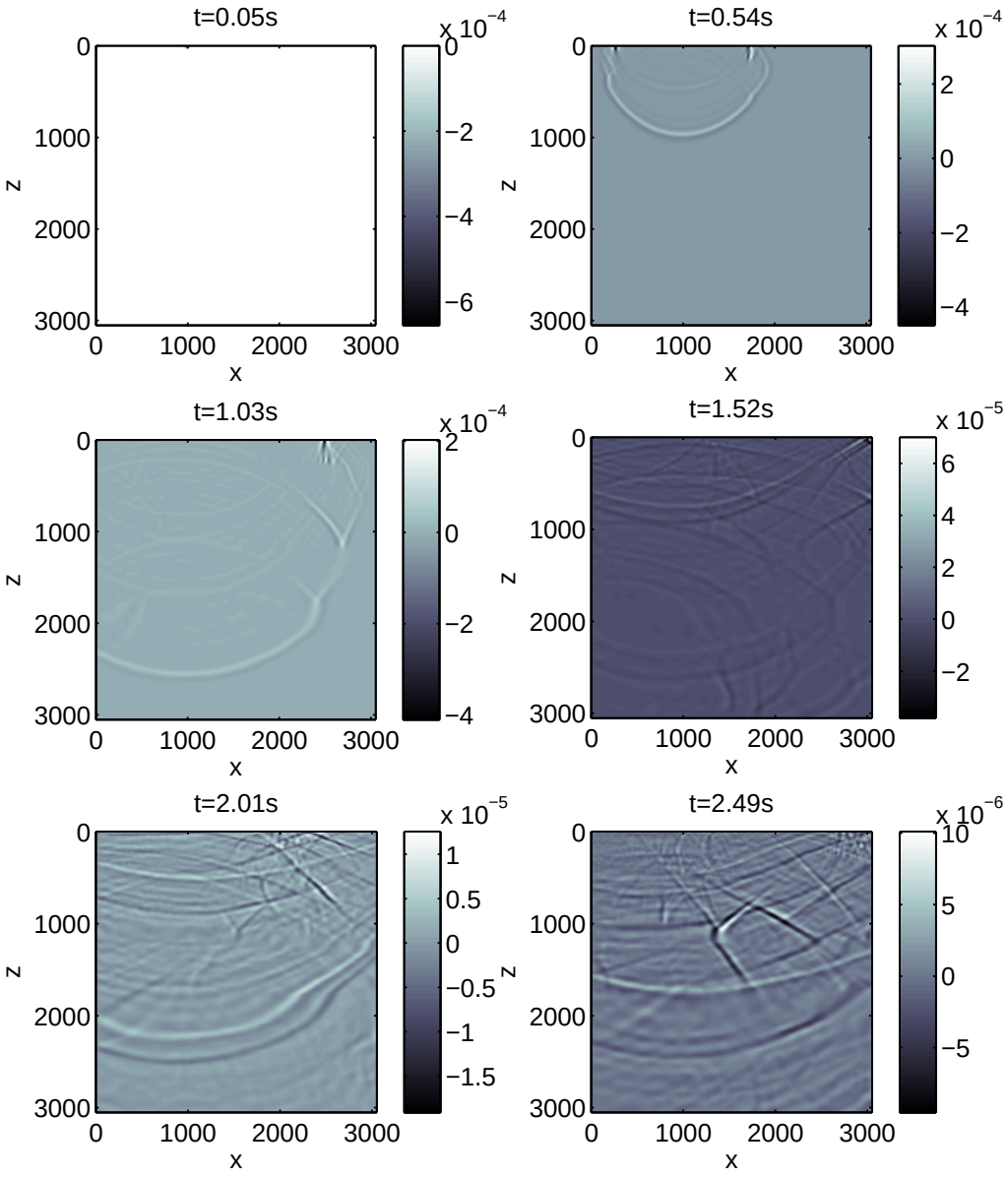



Figure 5.11: Wavefield for Marmousi velocity model

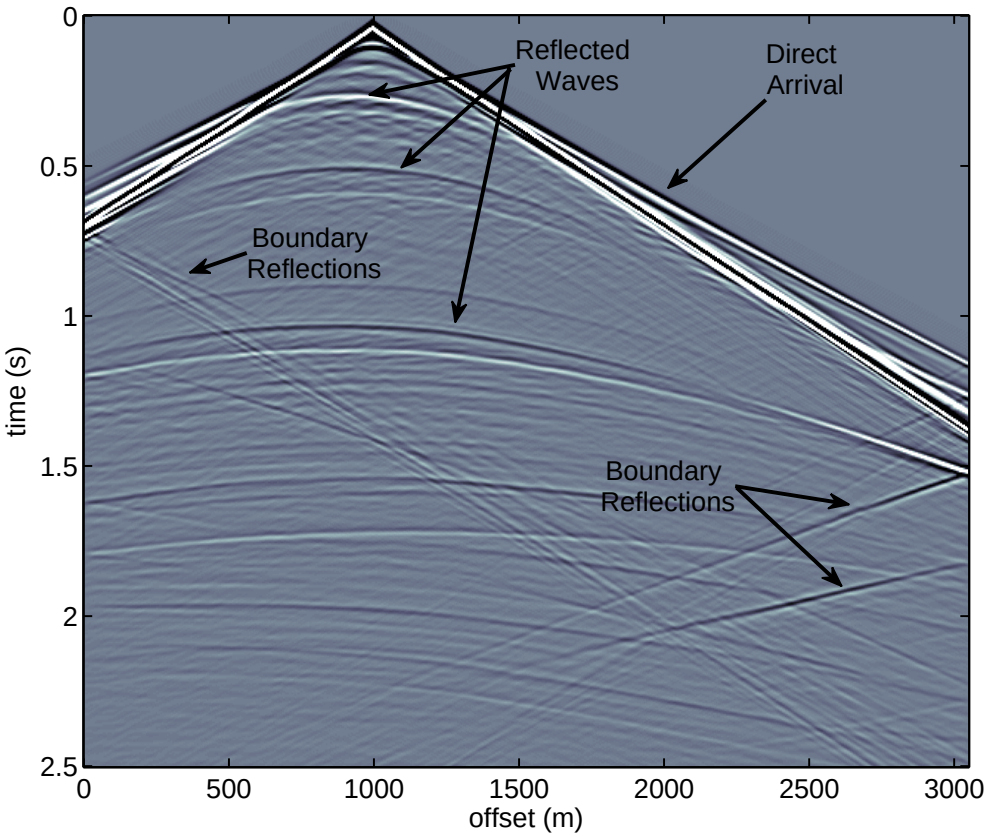



Figure 5.12: Surface data for Marmousi velocity model

# Chapter 6

# Elastic Waves

## 6.1 Weak Form of the Elastic Wave Equation

To define our method we first need to derive the *weak* form of the elastic wave equation. Consider the *strong* formulation of the elastic wave equation for an arbitrary isotropic heterogeneous medium $\Omega \in \mathbb{R}^d, d = 1, 2, 3$, with boundary $\partial\Omega = \Gamma$.

$$\begin{cases} \rho\ddot{u}_i = \partial_j \sigma_{ij}(\mathbf{u}) + f_i, & \mathbf{x} \in \Omega, t \geq 0 \\ \mathbf{u}(\mathbf{x}, t = 0) = \mathbf{u}_0(\mathbf{x}), & \mathbf{x} \in \Omega \\ \dot{\mathbf{u}}(\mathbf{x}, t = 0) = \mathbf{u}_1(\mathbf{x}), & \mathbf{x} \in \Omega \end{cases} \tag{6.1}$$

The stresses for an isotropic medium are

$$\sigma_{ij}(\mathbf{u}) = \lambda(\nabla \cdot \mathbf{u})\delta_{ij} + 2\mu\varepsilon_{ij}(\mathbf{u})$$

where $\partial_j$ denotes differentiation with respect to the $j^{\text{th}}$ element $x_j$ and

$$\varepsilon_{ij}(\mathbf{u}) = \frac{1}{2}\left(\partial_i u_j + \partial_j u_i\right).$$

Summation over repeated indices, as per Einstein notation, is assumed unless otherwise noted. The parameters $\lambda$, $\mu$ are the elastic constants of the medium and $\rho$ is the density. All may be bounded, spatially dependent, functions. $f_i(\mathbf{x}, t)$ is the $i^{\text{th}}$ component of the body force applied to the medium.

We obtain the *weak* form by multiplying both sides of 6.1 by an arbitrary test function $v \equiv v(\mathbf{x})$ and integrating over the entire space. This yields

$$\int_\Omega \rho \ddot{u}_i v d\Omega = \int_\Omega \partial_j \sigma_{ij}(\mathbf{u}) v d\Omega + \int_\Omega f_i v d\Omega. \tag{6.2}$$

Expanding the first term on the right hand side and applying Green's theorem gives us the relationship

$$\int_\Omega \partial_j \sigma_{ij}(\mathbf{u}) v d\Omega = \oint_\Gamma \sigma_{ij}(\mathbf{u}) v \hat{n}_j d\Gamma - \int_\Omega \sigma_{ij}(\mathbf{u}) \partial_j v d\Omega.$$

where $\hat{n}_j$ denotes the $j^{\text{th}}$ component of the outward-pointing normal vector. Substituting this into 6.2 yields the $i^{\text{th}}$ component of displacement of the weak form of 6.1

$$\int_\Omega \rho \ddot{u}_i v d\Omega + \int_\Omega \sigma_{ij}(\mathbf{u}) \partial_j v d\Omega = \int_\Omega f_i v d\Omega + \oint_\Gamma \sigma_{ij}(\mathbf{u}) v \hat{n}_j d\Gamma. \tag{6.3}$$

This is the form for which we will derive the numerical method. The boundary terms is what allows us to incorporate absorbing and free-surface boundary conditions. The most appropriate absorbing boundary conditions for our purposes are those for which the time and space derivatives appear independently. We omit the details of the derivations here for the sake of brevity as even the two-dimensional case involves 16 boundary integrals, 8 of which account for the absorbing boundary conditions (all of which factor into a single operator). We note, however, that there are several different choices available and refer the reader to [22], [21] and [18] for the construction and implementation of several higher-order methods that fit naturally into variational schemes.

