SIMULATING WAVE SCATTERING PROBLEMS BY PSEUDOSPECTRAL TIME-MARCHING ON SUPERCOMPUTERS

by

YONG LUO

B. Sc. (Electronic Engineering), Fudan University, 1987

M. A. Sc. (Electronic Engineering), The University of Electronic Sci. & Tech. of China, 1990

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

THE DEPARTMENT OF ELECTRICAL ENGINEERING

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

Sept. 1994

© Yong Luo, 1994

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

(Signature)

Department of Electrical Engineering

The University of British Columbia
Vancouver, Canada

Date Dec. 22, 1994

DE-6 (2/88)

Abstract

In this thesis, a new time-marching scheme for time-dependent PDEs with periodic and non-periodic boundary conditions is introduced and implemented. The new time-marching scheme is based on the polynomial interpolation of the symbolic solution of the original PDEs. The approximation of the space derivatives is performed by pseudospectral methods. The marching of the solution in the time domain is done by the polynomial expansion or the Newton-form polynomial interpolation, depending on the properties of the space derivatives. The boundary conditions are properly represented by a suitable pseudospectral approximation and some technical manipulations of the collocated operators. This technique can efficiently provide balanced spectral accuracy in both the space and time dimensions. The numerical stability and resolution are also improved by the new polynomial time-marching scheme. In the periodic boundary case, the spatial approximation generally can be done by Fourier collocation and the time-marching sometimes can be easily implemented by Chebyshev polynomial series. In the non-periodic case, the spatial operator should be approximated by a Chebyshev collocation, which can include different non-periodic boundary conditions through careful manipulations of the boundary conditions. In this case and the complicated periodic case, the time-marching has to rely on the more general Newton-form interpolation based on Fejḗr points.

Based on the new polynomial time-marching scheme, a two-dimensional, SH-seismic reflection model is simulated by full implementation of the new time-marching scheme with the approximated absorbing boundary conditions. Some scattering phenomena such as diffraction are illustrated through the visualization.

The simulation of the physical model is accomplished on two supercomputers: the TMC Connection Machine CM-5 and the Fujitsu VPX240/10. The parallel programming (CM-Fortran on CM-5 and Fortran 77/VP on the VPX240/10) and optimization issues on the two supercomputers are also discussed in this thesis.

TABLE OF CONTENTS

List of Tables

List of Figures

Acknowledgments

Dedicated to my parents

I would like to take this opportunity to express a much deserved "thank you" to my knowledgeable and supportive thesis supervisor, Dr. Matthew J. Yedlin, for his two-fisted assistance in helping me turn a vague idea into a finished product. His good connections made some state-of-the-art computing facilities (CM-2 and CM-5) available to this thesis work.

My sincerest thanks to Dr. Ian Cumming, Dr. Brian Seymour and Dr. E. V. Jull, in recognition of helpfulness and partial financial support, which made the accomplishment of this thesis work possible.

I gratefully appreciate the insightful and valuable discussions with Dr. Shenkai Zhao in Geophysics and Mr. Pin Lin in Mathematics, from both of whom I've been benefited so much on some complicated mathematical topics. Special thanks also to Dr. Peter Cahoon in Computer Science for his assistance in helping me make the data visualization tape which is part of this thesis.

Finally, a heartfelt note of thanks to my wonderful parents, who have supported me in every manner possible. Without their endless love, support and care, this would never have been possible. I am indebted to my dearest parents. At this moment, please allow me to say:

To the Love and Devotion of My Parents !

可怜天下父母心 ！

Chapter 1 Introduction

Accurately and efficiently solving many time-dependent Partial Differential Equations (PDEs) by numerical methods enables researchers to study some complex physical phenomena in detail. With the help of the quickly growing computational ability of some supercomputers, accurate simulation and visualization can be performed for many wave-field problems. On the latest supercomputers, researchers now can perform accurate time-dependent simulations in three dimensions and study phenomena they only dreamed of studying a decade ago. Increasing demands in the numerical simulation field and the availability of rapidly increasing computing power have greatly pushed and also challenged some traditional numerical algorithms.

1.1 Numerical Methods For Time-dependent PDEs

Major numerical algorithm candidates in the numerical simulation field are the Finite Difference (FD), Finite Element (FE) and spectral methods (including pseudospectral or collocation). Particularly, spectral algorithms are becoming more popular with more accurate simulation demands and the greater computing power of the supercomputers.