We may then add the $d$ equations in 6.3 together to obtain

$$\frac{d^2}{dt^2}\int \rho \mathbf{u}\cdot\mathbf{v}d\Omega + \int_\Omega \sigma_{ij}(\mathbf{u})\partial_j v_i d\Omega = \int_\Omega \mathbf{f}\cdot\mathbf{v}d\Omega + \oint_\Gamma \sigma_{ij}(\mathbf{u})\hat{n}_j v_i d\Gamma. \tag{6.4}$$

For simplicity we present the following derivation for $d = 2$, but the result is the same in any number of dimensions. Expanding the integrand of the second term on the left hand side of 6.4 produces

$$\begin{aligned}&\sigma_{11}(\mathbf{u})\partial_1 v_1 + \sigma_{12}(\mathbf{u})\partial_2 v_1 + \sigma_{21}(\mathbf{u})\partial_1 v_2 + \sigma_{22}(\mathbf{u})\partial_2 v_2\\ &= \sigma_{11}(\mathbf{u})\varepsilon_{11}(\mathbf{v}) + \sigma_{12}(\mathbf{u})\partial_2 v_1 + \sigma_{21}(\mathbf{u})\partial_1 v_2 + \sigma_{22}(\mathbf{u})\varepsilon_{22}(\mathbf{v})\end{aligned} \tag{6.5}$$

adding together the second and third terms in (6.5) yields

$$\begin{aligned}&\mu(\partial_1 u_2 + \partial_2 u_1)\partial_2 v_1 + \mu(\partial_2 u_1 + \partial_1 u_2)\partial_1 v_2\\ &= \mu(\partial_1 u_2 + \partial_2 u_1)(\partial_2 v_1 + \partial_1 v_2)\\ &= \sigma_{12}(\mathbf{u})\varepsilon_{12}(\mathbf{v}) + \sigma_{21}(\mathbf{u})\varepsilon_{21}(\mathbf{v}).\end{aligned}$$

So we obtain

$$\int_\Omega \sigma_{ij}(\mathbf{u})\partial_j v_i d\Omega = \int_\Omega \sigma_{ij}(\mathbf{u})\varepsilon_{ij}(\mathbf{v})d\Omega$$

If we then define the vector $\mathbf{T}$ component-wise by

$$T_i = \sigma_{ij}(\mathbf{u})\hat{n}_j$$

we may write the second term in the right hand side of (5.4) as

$$\int_\Gamma \sigma_{ij}(\mathbf{u})\hat{n}_j v_i d\Gamma = \oint_\Gamma \mathbf{T}\cdot\mathbf{v}d\Gamma$$

and so (5.3) becomes

$$\frac{d^2}{dt^2}\int \rho \mathbf{u}\cdot\mathbf{v}d\Omega + \int_\Omega \sigma_{ij}(\mathbf{u})\varepsilon_{ij}(\mathbf{v})d\Omega = \int_\Omega \mathbf{f}\cdot\mathbf{v}d\Omega + \oint_\Gamma \mathbf{T}\cdot\mathbf{v}d\Gamma i, j = 1, \ldots, d.$$

In order to make the variational approach rigorous we need to identify a few function spaces in which our various trial and test function will reside. Define $H^1(\Omega)$ to be the classic Sobolev space of square integrable functions defined on $\Omega$, with square-integrable weak first derivatives. That is,

$$H^1(\Omega) = \left\{ f \in L^2(\Omega) \middle| D^1 f \in L^2(\Omega) \right\},$$

where

$$L^2(\Omega) = \left\{ f : \Omega \to \mathbb{R} \middle| \int_\Omega |f(\mathbf{x})|^2 d\Omega < \infty \right\}$$

and $D^1$ denotes the weak first derivative operator.

Then the vector-valued version is defined as those vector-valued functions whose components reside in $H^1(\Omega)$. Denote $L^2(\Omega)^d = L^2(\Omega) \times \cdots \times L^2(\Omega)$ and define this space to be

$$\mathcal{H} = \left\{ \mathbf{u}(\mathbf{x}) \in L^2(\Omega)^d | \nabla \mathbf{u} \in L^2(\Omega)^d \right\}.$$

The variational problem is thus: find $\mathbf{u} \in \mathcal{H}$ such that, for all $t \geq 0$,

$$\langle \rho \ddot{\mathbf{u}}, \mathbf{v} \rangle_\Omega + a(\mathbf{u}, \mathbf{v}) = \langle \mathbf{f}, \mathbf{v} \rangle_\Omega + \langle \mathbf{T}, \mathbf{v} \rangle_\Gamma, \forall \mathbf{v} \in \mathcal{H}. \tag{6.6}$$

The inner-products in (6.6) are the standard vector-valued $L^2(\Omega)$ inner-products

$$\langle \mathbf{f}, \mathbf{g} \rangle_\Omega = \int_\Omega \mathbf{f} \cdot \mathbf{g} d\Omega,$$

$$\langle \mathbf{f}, \mathbf{g} \rangle_\Gamma = \oint_\Gamma \mathbf{f} \cdot \mathbf{g} d\Gamma$$

and the norm on $\mathcal{H}$ is given by

$$\|\mathbf{v}\|_\mathcal{H} = \left( |\langle \mathbf{v}, \mathbf{v} \rangle_\Omega|^2 + |\langle \nabla \mathbf{v}, \nabla \mathbf{v} \rangle_\Omega|^2 \right)^{1/2}$$

To show that there exists a unique solution to (5.6) we need to show that

$$a(\mathbf{u},\mathbf{v}) = \int_{\Omega} \sigma_{ij}(\mathbf{u})\varepsilon_{ij}(\mathbf{v})d\Omega$$

is a symmetric, V-elliptic and continuous bi-linear form and apply the Lax-Milgram lemma [1].

To see that $a(\mathbf{u},\mathbf{v})$ is symmetric and bi-linear we must show that the following all hold:

1. $a(\mathbf{u},\mathbf{v}) = a(\mathbf{v},\mathbf{u})$, $\forall \mathbf{u},\mathbf{v} \in \mathcal{H}$,

2. $a(\mathbf{u}+\mathbf{u}',\mathbf{v}) = a(\mathbf{u},\mathbf{v}) + a(\mathbf{u}',\mathbf{v})$, $\forall \mathbf{u},\mathbf{u}',\mathbf{v} \in \mathcal{H}$,

3. $a(\alpha\mathbf{u},\mathbf{v}) = \alpha a(\mathbf{u},\mathbf{v})$, $\forall \mathbf{u},\mathbf{v} \in \mathcal{H}$, and $\alpha \in \mathbb{R}$.

The second and third equalities follow trivially from the linearly of the derivative. For brevity we show the first equality only for $d = 2$. This follows from expanding $\sigma_{ij}(\mathbf{u})\varepsilon_{ij}(\mathbf{v})$ as

$$\sigma_{ij}(\mathbf{u})\varepsilon_{ij}(\mathbf{v}) = \sigma_{11}(\mathbf{u})\varepsilon_{11}(\mathbf{v}) + \sigma_{12}(\mathbf{u})\varepsilon_{12}(\mathbf{v}) + \sigma_{21}(\mathbf{u})\varepsilon_{21}(\mathbf{v}) + \sigma_{22}(\mathbf{u})\varepsilon_{22}(\mathbf{v}).$$

$$= [\lambda(\nabla\cdot\mathbf{u})+2\mu\varepsilon_{11}(\mathbf{v})]\varepsilon_{11}(\mathbf{v})+2\mu\varepsilon_{12}(\mathbf{u})\varepsilon_{12}(\mathbf{v})+2\mu\varepsilon_{21}(\mathbf{u})\varepsilon_{21}(\mathbf{v})+[\lambda(\nabla\cdot\mathbf{u})+2\mu\varepsilon_{22}(\mathbf{v})]\varepsilon_{22}(\mathbf{v}).$$

The terms corresponding to $i \neq j$ are symmetric and so we only need to show that the sum of the $i = j$ terms are as well. Adding together the first and last term produces

$$\begin{aligned}&[\lambda(\nabla\cdot\mathbf{u}) + 2\mu\varepsilon_{11}(\mathbf{v})]\varepsilon_{11}(\mathbf{v}) + [\lambda(\nabla\cdot\mathbf{u}) + 2\mu\varepsilon_{22}(\mathbf{v})]\varepsilon_{22}(\mathbf{v})\\ &= \lambda(\nabla\cdot\mathbf{u})[\varepsilon_{11}(\mathbf{v}) + \varepsilon_{22}(\mathbf{v})] + 2\mu[\varepsilon_{11}(\mathbf{u})\varepsilon_{11}(\mathbf{v}) + \varepsilon_{22}(\mathbf{u})\varepsilon_{22}(\mathbf{v})]\end{aligned}$$

$$= \lambda(\nabla \cdot \mathbf{u})(\nabla \cdot \mathbf{v}) + 2\mu(\nabla \mathbf{u} \cdot \nabla \mathbf{v})$$

which is symmetric and, thus, $a(\mathbf{u}, \mathbf{v})$ is symmetric and bi-linear. Continuity follows directly from our assumptions that all the functions included in our scheme are bounded.

A bi-linear form $a(\cdot, \cdot)$ is V-elliptic, or strongly positive, if there exists $C > 0$ such that

$$a(\mathbf{v}, \mathbf{v}) \geq C\|\mathbf{v}\|^2_{\mathcal{H}}, \forall v \in V.$$

This follows directly from Korn's inequality [15] which ensures us that there exists a constant $K$ depending only on $\Omega$ such that:

$$\|\mathbf{v}\|^2_{\mathcal{H}} \leq K(\Omega) \int_{\Omega} \{\mathbf{v} \cdot \mathbf{v} + \varepsilon_{ij}(\mathbf{v})\varepsilon_{ij}(\mathbf{v})\}\, d\Omega, \forall \mathbf{v} \in \mathcal{H}.$$

## 6.2 Pseudospectral Elements

As with any method involving domain-decomposition we decompose $\Omega$ into a union of smaller subdomains,

$$\Omega = \bigcup_{k=1}^{M} \Omega_k.$$

On each element we then define a tensor-product grid of LGL nodes and make the definition that the edges of each element share the associated nodes, as seen in figure 6.2.

For the elastic wave equation in 2D we are solving for two components of displace-

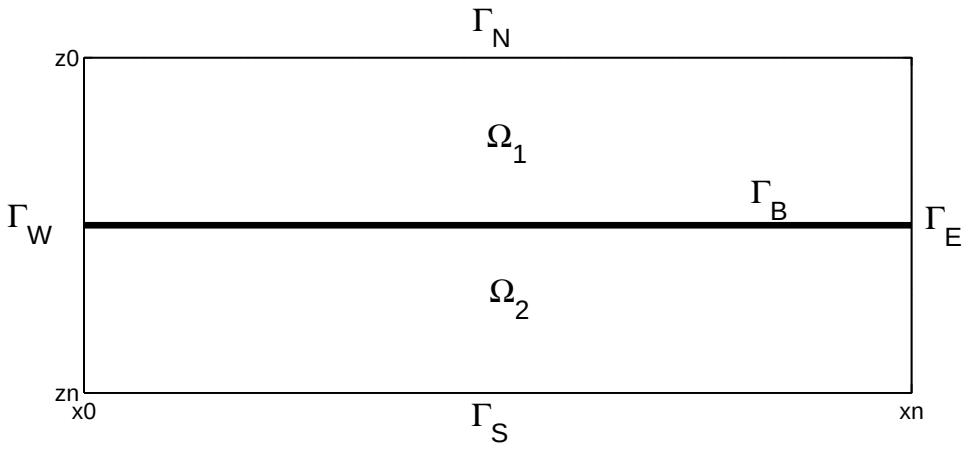



Figure 6.1: Two subdomains and their shared boundaries over the entire domain.

ment $u_1(x,z,t)$ and $u_2(x,z,t)$ which are the horizontal and vertical displacement of the medium. In domain decomposition methods we then define these to be combinations of the contributions over each subdomain.

$$u_i(x,z,t) = \sum_{k=1}^{M} u_i^k(x,z,t)$$

and split up the weak form 6.3 over the subdomains. This results in

$$\int_{\Omega_k} \rho \ddot{u}_i v d\Omega_k + \int_{\Omega_k} \sigma_{ij}(\mathbf{u}) \partial_j v d\Omega_k = \int_{\Omega_k} f_i v d\Omega_k + \oint_{\Gamma_k} \sigma_{ij}(\mathbf{u}) v \hat{n}_j d\Gamma_k. \tag{6.7}$$

where $\Gamma_k$ is the boundary of the $k^{th}$ subdomain $\Omega_k$. To enforce proper interface conditions we require the displacements to be continuous across the boundaries of each element and that the stresses across the interface vanish. Thus, the boundary integral vanishes everywhere except at the boundaries where we enforce the absorbing boundary conditions. The continuity of displacement is represented by defining the basis functions associated with the edge nodes to be the piece-wise continuous

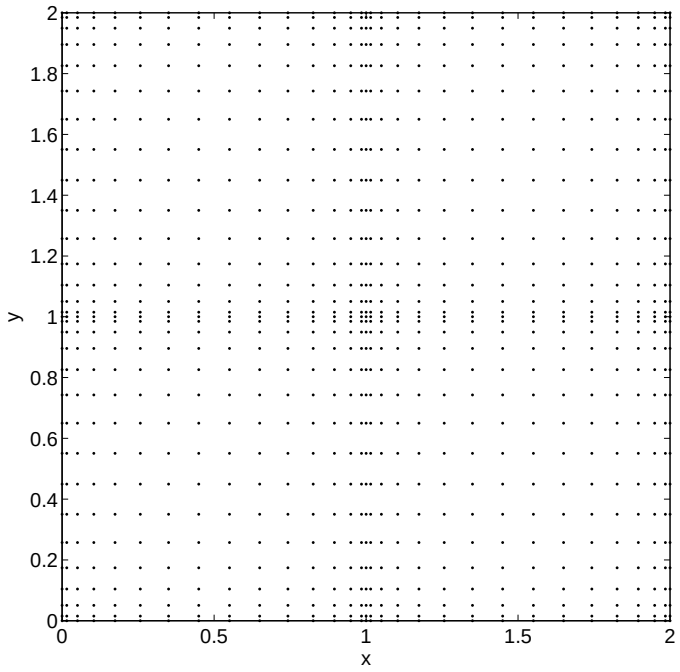



Figure 6.2: 2D Legendre-Gauss-Lobatto SEM nodes distributed over 4 subdomains.

functions constructed by equating the basis functions from each element. Several examples of these functions are seen in figure 6.3.