The solution of partial differential equations (PDEs) by pseudospectral methods has been extensively studied and comprehensively applied in various numerical analysis application areas such as Computational Fluid Dynamics (CFD), geophysics, oceanography, mechanics and electromagnetic, etc. Pseudospectral methods (or the spectral collocation methods) are characterized by the expansion of the solutions in terms of global basis functions, where the expansion coefficients are computed so that the differential equation is satisfied exactly at a set of so-called collocation (or interpolation) points. The popularity of these methods arises from several advantages in computation efficiency and much higher accuracy over common finite difference methods. [24, 5]. Conventional pseudospectral methods only deal with the approximation of the derivatives in space.

For time evolution of the solutions, so far the various finite difference time-marching techniques still play a dominant role in actual applications.

1.2 Description of the Polynomial Time-marching Method

A unique technique for time-marching has been proposed by Tal-Ezer since 1986[39, 40, 41]. This technique is based on the polynomial approximation theory of complex matrix polynomials and has arbitrary order accuracy for the time-marching approximation. Furthermore, this technique actually evaluates time evolution by polynomials of the derivatives in space. Tal-Ezer's original algorithm is based on Fourier pseudospectral space approximation. Therefore, it can only be applied to those problems with periodic boundaries in space. However, his Newton-form polynomial interpolation time-marching scheme can be incorporated into many other space descretization methods such as Chebyshev pseudospectral or finite-difference. This thesis extends the Newton-form interpolation time-marching method to Chebyshev pseudospectral space descretization incorporated with non-periodic and non-reflecting boundary conditions. With this incorporation, some experimental calculations have been performed on the one-dimensional wave equation in constant medium with homogeneous Dirichlet/Neumann/absorbing boundary conditions. The results show that the overall error converges to zero at a rate faster than some fixed orders of finite-difference error convergence, but not at an exponential rate.

1.3 Thesis Outline

In Chapter 2 of this thesis, mathematical principles, accuracy, convergence, stability, boundary-condition considerations and computational requirements of conventional pseudospectral methods for the PDEs are briefly described. Domain decomposition techniques related to the use of pseudospectral methods for elliptic equations are also introduced. Tal-Ezer's time-marching technique is introduced in Chapter 3. Chebyshev polynomial expansion, Newton-form polynomial interpolation and Faber polynomial expansion are then described. Their advantages and weaknesses are also investigated. Chapter 4 introduces a new polynomial time-marching technique for non-periodic boundary value problems. Stability improvement and computing efficiency are emphasized. Chapter 5 extends the technique in Chapter 4 to time-dependent boundary cases: non-reflecting boundary approximations. Chapter 6 gives a general analysis of applying the foregoing techniques to simulate an SH-wave seismic reflection model. Chapter 7 focuses on the brief introduction of the supercomputing implementation, including the Connection Machine CM-5 architecture, CM-Fortran parallel programming issues and the Fujitsu VPX 240/10 issues. The last chapter, summary chapter, summarizes the motivation, research methods and achievements of this thesis.

The major contribution of this thesis work is incorporating the non-periodic and even non-reflecting boundary conditions into the original polynomial time-marching scheme. Thus, the applicability of the polynomial time-marching has been greatly expanded.

Chapter 2 Pseudospectral Methods for PDEs

The success of spectral methods in practical computation has led to an increasing interest in their theoretical aspects, especially since the mid-1970s. Spectral methods generally can be viewed as an extreme development of the class of discretization schemes for differential equations known generically as the method of weighted residuals (MWR). The key elements of the MWR are trial functions (also called the expansion or approximating functions) and test functions (also known as weight functions). Choice of the trial functions is one of the features which distinguish spectral methods from finite-element and finite-difference methods. Choice of the test functions distinguishes between the three most commonly used spectral schemes: the Galerkin, pseudospectral or collocation, and tau methods. In this chapter, the basic theoretical principles will be introduced and focussed mainly on the pseudospectral ones, through comparison with the FD, FE and the other spectral schemes.

2.1 Introduction of Pseudospectral Methods

The general idea of spectral methods is based on approximating *U(x)*, the solution of a PDE by a sum of N+1 "basis functions" $\phi_n(x)$:[24]

$$U(x) \approx U_N(x) = \sum_{n=0}^{N} a_n \phi_n(x) \tag{2.1.1}$$

The second step is to obtain equations for the discrete values $U_N(x)$ or the coefficients a_n from the original equation. In the case of a differential equation, this second step involves finding an approximation for a differential operator in terms of the grid point values of $U_N(x)$ or , equivalently, the expansion coefficients. The goal is to choose the series coefficients $\{a_n\}$ in such a way that the so-called residual function is made as small as possible. For the equation