Interior to each domain, equation 6.7 is discretized using pseudospectral differentiation matrices and integration weights by writing

$$u_i^k(x,t) = \sum_{i=1}^{n} u_i^k(\mathbf{x}_i,t) l_i(\mathbf{x},z).$$

Substituting this into 6.7 and choosing the functions $v$ to be equal to $l_j(x,z)$ produces the system of equations for the vector of nodal values $\mathbf{u}_i^k(t)$ in the $k^{th}$ element

$$M^k \ddot{\mathbf{u}}_i^k(t) + A_i^k \dot{\mathbf{u}}_i^k(t) + \sum_j K_{ij}^k \mathbf{u}_j^k(t) = M^k \mathbf{f}_i^k(t)$$

The element mass matrix $M^k$ is a diagonal matrix with the integration weights along

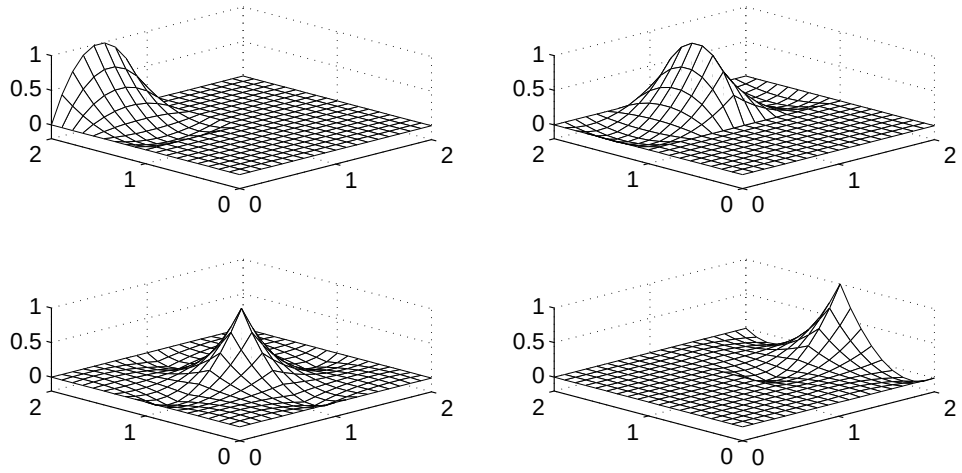



Figure 6.3: 2D SEM basis functions defined on 4 elements.

the main diagonal and the structure of the element damping matrices $A_i^k$ depends on the absorbing boundary conditions but is generally diagonal and only non-zero along the main diagonal at the positions corresponding to the indices of nodes along outer boundaries. The element stiffness matrix $K_{ij}^k$ are the discrete representation of the integro-differential operator in the $i^{th}$ equation 6.7 acting on the nodal values of the $j^{th}$ component of the displacement. The global mass, damping and stiffness matrices are assembled by transforming their respective indices into the global indices and summing over the connected nodes. This is done using the so called *connectivity* matrix wherein the $i^{th}$ column contains the global node numbers of the $i^{th}$ element. This is easier to portray in an example. In figure 6.4 we show 4 elements defined on $[-1, 1] \times [-1, 1]$ numbered column-major. If we number the global nodes column-

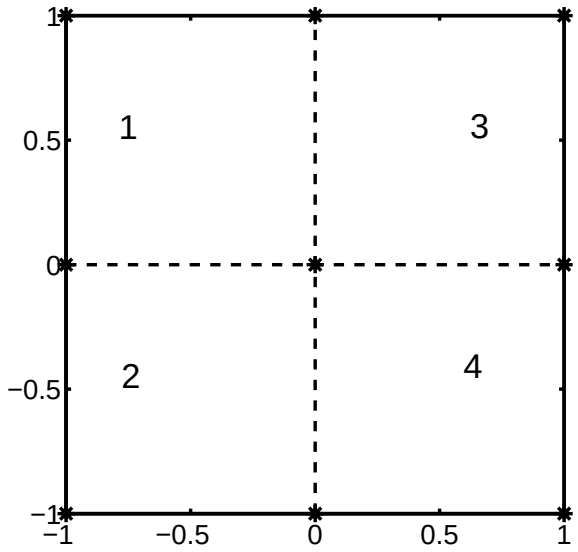



Figure 6.4: 4 elements with 4 nodes each for a total of 9 global nodes.

major as well, the connectivity matrix is defined as

$$C = \begin{pmatrix} 1 & 2 & 4 & 5 \\ 2 & 3 & 5 & 6 \\ 4 & 5 & 7 & 8 \\ 5 & 6 & 8 & 9 \end{pmatrix}$$

Using the connectivity matrix we could assemble the global stiffness matrix via Code 7. This, however, requires 3 for-loops and, for large numbers of nodes and/or elements, takes extremely long. It also assembles a fully-populated matrix that would be a drastic waste of memory.

```
for i=1:Nx*Nz
  Dxi=Dx*2/dXk(i);
  Dzi=Dz*2/dZk(i);
  Mi=M*(dXk(i)*dZk(i))/4;
  Ki=Dxi.'*Vp(i)*Vp(i)*Mi*Dxi+Dzi.'*Vp(i)*Vp(i)*Mi*Dzi;
```

```
  for j=1:Np*Np
    for k=1:Np*Np
      K(C(k,i),C(j,i))=K(C(k,i),C(j,i))+Ki(j,k);


    end
  end
end
```

Code 7: Assemble the global stiffness matrix *very* slowly.

A much better way of assembly is to define 3 vectors containing the row and column indices and element entries of the matrix and then call the Matlab function *sparse* to define the global matrix. This is done for a single block of the larger block stiffness matrix in code 8. In practice it is again, much faster to assemble all the blocks at the same time, but the code is much longer and less readable.

```
for i=1:Nx*Nz
  Dxi=Dx*2/dXk(i);
  Dzi=Dz*2/dZk(i);
  Mi=M*(dXk(i)*dZk(i))/4;
  Ki=(Dxi.'*Vp(i)*Vp(i)*Mi*Dxi+Dzi.'*Vp(i)*Vp(i)*Mi*Dzi);
  for j=1:Np*Np
    idx=((j-1)*Np^2+1:j*Np^2)+(i-1)*Np*Np*Np*Np;
    I(idx)=C(j,i)*ones(Np^2,1); % row positions
    J(idx)=C(:,i);% col positions
```

```
    X(idx)=Ki(:,j); % entries
  end
end
dim=(Nz*(Np-1)+1)*(Nx*(Np-1)+1);
K=sparse(I,J,X,dim,dim);
```

Code 8: Assemble the global stiffness matrix using sparse.

The vectors dXk and dZk are the width and height of the elements and are used to map the differentiation matrices and integration weights from local coordinates to global coordinates.

Once the global system is assembled it can be written in block matrix form

$$\begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix} \frac{\partial^2}{\partial t^2} \begin{pmatrix} \mathbf{u}_1^k \\ \mathbf{u}_2^k \end{pmatrix}(t) + \begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix} \frac{\partial}{\partial t} \begin{pmatrix} \mathbf{u}_1^k \\ \mathbf{u}_2^k \end{pmatrix}(t)$$

$$+ \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & A_{22} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1^k \\ \mathbf{u}_2^k \end{pmatrix}(t) = \begin{pmatrix} \mathbf{f}_1^k \\ \mathbf{f}_2^k \end{pmatrix}(t)$$

which we will write simply as

$$M\ddot{\mathbf{u}} + A\dot{\mathbf{u}} + K\mathbf{u} = \mathbf{F}. \tag{6.8}$$

The sparsity pattern of stiffness matrix $K$ is shown in figure 6.5.

## 6.3 Time Stepping

To deal with the time-dependent system a numerical procedure must be implemented that is capable of handling the first and second order derivatives in equation (6.8).

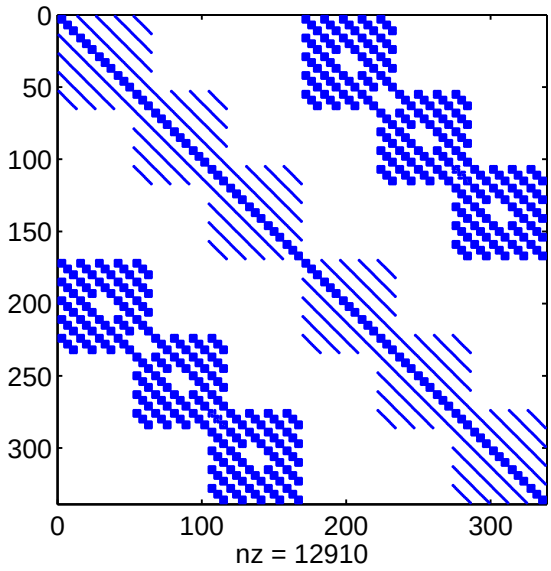



Figure 6.5: Sparsity patterns of the stiffness matrix.

A second order in time scheme can be constructed by replacing the derivatives with second-order central difference approximations

$$\ddot{\mathbf{u}}(t_j) = \frac{\mathbf{u}(t_{j+1}) - 2\mathbf{u}(t_j) + \mathbf{u}(t_{j-1})}{\Delta t^2} + \mathcal{O}(\Delta t^2),$$

$$\dot{\mathbf{u}}(t_j) = \frac{\mathbf{u}(t_{j+1}) - \mathbf{u}(t_{j-1})}{2\Delta t} + \mathcal{O}(\Delta t^2).$$

After dropping the error term, (6.8) then becomes

$$M\left(\frac{\mathbf{u}(t_{j+1}) - 2\mathbf{u}(t_j) + \mathbf{u}(t_{j-1})}{\Delta t^2}\right) + A\left(\frac{\mathbf{u}(t_{j+1}) - \mathbf{u}(t_{j-1})}{2\Delta t}\right) + K\mathbf{u}(t_j) = M\mathbf{F}(t_j)$$

or in terms of the $t_j$'s

$$\left[M + \frac{\Delta t}{2}A\right]\mathbf{u}(t_{j+1}) + \left[\Delta t^2 K - 2M\right]\mathbf{u}(t_j) + \left[M - \frac{\Delta t}{2}A\right]\mathbf{u}(t_{j-1}) = \Delta t^2 M\mathbf{F}(t_j)$$

The matrix $\left[M + \frac{\Delta t}{2}A\right]$ must now be inverted in order to step forward in time. Since both the matrices $M$ and $A$ are diagonal this is trivial. The method is implemented in Code 9 where the solution is returned sampled at SR ms.

```
function [U t] = CFD_SR(M,A,K,U1,U2,tn,dt,fx,ft,SR)
Np=length(fx);
P=(M+.5*dt*A)\(2*M-dt*dt*K);
Q=(M+.5*dt*A)\(.5*dt*A-M);
Fx=dt*dt*((M+.5*dt*A)\M)*fx;
numskip=ceil(SR/dt);
numkept=ceil(tn/(numskip*dt));
t=0:dt:(numskip*numkept*dt);
Ft=ft(t);
U=zeros(Np,numkept+1);
for k=1:numkept
  for j=1:numskip
    U3=P*U2+Q*U1+Ft(j+(k-1)*numskip)*Fx;
    U1=U2;
    U2=U3;
  end
  U(:,k+1)=U3;
end
t = 0:(numskip*dt):(numskip*numkept*dt);
```

Code 9: Matlab function for time-stepping mixed order ODE systems by central finite-differences.

Another way to time-step the problem would be to rewrite it as a first order system by making the substitution $\mathbf{v} = \dot{\mathbf{u}}$. Then (6.8) can be rewritten as

$$\begin{bmatrix} \dot{\mathbf{u}} \\ \dot{\mathbf{v}} \end{bmatrix} + \begin{bmatrix} 0 & I \\ M^{-1}K & M^{-1}A \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{F} \end{bmatrix}$$

and solved by an appropriate method for first-order systems. In Code 10 we define a function that time steps this equation by the $4^{th}$-order low-storage explicit-Runge-Kutta scheme [7] and returns the solution sampled at SRms.

```
function [U t] = LSERK_SR(K,U0,tn,dt,fx,ft,SR)
Np=length(U0);
numskip=ceil(SR/dt);
numkept=ceil(tn/(numskip*dt));
t=0:dt:(numskip*numkept*dt);
U=zeros(Np,numkept+1);
U(:,1)=U0;
Pk=zeros(Np,1);
Kk=zeros(Np,1);
[a b c]=LSERKcoefs;
for i=1:numkept
  for j=1:numskip
    for k=1:5
      Kk=a(k)*Kk+dt*(K*Pk+fx*ft(t(j+(i-1)*numskip)+c(k)*dt));
      Pk=Pk+b(k)*Kk;
    end
```

```
  end
  U(:,i+1)=Pk;
end
t = 0:(numskip*dt):(numskip*numkept*dt);
```

Code 10: Matlab function for 1st order ode systems by LSERK.

The LSERK method is desirable over standard Runge-Kutta methods in that it only requires a single extra level of storage, while a standard RK45 scheme requires an extra 5. The trade-off, however, is an extra level of computation.

## 6.4 Numerical Results

To test the method we consider a forcing term of the form

$$\mathbf{F}(\mathbf{x},t) = \delta(\mathbf{x}-\mathbf{x}_0)f(t)$$

where the time-component is a Ricker wavelet

$$f(t) = \frac{2}{\sqrt{3\sigma}\pi^{\frac{1}{4}}}\left(1-\frac{t^2}{\sigma^2}\right)e^{\frac{-t^2}{2\sigma^2}}.$$

The model is a simple 2-layer medium with $V_p$,$V_s$, and $\rho$ constant in each layer. The propagating wavefield is shown in figures 6.6 to 6.11. At first we can see the pressure, shear and surface waves originating from the source term. Then, as the wave propagates through the interface each wave is converted into more pressure and shear waves until, finally, as the waves reach the side and bottom boundaries, they are absorbed.

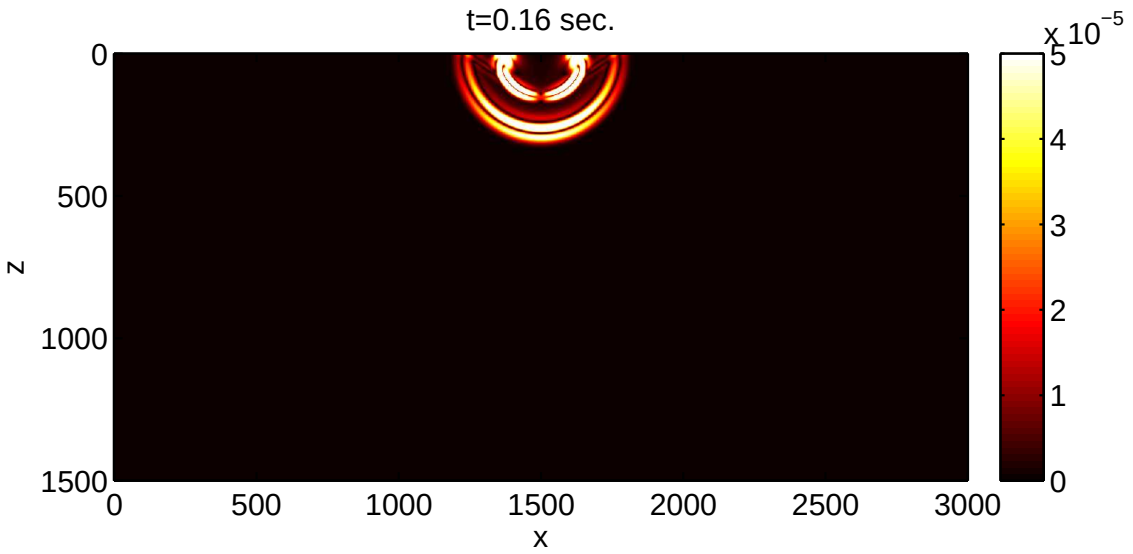



Figure 6.6: 2-Norm of elastic displacement.

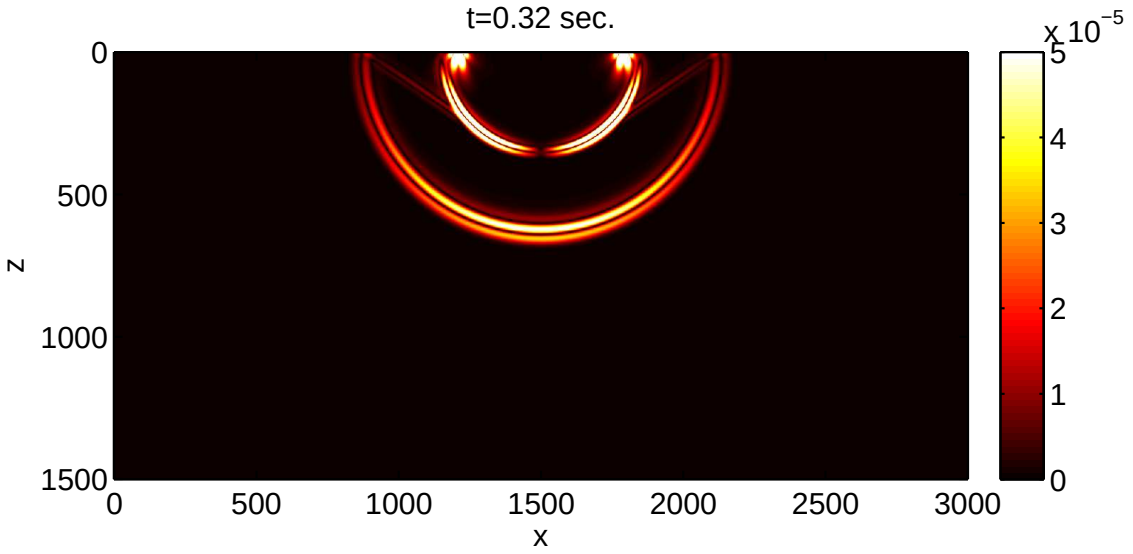



Figure 6.7: 2-Norm of elastic displacement.

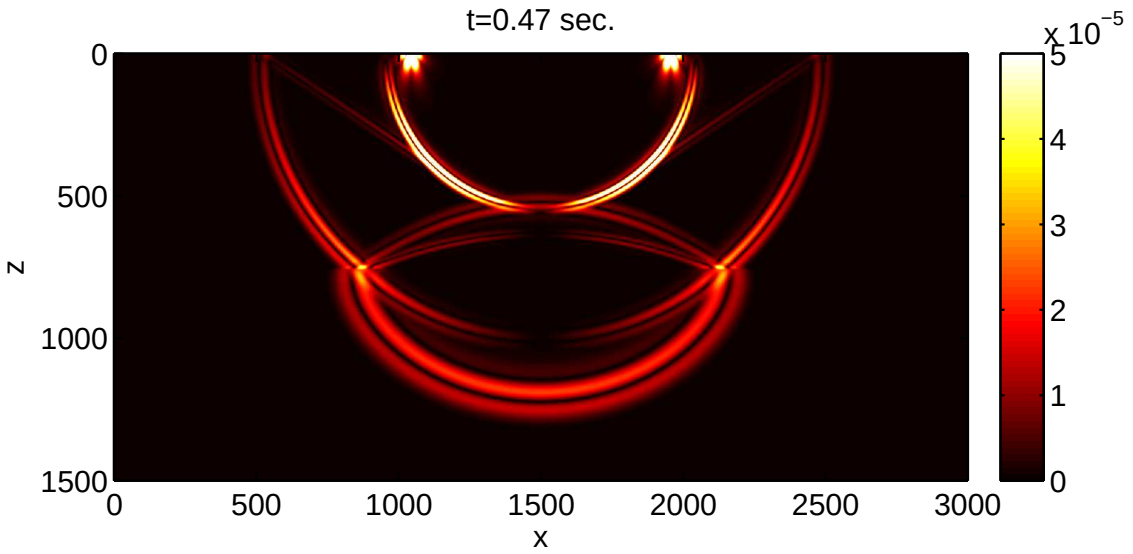



Figure 6.8: 2-Norm of elastic displacement.

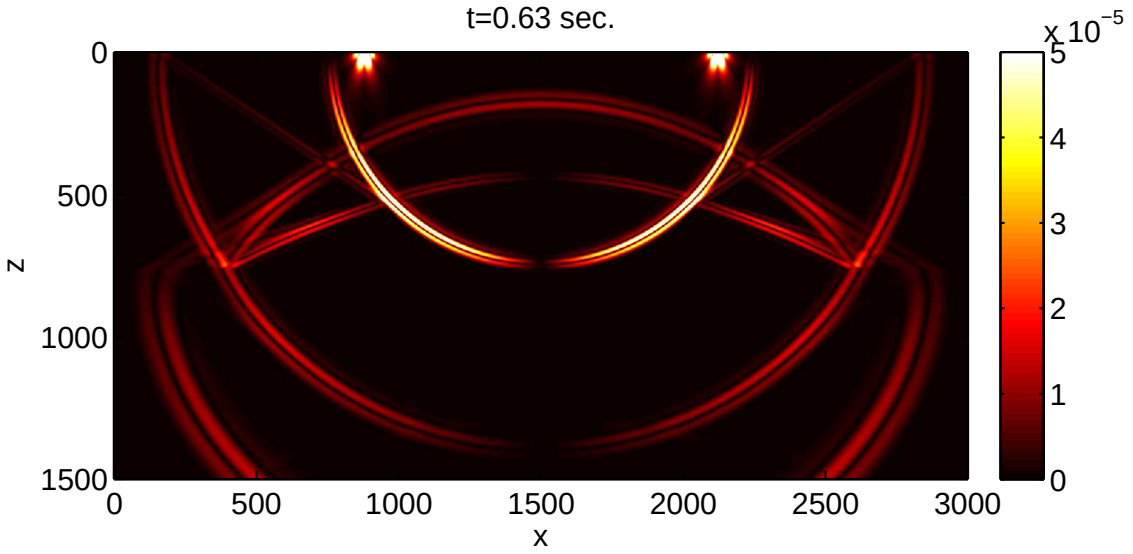



Figure 6.9: 2-Norm of elastic displacement.