$$\mathcal{L}U = f(x) \tag{2.1.2}$$

where $\mathcal{L}$ is the operator of a differential equation, the residual function is defined by

$$R(x; a_0, a_1, \cdots, a_N) = \mathcal{L}U_N - f \tag{2.1.3}$$

The different spectral and pseudospectral methods differ mainly in their ways of minimizing $R(x; a_0, a_1, \cdots, a_N)$. In this sense, the general spectral methods fall into two broad categories: the interpolating or pseudospectral, and the non-interpolating (or sometimes directly called spectral method). The pseudospectral methods associate a grid of points with each basis set. The coefficients of a known function *f(x)* are found by requiring that the truncated series agrees with *f(x)* at each point of the grid. The solution of a differential equation is found by demanding that the coefficients a_n be such that the residual function has the property

$$R(x_i; a_0, a_1, \cdots, a_N) = 0, \;\; i = 0, 1, \cdots, N \tag{2.1.4}$$

Presumably, as $R(x;a_n)$ is forced to vanish at an increasingly large number of discrete points known as "collocation" or "interpolation" points, it will be smaller and smaller in the gaps between the collocation points so that $U_N(x)$ will converge to U(x) as N increases. Methods in this category are also called "orthogonal collocation" or "method of selected points".

The "non-interpolating" methods include Galerkin's method and the Lanczos tau method which were developed long before the pseudospectral methods. These algorithms are usually labelled as spectral methods. There is no grid of interpolating points associated with these procedures. Instead, the coefficients of a known function f(x) are computed by multiplying f(x) by a given basis function and integrating (projection method). The pseudospectral methods are equivalent to the spectral methods if one evaluates the integrals of the latter algorithm by means of numerical quadrature with (N+1) points [24, 8].

In [24], [8], it has been shown that the accuracy of the pseudospectral methods is only a little bit poorer than that of the non-interpolating ones — too little to outweigh the much greater simplicity and computational efficiency of the pseudospectral algorithms. This approach is especially attractive because of the ease with which it can be applied to variable-coefficient and non-linear problems. Consequently, this thesis will focus on the pseudospectral techniques for applications involving variable coefficients.

The so-called "finite-element" methods are similar in philosophy to the spectral algorithms. The major difference is that the finite-element (FE) techniques chop the interval in x into a number of sub-intervals, and choose the $\phi_n(x)$ to be local functions which are polynomials of low, fixed degree and non-zero only over neighboring sub-intervals. In contrast, the spectral methods use global basis functions in which each basis function is a polynomial of high degree and non-zero, except for isolated points, over the entire computational domain. To increase the accuracy, the finite element methods chop the interval into smaller pieces without changing the degree of the polynomials in the basis. In contrast, the spectral methods raise the highest degree of the polynomials without subdividing the interval any more. The FE methods convert differential equations into matrix equations that are sparse. The spectral methods generate algebraic equations with full matrices. As compensation, the high order of the basis functions gives high accuracy for a given N. To model problems, when fast iterative matrix-solvers are used, the spectral methods can be much more efficient than FE methods.

For comparison between the pseudospectral methods and the finite difference (FD) methods, Fornberg has given more details in [14, 15]. He showed that the pseudospectral methods can be viewed as a limit of finite differences of increasing orders. In each space dimension, the pseudospectral methods require as little as a quarter the number of grid points compared to a fourth-order finite difference scheme for the same accuracy requirements. Here, some simple and heuristic explanations for the accuracy of the pseudospectral methods are addressed. More detailed discussions can be found in [26, 24, 5, 20, 8, 14].

A standard fourth-order finite difference approximation to the 1st-order spatial derivative can be written as:

$$\frac{\partial f}{\partial x} \approx [-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)]/12h + O(h^4) \qquad (2.1.5)$$

where $h=1/N$ is the grid interval. Since each spectral coefficient is determined by all the grid point values of f(x) for the N-point pseudospectral approximation, it follows that the pseudospectral rules are an N-point differencing formula with an error of $O(h^N)$.

As N is increased, the pseudospectral method benefits in two ways: the grid interval h becomes smaller, and the order of the approximation degree increases. This is decreasing faster than any finite power of N because the power in the error formula is increasing , too. This is the so-called "infinite order" or "exponential" convergence, as J.P. Boyd noted in [5]. Specifically, this "exponential" convergence property is useful in solving linear elliptic problems with smoothly variable coefficients. The property of "exponential" convergence to hyperbolic equations only holds for sample model problems discussed in Section 5.3.