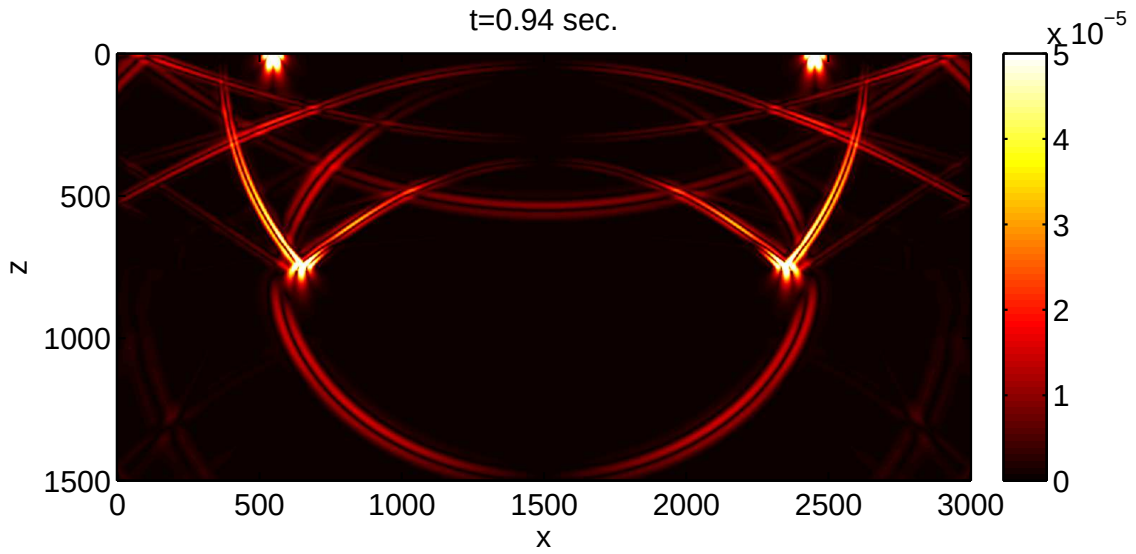



Figure 6.10: 2-Norm of elastic displacement.

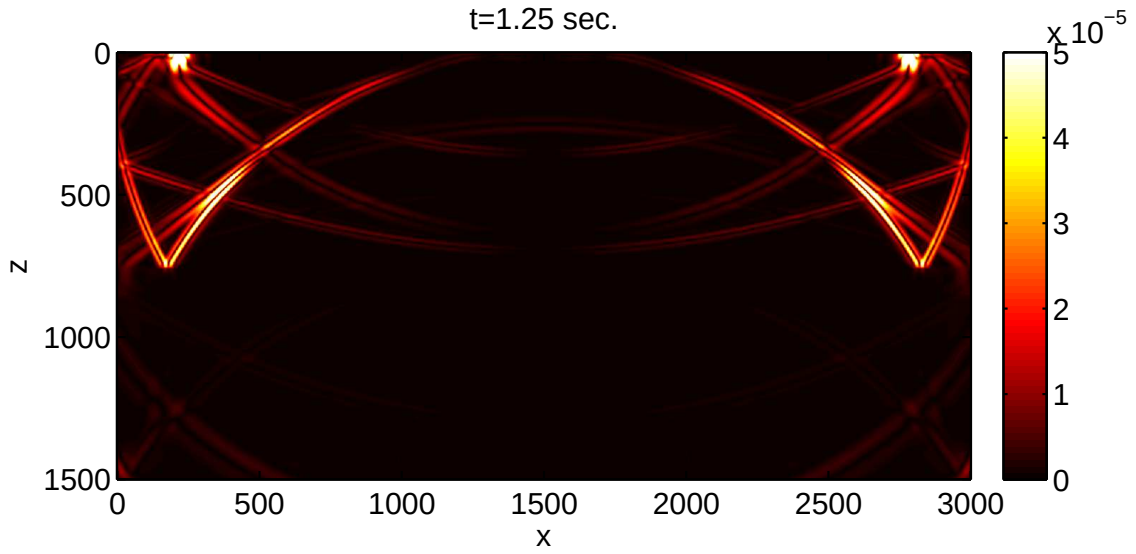



Figure 6.11: 2-Norm of elastic displacement.

Next, a nodal Galerkin method is compared to fourth and second order finite difference methods on a 501 by 501 node grid. To test this we consider a forcing

term with Ricker wavelet time-component and conservative spatial component

$$(u_1(\mathbf{x}), w_1(\mathbf{x})) = -\nabla e^{-r\|\mathbf{x}-\mathbf{x_0}\|^2}$$

and propagate a 15 Hz wavelet in a 4500m square bipartite medium with properties listed in the following table.

| Region | $\rho$ | $V_p$ | $V_s$ |
|---|---|---|---|
| 1 | 2.064 | 2305 | 997 |
| 2 | 2.14 | 4500 | 2600 |

Figures 6.12, 6.13 and 6.14 show the norm of the displacement for the three models propagated to 1 second and then normalized and clipped to exaggerate the dispersion effects. The extended arcs in the fourth-order model result from the wider stencil moving over the large step in the velocity model and then being propagated. Again, this is exaggerated here and is mainly due to the relatively small number of grid points we are using, but the effect is apparent.

The computation times are listed with the figures but are not very indicative of the associated computation costs of the three methods. The implementation of the three methods are nearly identical, with the only difference being the application of the derivative approximations. For the finite-difference methods the cost of this is $k * N^2$ where $k$ is the width of the finite-difference stencil (3 for second order, 5 for fourth order). The differentiation matrices for the nodal Galerkin methods can be considered finite-difference matrices with stencils of width $N$ and so the cost of applying these methods is $N^3$. We take $dt = .0008$ and so take 1179 steps to reach

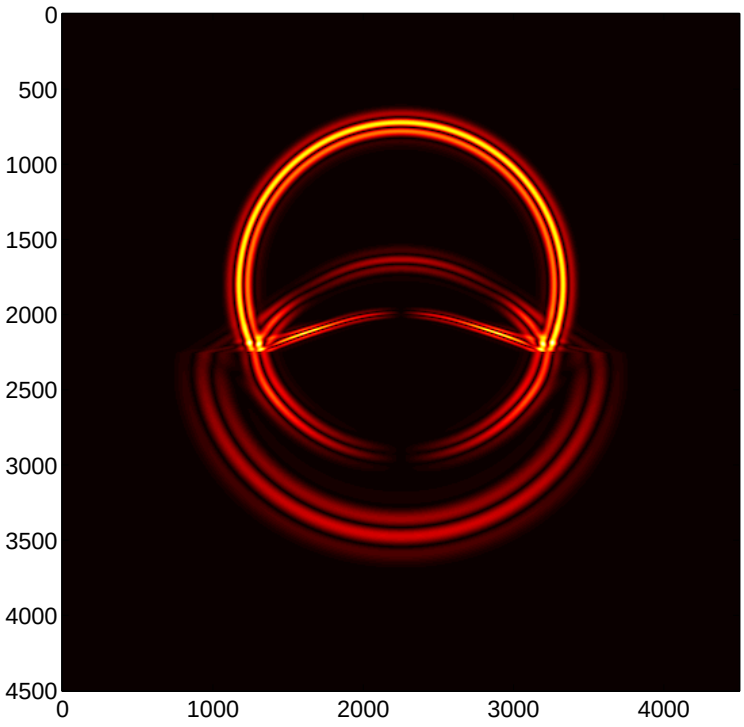



Figure 6.12: Nodal Galerkin. Comp time = 206 s.

one second. We could take a smaller time step, but the one chosen assures us that the wavelet in time is well-represented and that the error associated with time stepping will not taint our results as we are more interested in spatial accuracy.

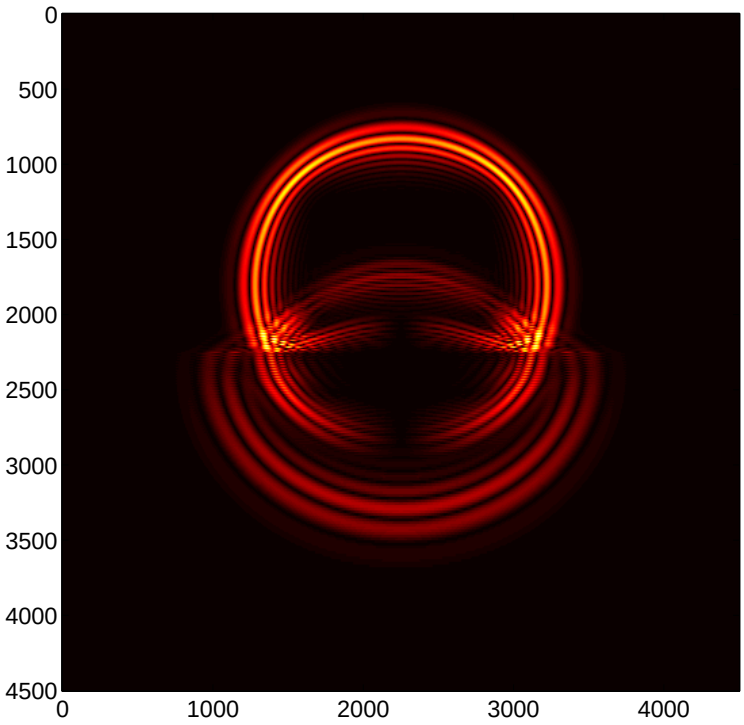



Figure 6.13: Second-Order Finite Difference. Comp time = 64 s.

Locally the fourth order model approximates the wavefronts better than the second-order model (as is expected), but the size of the stencil means that we must alter it somehow at the boundaries, this is not the case with the differentiation matrices that appear in the nodal Galerkin methods as they are global and so for the approximation of the derivative at one node they take information from all other

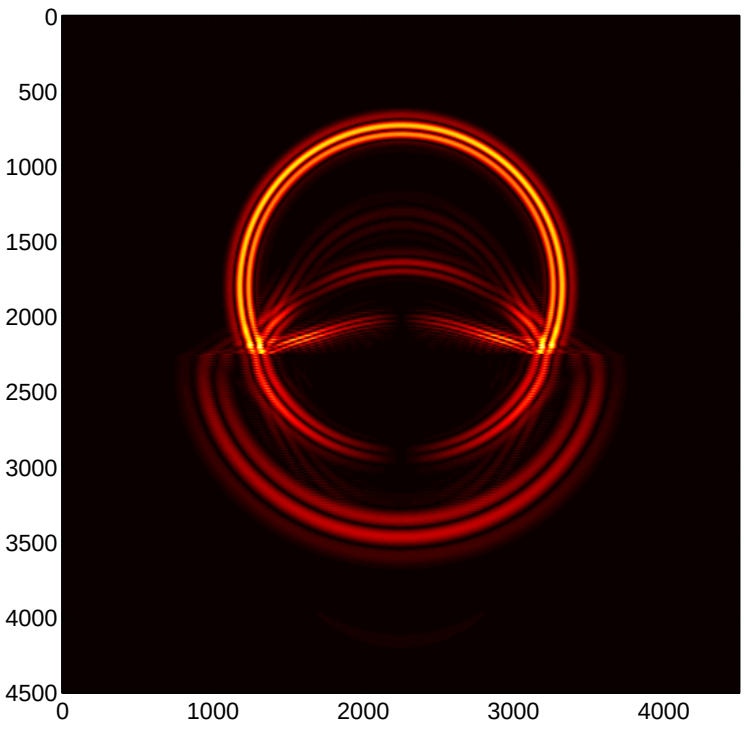



Figure 6.14: Fourth-Order Finite Difference. Comp time = 75 s.

nodes in the model.

Figures 6.15 and 6.16 show comparisons of the centerline of the model ($x = 2250$, for all $z$) at time corresponding wavefront in various regions of the velocity model. The amplitude error associated with the second-order stencil is apparent as is the dispersion of all three methods near a jump.

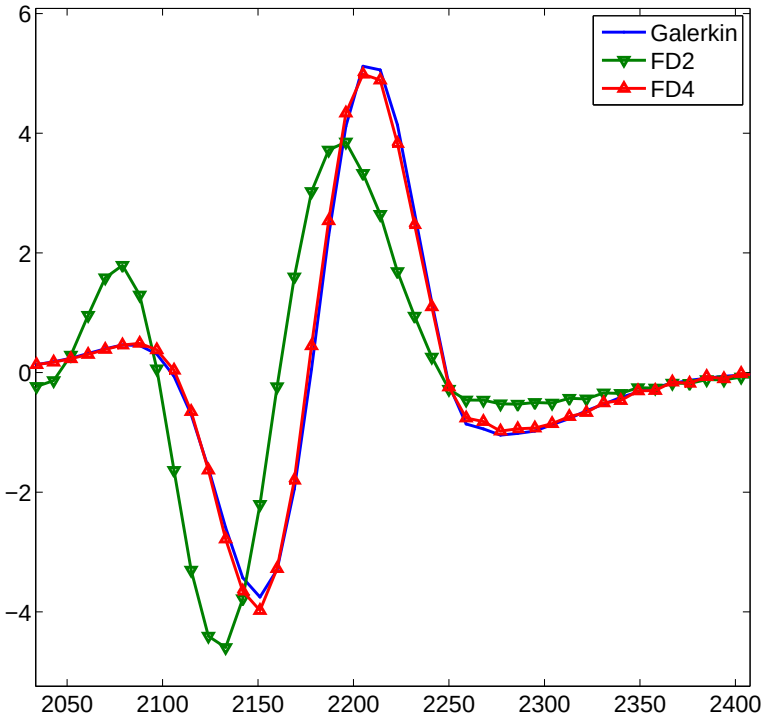



Figure 6.15: Close-up of the centerline of the horizontal component of a 2D elastic wave propagated to $t = .4150$ sec. The region plotted shows the disagreement of the three methods in a smooth region of the velocity model.

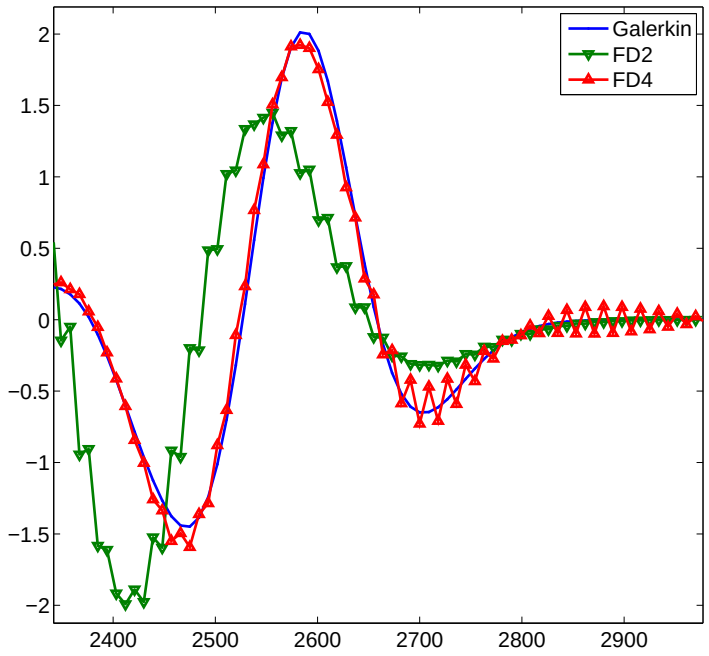



Figure 6.16: Close-up of the centerline of the horizontal component of a 2D elastic wave propagated to $t = .6$ sec. The region plotted shows the disagreement of the three methods in a the presence of a sharp jump in the velocity model.

# Chapter 7

# Viscoelastic Waves

## 7.1 Introduction

We present a method for numerically modelling viscoelastic wave propagation using domain decomposition combined with a pseudospectral method based on Legendre-Gauss-Lobatto nodes defined on a structured quadrilateral grid. The physics of the method is modelled using the Kelvin-Voigt equation for the time-dependent relation of stress and strain. Here we derive a coupled system of first-order equations for the particle velocities and accelerations which only doubles the number of required equations as opposed to the increase from 2 to 5 in the 2D case and 3 to 9 in the 3D case required when modelling the accelerations and stresses. Working with the first order system also allows us to incorporate absorbing boundary conditions by modifying the damping matrix at the boundary nodes in a way that further increases sparsity of the damping matrix and allows us to maintain the use of a low-storage explicit Runge-Kutta time-stepping algorithm.

## 7.2 Viscoelastic Models

When a material is viscoelastic it means that the materials response to an applied stress is time-dependent, i.e. not instantaneous [24, 4]. What this means is that the strain of a viscoelastic material due to an applied stress is time delayed, the material has "memory" [24]. Viscoelastic behaviour is a prevalent feature in hydrocarbon reservoirs, for instance, heavy oils are viscoelastic [24] and the ability to determine the viscosity in heavy oil reservoirs could greatly impact drilling programs and lead to the recovery of potentially stranded reserves [24]. The Kelvin-Voigt model for viscoelastic behaviour using springs and dash-pots is shown in figure 7.1. Here, $E$ is the modulus of elasticity and $\eta$ is the viscosity of the model.

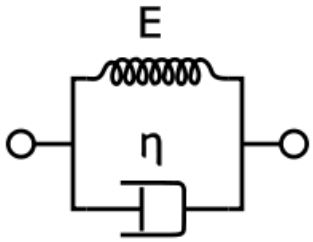



Figure 7.1: Kelvin-Voigt spring and damper model.

The relationship between stress, $\sigma$, and strain, $\varepsilon$, is

$$\sigma = E\varepsilon + \eta\dot{\varepsilon},$$

where $\dot{\varepsilon}$ is the derivative with respect to time of the strain. In two spatial dimensions there are three independent stresses, $\{\sigma_{11}, \sigma_{22}, \sigma_{12}\}$, and strains $\{\varepsilon_{11}, \varepsilon_{22}, \varepsilon_{12}\}$. The

strain operator is defined

$$\varepsilon_{ij}(\mathbf{u}) = \frac{1}{2}\left(\partial_j u_i + \partial_i u_j\right)$$

where

$$\partial_i u_j = \frac{\partial u_j}{\partial x_i}$$

For an isotropic medium the stress-strain relation is the matrix equation

$$\begin{pmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{12}\end{pmatrix} = \begin{pmatrix}\lambda+2\mu & \lambda & 0\\ \lambda & \mu & 0\\ 0 & \lambda & \lambda+2\mu\end{pmatrix}\begin{pmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ \varepsilon_{12}\end{pmatrix} + \begin{pmatrix}\lambda'+2\mu' & \lambda' & 0\\ \lambda' & \mu' & 0\\ 0 & \lambda' & \lambda'+2\mu'\end{pmatrix}\begin{pmatrix}\dot{\varepsilon}_{11}\\ \dot{\varepsilon}_{22}\\ \dot{\varepsilon}_{12}\end{pmatrix}$$

We can write this component-wise in terms of the vectors of displacements $\mathbf{u}$ and velocities $\mathbf{v}$ as

$$\sigma_{ij} = \lambda\nabla\cdot\mathbf{u}\delta_{ij} + 2\mu\varepsilon_{ij}(\mathbf{u}) + \lambda'\nabla\cdot\mathbf{v}\delta_{ij} + 2\mu'\varepsilon_{ij}(\mathbf{v}).$$

Assuming the displacements are of the form

$$u_j(x,z,t) = \hat{u}_j(x,z)e^{i\omega t}$$

produces

$$\begin{aligned}\sigma_{ij} &= \lambda\nabla\cdot\hat{\mathbf{u}}\delta_{ij} + 2\mu\varepsilon_{ij}(\hat{\mathbf{u}}) + i\omega\left(\lambda'\nabla\cdot\hat{\mathbf{u}}\delta_{ij} + 2\mu'\varepsilon_{ij}(\hat{\mathbf{u}})\right)\\ &= \Lambda\nabla\cdot\hat{\mathbf{u}}\delta_{ij} + 2M\varepsilon_{ij}(\hat{\mathbf{u}})\end{aligned}$$

where $\Lambda = \lambda + i\omega\lambda'$ and $M = \mu + i\omega\mu'$ are the complex Lamé parameters dependent on the frequency $\omega$. This is the so-called correspondence principle, which assures that, for a given elastic model a viscoelastic counterpart is also available. Naturally then, the complex P and S wave velocities are defined as

$$\hat{V}_p = \sqrt{\frac{\Lambda+2M}{\rho}}, \quad \text{and} \quad \hat{V}_s = \sqrt{\frac{M}{\rho}}$$

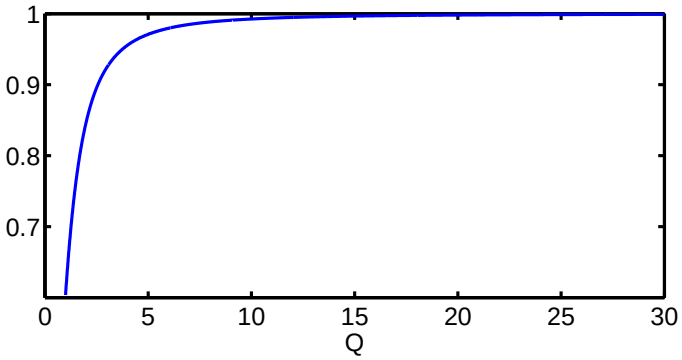



Figure 7.2: The function $g(Q)$.

The real velocities are obtained as

$$V_\alpha = \Re\left(\frac{1}{\hat{V}_\alpha}\right)^{-1}$$

where $\alpha = P$ or $S$. The frequency-dependent P and S wave quality factors are given by

$$Q_p = \frac{\lambda + 2\mu}{\omega(\lambda' + 2\mu')}, \quad \text{and} \quad Q_s = \frac{\mu}{\omega\mu'}.$$

Thus, we can obtain the elastic parameters, $\lambda$ and $\mu$ as

$$\mu = \rho V_s^2 g(Q_s)$$

and

$$\lambda = \rho V_p^2 g(Q_p) - 2\mu$$

where $g$ is obtained algebraically from the above equations as

$$g(Q) = \frac{1}{2}(1 + Q^{-2})^{-1/2}(1 + (1 + Q^{-2})^{-1/2}).$$

The viscoelastic parameters are then obtained from

$$\lambda' = \frac{1}{\omega}\left(\frac{\lambda + 2\mu}{Q_p} - \frac{2\mu}{Qs}\right)$$

and

$$\mu' = \frac{1}{\omega Q_s}.$$

The inputs of the model are thus $\rho$, $V_p$, $V_s$, $Q_p$ and $Q_s$. The choice of $\omega$ is arbitrary but a good choice is generally the dominant frequency of the source wavelet

Further analysis of the viscoelastic properties of the model are available in [5].