Choosing a proper basis function set is the first step of the pseudospectral approximation. This is usually decided by the boundary conditions of the problem. The boundary conditions may be divided into two broad categories: behavioral and numerical. It is almost always necessary to impose explicit numerical boundary conditions upon the approximate solution. In contrast, behavioral conditions may be satisfied implicitly by choosing basis functions that have the required property or behavior. Based on these considerations and the properties of different basis function sets, the proper basis set can be chosen as follows:

Function Behavior Over The Interval	Basis Set
f(x) is periodic & symmetric or antisymmetric about x=0	Fourier Series
$x \in [a,b]$ $\& f(x)$ is non-periodic	Chebyshev Polys; Legendre Polys.
$x \in [0,\infty]$ $\& f(x)$ decays exponentially as $x \to \infty$	TL_n(x) (rational Chebyshev); Legendre Polys.
$x \in [0,\infty]$ $\& f(x)$ has asymptotic series in inverse powers of x	TL_n only
$x \in [-\infty,\infty]$ $\& f(x)$ decays exponentially as $x \to \infty$	*sinc* functions or Hermite functions
$x \in [-\infty,\infty]$ $\& f(x)$ has asymptotic series in inverse powers of x	TB_n only
f(λ,θ) is a function of latitude and longtitude on a sphere	Spherical harmonics

Table 2.1 The Choices of Basis Functions

In most actual applications, the Fourier series and the Chebyshev polynomials are the best choices, not only because Fast Fourier Transform (FFT) can be used to compute these two bases, but also because most physical problems can be modelled as either periodic boundary value problems or finite interval boundary value problems. Consequently, all discussions and analysis will be concerned with the Fourier and the Chebyshev pseudospectral methods.

2.2 Pseudospectral Fourier Method

In the pseudospectral Fourier approximation, the problem involves periodic boundary conditions. Thus, all functions appearing in the problem are periodic. Suppose f(x) to be a smoothly differentiable function with period 2π. It can be approximated by the trigonometric polynomial $P_N f(x)$ that interpolates it at a set of collocation points. The following is one of the possible choices of the points [20]:

$$x_j = \frac{\pi j}{N} \; (j = 0, \cdots, 2N-1) \tag{2.2.1}$$

Then the approximation $P_N f(x)$ has the form

$$P_N f(x) = \sum_{j=0}^{2N-1} f(x_j) g_j(x) \tag{2.2.2}$$

$$\text{where } g_j(x_k) = \delta_{jk}, \text{ and}$$

$$g_j(x) = \frac{1}{2N} \sum_{l=-N}^{N} \frac{1}{c_l} e^{il(x-x_j)}$$

$$c_l = 1 \; (|l| \neq N), \; c_N = c_{-N} = 2$$

Defining

$$a_l = \frac{1}{2Nc_l} \sum_{j=0}^{2N-1} f(x_j) e^{-ilx_j} \;, \tag{2.2.3}$$

then (2.2.2) becomes

$$P_N f(x) = \sum_{l=-N}^{N} a_l e^{ilx} \tag{2.2.4}$$

The next step in the pseudospectral method is to seek equations for an approximate solution U_N to a differential equation whose exact solution is U. Indeed, (2.2.4) gives

$$\frac{\partial^k U_N(x_j)}{\partial x^k} = \sum_{|n| \leq N} (in)^k a_n e^{inx_j} \;, \tag{2.2.5}$$

where a_n is given by (2.2.3). If N is a highly composite integer, the Discrete Fourier Transforms (DFTs) (2.2.3), (2.2.5) can be efficiently evaluated by the FFT algorithm in $O(N \log_2 N)$ operations. Thus, evaluation of derivatives requires just two FFTs together with the complex multiplications by $(in)^k$ in (2.2.5).

Expressing the same procedure in matrix form, one obtains

$$\frac{\partial^k U_n(x_j)}{\partial x^k} = \left(D_k \vec{U}\right)_j \tag{2.2.6}$$

where D_k is a $2N \times 2N$ matrix, and

$\vec{U}$ is a column vector $\vec{U} = \begin{pmatrix} U_N(x_0) \\ \vdots \\ U_N(x_{2N-1}) \end{pmatrix}$

Explicitly,

$$(D_1)_{jn} = \begin{cases} \frac{1}{2}(-1)^{j+n} \cot\left(\frac{x_j - x_n}{2}\right) & j \neq n \\ 0 & j = n \end{cases} \tag{2.2.7}$$

$$D_k = (D_1)^k$$

Obviously, D_1 is a real, antisymmetric matrix. Consequently, D_{2k} is a real, symmetric matrix while D_{2k+1} is a real, antisymmetric matrix.