## 7.3 Spatial discretization

The strong-form of the equation of conservation of angular momentum can be written as

$$\rho \ddot{u}_i = \partial_j \sigma_{ij} + f_i, \quad \mathbf{x} \in \Omega, \quad t > 0$$

where $\rho$ is the density of the medium $\Omega$, $\mathbf{x}$ is the vector of spatial variables and $f_i$ is the $i^{th}$ component of the applied force. Einstein's convention for summation over repeated indices is assumed.

To obtain the weak form we apply the method of Galerkin to obtain

$$\int_\Omega \rho \ddot{u}_i v d\Omega + \int_\Omega \sigma_{ij}(\mathbf{u}) \partial_j v d\Omega = \int_\Omega f_i v d\Omega + \oint_\Gamma \sigma_{ij}(\mathbf{u}) v \hat{n}_j d\Gamma. \tag{7.1}$$

The domain is then split up into several subdomains over which the integral is summed

$$\int_{\Omega_k} \rho \ddot{u}_i v d\Omega_k + \int_{\Omega_k} \sigma_{ij}(\mathbf{u}) \partial_j v d\Omega_k = \int_{\Omega_k} f_i v d\Omega_k + \oint_{\Gamma_k} \sigma_{ij}(\mathbf{u}) v \hat{n}_j d\Gamma_k. \tag{7.2}$$

Removing the boundary term at the inter-element boundaries enforces and continuous-stress interface condition. Removing it the boundary of the modelling region enforces

a free-surface boundary condition allowing for the propagation of surface waves. At the artificial boundaries we impose the non-reflecting condition in [22, 21]. We omit the finer details of the element-wise discretization as they have been covered extensively in previous chapters.

## 7.4 Temporal-discretization

Implementing the spatial discretization results in the time-dependent system of equations for the nodal displacements in the $k^{th}$ element,$\mathbf{u}_i^k(t)$,

$$M^k\ddot{\mathbf{u}}_i^k(t) + A_i^k\dot{\mathbf{u}}_i^k(t) + \sum_j \hat{K}_{ij}^k\dot{\mathbf{u}}_i^k(t) + \sum_j K_{ij}^k\mathbf{u}_j^k(t) = M^k\mathbf{f}_i^k(t).$$

$M^k$ is the element mass matrix, $A_i^k$ is the damping matrix corresponding to the absorbing boundary conditions applied to the $i^{th}$ displacement, $\hat{K}_{ij}^k$ is the stiffness matrix associated with the viscoelastic term, $K_{ij}^k$ is the stiffness matrix associated with the elastic term and $\mathbf{f}_i^k(t)$ is the vector of applied nodal forces. Note that the matrices $A_i$ and $\hat{K}_{ij}^k$ have no overlapping entries and ultimately can be combined to form a single damping matrix that represents both the absorbing boundaries and the viscoelastic damping by editing the matrix $\hat{K}_{ij}^k$. For brevity we will simply denote this combined absorbing/damping matrix as $\hat{K}_{ij}^k$.

The global system is then assembled using the so-called connectivity matrix as defined in chapter 3. Where the viscoelastic case differs from the elastic case defined therein, is that the damping matrix is not diagonal and so the system cannot be

numerically time-stepped using central finite differences without having to solve a large system of equations at each time step. Thus we must reduce the order of the system by doubling the number of equations.

Let $\mathbf{U}$ be the vector of nodal displacements ordered vertically

$$\mathbf{U} = [u_1(x_1, z_1, t), ..., u_1(x_n, z_n, t), u_2(x_1, z_1, t), ..., u_2(x_n, z_n, t)]^T.$$

Similarly let $\mathbf{V}$ be the vector of nodal velocities. The system is then re-written as

$$\begin{pmatrix} M & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \dot{\mathbf{V}} \\ \dot{\mathbf{U}} \end{pmatrix} + \begin{pmatrix} \hat{K} & K \\ I & 0 \end{pmatrix} \begin{pmatrix} \mathbf{V} \\ \mathbf{U} \end{pmatrix} = \begin{pmatrix} \mathbf{F} \\ 0 \end{pmatrix}$$

We solve this by the $4^{th}$-order low-storage explicit Runge-Kutta method [7]. These methods have the advantage over standard Runge-Kutta methods of only requiring a single extra storage level, however, they must compute an additional intermediate step to update a single time-step. This is less of an issue than it may seem though as the resulting evolution equation is stable for larger time-steps than the standard methods.

## 7.5 Experimental results

We now present several numerical experiments illustrating the use of our procedure. The source in each is a Ricker wavelet in time with dominant frequency $w_0 = 30$ applied at a single node at $x = 500$,$z = 0$. A free surface condition is enforced at $z = 0$ and absorbing boundaries are placed at the sides and bottom.

First, we wish to show the high-frequency damping present in the viscoelastic model by purposefully choosing a grid too coarse to represent the source wavelet and then propagating the dispersed wavefield through the elastic part of the model to see how it is affected once it reaches the viscoelastic part. The model is 1000 meters by 1000 meters with $V_p = 2400$, $V_s = 1500$ and $\rho = 206$. From 0 to 250 meters the model is purely elastic with $Q_p = Q_s = \infty$, beyond that we add $Q_p = Q_s = 10$. We can see in figure 7.3 that as the source is propagated into the medium significant numerical dispersion is present. However, once the wave makes it to the viscoelastic region the high-frequency dispersion is damped considerably.

Next we wish to show the difference in the wavespeeds in elastic and viscoelastic wave propagation and the presence of reflections that result strictly from a change in $Q$. The elastic part of the model is the same as before. We compare the case of $Q_p = Q_s = \infty$, i.e a purely elastic medium, with $Q_s = 16$, and $Q_p = 24$. As we can see in figures 7.4 the wavelength in the viscoelastic media grows as the high-frequency portions of the wave is damped as it was in our first experiment.

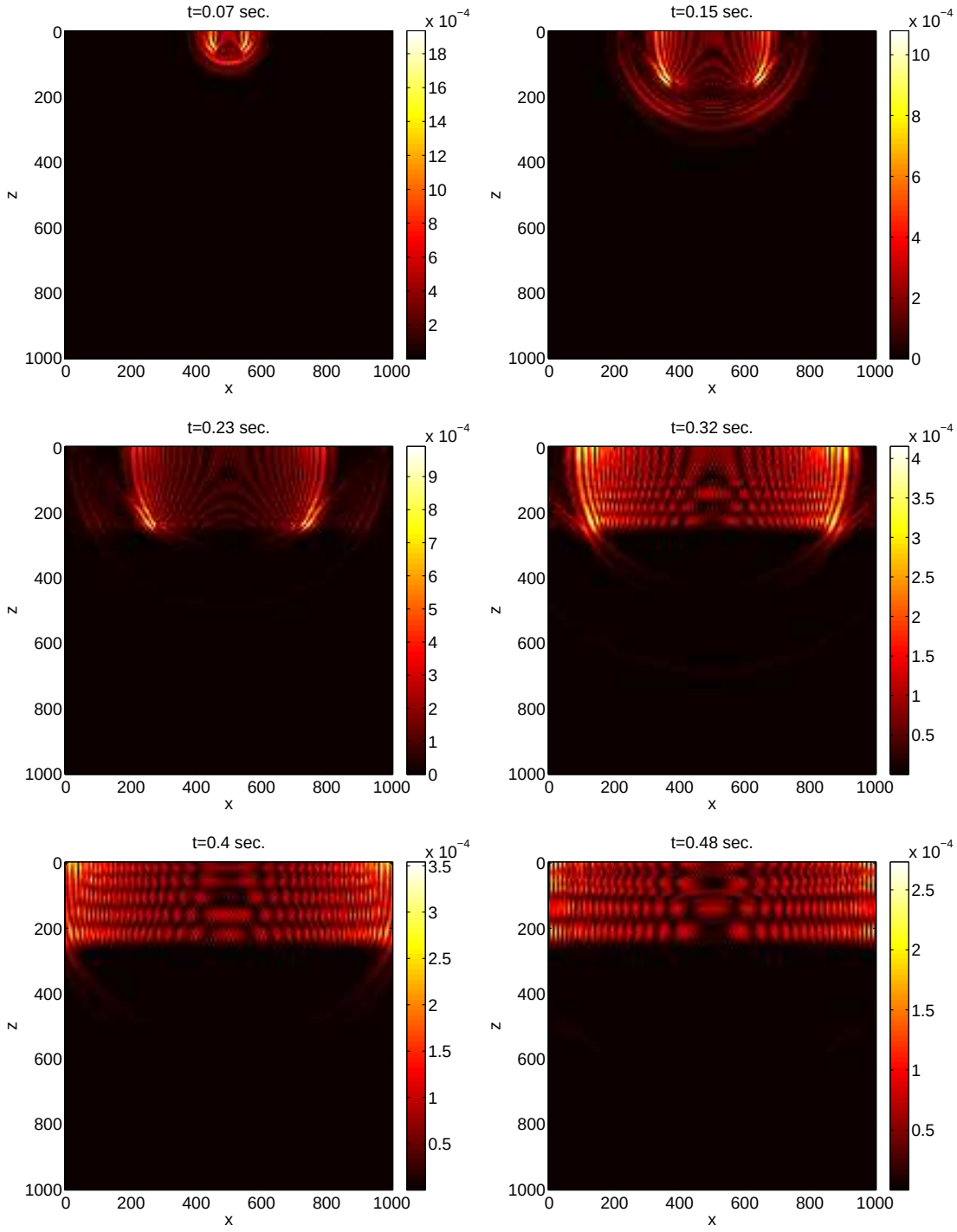



Figure 7.3: Numerical dispersion damped by viscoelastic media

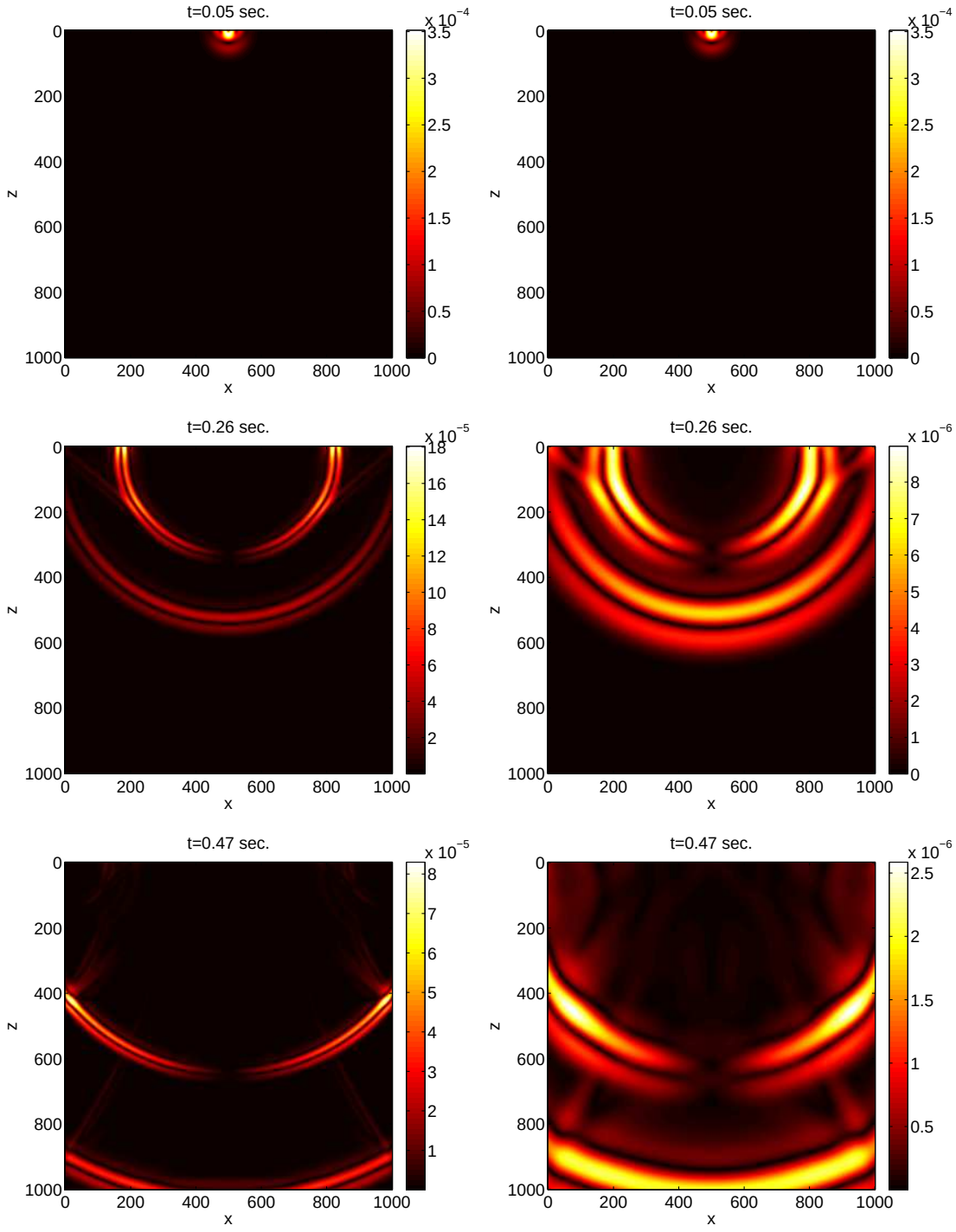



Figure 7.4: Elastic vs. viscoelastic wave propagation.

# Chapter 8

# Conclusion

## 8.1 Conclusion

The research showed that the study of pseudospectral-element method is a feasible and computationally efficient method for the numerical modelling of several types of seismic waves. Special attention was paid to the treatment of numerically imposed boundaries and discontinuous interfaces through the use of the weak form of the dynamic equilibrium equations. Further, it was shown that a significant amount of analytic work is required to build the equation used in the numerical procedure, but that the payoff for such work is a more efficient model that is capable of representing desirable properties such as free-surface and absorbing boundary conditions with less input from the end-user.

## 8.2 Future work

Preliminary work was done investigating the stability and character of the system matrices that appear in the spatially-discretized acoustic, elastic and anelastic equations using Embree and Trefethen's framework of $\varepsilon - pseudospectra$. Their idea is

to generalize the idea of the spectrum of an operator, to the set of points that are within some $\varepsilon$ of the actual spectrum. Because of the delicate nature of pseudospectral differentiation matrices, when operators are discretized using these as their base the operators exhibit the same type of delicate response to numerical precision. This led Embree and Trefethen to re-frame the language of the stability of a numerical time stepping algorithm in terms of $\varepsilon - pseudospectra$, postulating that for a time-stepping equation to be stable, not only must the eigenvalues of the system lie within the unit-circle, but the pseudospectra must be well-behaved. Their book [23] serves as an anthology of all their work in spectra and pseudospectra and includes several examples of cases where the usual eigenvalue analysis fails and pseudospectra correctly predict the instabilities. Apart from the more useful applications of pseudospectra, when working with them enough one develops a sense of the character of an operator and an intuition of how they will act. They are also quite beautiful visually as can be seen in figure 8.1. It would be interesting to attempt to use pseudospectra as a way to investigate the stability of the numerical methods as different types of absorbing boundary conditions are applied. Many can cause strange instabilities that seem to have nothing to do with the eigenvalues of the matrix for a large number of degrees of freedom. Because it is expensive to compute pseudospectra however, this was not something that was explored. There is also a fair bit of new analytic material coming out about pseudospectra that could be used to develop

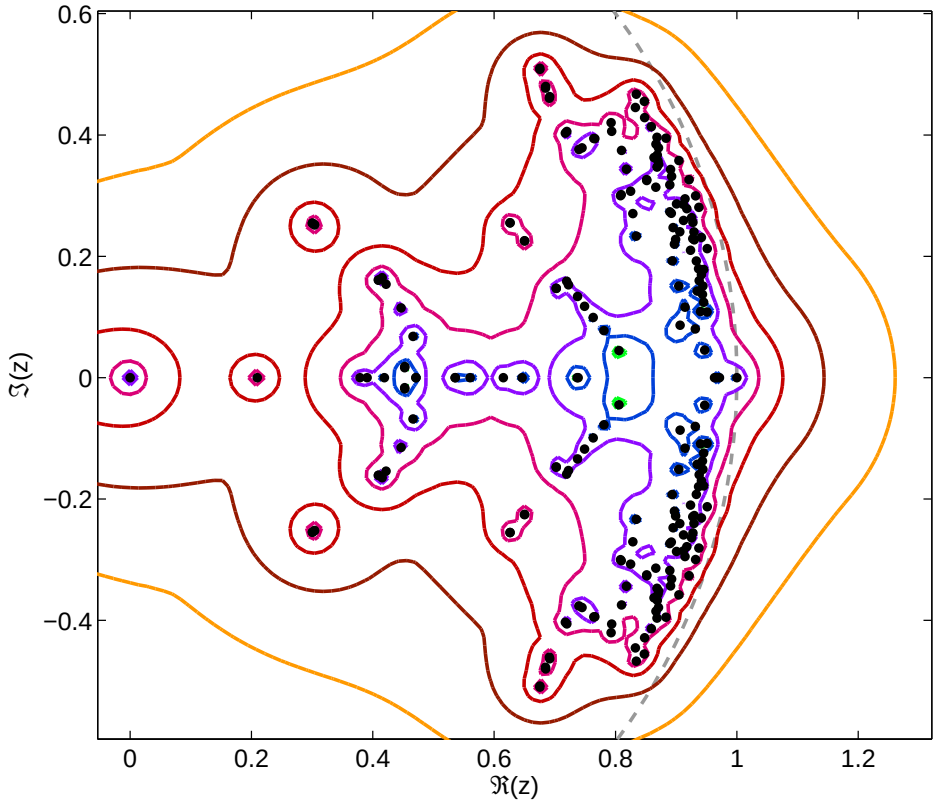



Figure 8.1: Eigenvalues and pseudospectra of the discretized spatial part of the elastic operator.

more robust stability criteria.

In terms of the numerical implementation of the methods in this research, some preliminary work was done with 3D modelling but not enough to warrant inclusion. Some research level software packages do exist for 3D pseudospectral-element modelling, but are more geared towards global-scale seismic modelling (see SPECFEM3D available at http://www.geodynamics.org/cig/software/specfem3d for instance). It would be interesting to attempt to build attempt to replicate three-component three-

dimensional seismic acquisition for a 2D line by restricting one of the horizontal dimensions to something small and then seeing if the absorbing boundary conditions properly handle the reflections from the relatively near boundaries. Another approach would be to assume a model that was constant in one spatial dimension and then analytically integrate the equations along that dimension to produce a 2.5D model. Both of these approaches could then be compared to a full 3D model and see if the results along the surface are similar. If so, this would lead to more accurate amplitudes in the modelled data and increase the computational efficiency drastically.

# Bibliography

[1] K. E. Atkinson and W. Han. *Theoretical Numerical Analysis: A Functional Analysis Framework*. Springer, 2009.

[2] I. Babuska and B.Q. Guo. The h, p and h-p version of the finite element method: basis theory and applications. *Advances in Engineering Software*, 1992.

[3] L. Bednath and T. Myint-U. *Linear Partial Differential Equations for Scientists And Engineers*. Birkhauser, 2007.

[4] J. M. Carcione. *Wave Fields in Real Media - Wave Propagation in Anisotropic, Anelastic and Porous Media*. Elsevier, 2001.

[5] J.M. Carcione, F. Poletto, and D Gei. 3-D wave simulation in anelastic media using the Kelvin-Voigt constitutive equation. *J. Comput. Phys.*, 196:282–297, May 2004.

[6] R. Clayton and B Engquist. Absorbing boundary conditions for acoustic and elastic wave equations. *Bulletin of the Seismological Society of America*, 1977.

[7] David and Ketcheson. Runge Kutta methods with minimum storage implementations. *Journal of Computational Physics*, 229(5):1763 – 1773, 2010.

[8] E. Faccioli, F. Maggio, R. Paolucci, and A. Quarteroni. 2d and 3d elastic wave

propagation by a pseudo-spectral domain decomposition method. *Journal of Seismology*, 1:237–251, 1997. 10.1023/A:1009758820546.

[9] E. Faccioli, F. Maggio, A. Quarteroni, and A. Tagliani. Spectral-domain decomposition methods for the solution of acoustic and elastic wave equations. *Geophysics*, 1996.

[10] E. Faccioli and A. Quarteroni. Comment on "The spectral element method: An efficient tool to simulate the seismic response of 2D and 3D geological structures". *Bulletin of the Seismological Society of America*, 1999.

[11] B Fornberg. *A Practical Guide to Pseudospectral Methods*. Cambridge University Press, 1984.

[12] D. Funaro. *Polynomial Approximation of Differential Equations*. Springer-Verlag, 1992.

[13] J. S. Hestaven and T. Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis and Applications*. Springer, 2008.

[14] R. L. Higdon. Absorbing boundary conditions for acoustic and elastic waves in stratified media. *Journal of Computational Physics*, 101(2):386 – 418, 1992.

[15] C. O. Horgan. Korn's inequalities and their applications in continuum mechanics. *SIAM Rev.*, 37:491–511, December 1995.

[16] D. Komatitsch and J.P. Vilotte. The spectral element method: An efficient tool to simulate the seismic response of 2d and 3d geological structures. *Bulletin of the Seismological Society of America*, 1998.

[17] Seymour V. Parter. On the Legendre Gauss Lobatto points and weights. *J. Sci. Comput.*, 14:347–355, December 1999.

[18] A. Quarteroni, A. Tagliani, and E. Zampieri. Generalized galerkin approximations of elastic waves with absorbing boundary conditions. *Computer Methods in Applied Mechanics and Engineering*, 163(1-4):323 – 341, 1998.

[19] J. Reddy. *An Introduction to the Finite Element Method*. McGraw Hill, 2005.

[20] G. S. Sarma, K. Mallick, and V. R. Gadhinglajkar. Nonreflecting boundary condition in finite-element formulation for an elastic wave equation. *Geophysics*, 63(3):1006–1016, 1998.

[21] J. Sochacki. Absorbing boundary conditions for the elastic wave equations. *Applied Mathematics and Computation*, 28(1):1 – 14, 1988.

[22] R. Stacey. Improved transparent boundary formations for the elastic-wave equation. *Bulletin of the Seismological Society of America*, 78:2089–2097, December 1988.

[23] L.N. Trefethen and M. Embree. *Spectra and Pseudospectra: The Behavior of*

*Nonnormal Matrices and Operators*. Princeton University Press, 2005.

[24] F. Vasheghani and L.R. Lines. Viscosity and Q in heavy-oil reservoir characterization. *The Leading Edge*, July:856–860, 2009.

[25] J.P. Vilotte and D. Komatitsch. Reply to comment by E. Faccioli and A. Quarteroni on "The spectral element method: An efficient tool to simulate the seismic response of 2D and 3D geological structures". *Bulletin of the Seismological Society of America*, 1999.

[26] T. Warburton. An explicit construction of interpolation nodes on the simplex. *Journal of Engineering Mathematics*, 56:247–262, 2006.

[27] J. A. Weideman and S. C. Reddy. A matlab differentiation matrix suite. *ACM Trans. Math. Softw.*, 26:465–519, December 2000.