To show the accuracy of the Fourier pseudospectral method, an approximation theory result has been given in [9] : Let p>1/2. Then for *$0 \leq q \leq p$,* there exists a constant C independent of N and such that

$$\|U - P_N U\|_q \leq C N^{q-p} |U|_p$$

Here the norm $\|\ \|$ is defined as:

$$\|U\|_q^2 = \sum_{j=0}^{q} |u|_j^2 \text{ , where } |U|_q^2 = \sum_{k=-\infty}^{\infty} k^{2q} |a_k|^2$$

It measures the magnitude of U and its first q derivatives. Obviously, the rate error decrease with increasing N only depends on the smoothness of the function being approximated. However, if some smoothing procedure is used, the pseudospectral method is still applicable to pointwise smooth problems[20].

It is necessary to review some results regarding the stability and convergence of the Fourier pseudospectral method. A scheme is said to be stable if

$$\|U_N(t)\| \leq C\|U_N(0)\| \text{ , } (0 \leq t \leq T) \tag{2.2.8}$$

for some proper norm. Here C may depend on T, but not on N. By the energy estimate method, Gotllieb *et. al.*[20] and Hussaini *et. al.*[24] have demonstrated the stability of the pseudospectral method for three major types of partial differential equations. The exception is the hyperbolic equation with zero order derivative (damping term):

$$U_t + a(x)U_x + bU = 0 \tag{2.2.9}$$

The straightforward discretization for this equation is often unstable. The Fourier method is stable if a(x) is of fixed sign. Otherwise only some typical cases have been proven stable. Usually, writing the spatial derivative in skew-symmetric form, or filtering the solution results in a practical and stable approximation.

As for the accuracy and convergence of Fourier pseudospectral method in elastic wave equations, Fornberg [14, 15] has indicated analytically that the Fourier pseudospectral method can be interpreted as a limit of finite differences of increasing orders. Based on spectral analysis of a model equation, he showed that the pseudospectral method (integration in time) may require, in each space dimension, as little as a quarter the number of grid points compared to a fourth-order finite-difference scheme and one-sixteenth the number of points as a second-order finite-difference scheme. For the total number of points in two dimensions, these factors become 1/16 and 1/256, respectively. In series of his test calculations on the two-dimensional elastic wave equation, only minor degradations are found in cases with variable coefficients and discontinuous interfaces. He also noted in [14] that in the acoustic case, the governing equations take a special form which can be exploited by low-order finite-difference methods. The advantages of the pseudospectral method are then less than in the more general elastic case.

2.3 Pseudospectral Chebyshev Method

When a function f(x) is not periodic, or the computational domain has some numerical boundary conditions, a trigonometric interpolation polynomial does not provide a good enough approximation to yield accurate approximations to the derivatives of f(x).

It is better to approximate f(x) by polynomials in x. However, it is well known that the *Lagrange* interpolation polynomial based on equally spaced points does not give a satisfactory approximation to a general smooth function f. This poor behavior of polynomial interpolation can be avoided for smoothly differentiable functions by removing the restriction to equally spaced collocation points. In the most common pseudospectral Chebyshev method, the interpolation points in the interval [-1,1] are chosen to be one of the extrema of the Nth order Chebyshev polynomials $T_N(x)$. These are :

$$\textit{Gauss} - \textit{Lobatto:}\ x_j = \cos\frac{\pi j}{N},\quad (j = 0, \cdots, N) \tag{2.3.1}$$

$$\textit{Gauss} - \textit{Radau:}\ x_j = \cos\frac{2\pi j}{2N+1},\quad (j = 0, \cdots, N) \tag{2.3.2}$$

$$\textit{Gauss:}\ x_j = \cos\frac{(2j+1)\pi}{2N+1},\quad (j = 0, \cdots, N) \tag{2.3.3}$$

Following the definition of the Chebyshev polynomial

$$T_n(x) = \cos\left(n\cos^{-1}x\right) \tag{2.3.4}$$

in the *Gauss-Lobatto* grid point case, one obtains

$$T_n(x_j) = \cos\frac{\pi j n}{N} \tag{2.3.5}$$

which indicates a close relation between the pseudospectral Chebyshev and the pseudospectral Fourier method. The Nth degree interpolation polynomial $P_N f(x)$ to f(x) is given by

$$P_N f(x) = \sum_{n=0}^{N} a_n T_n(x) \tag{2.3.6}$$

$$\text{where } a_n = \frac{2}{N}\frac{1}{c_n}\sum_{j=0}^{N}\frac{f(x_j)T_n(x_j)}{c_j}$$

$$c_0 = c_n = 2,\ c_j = 1,\ (1 \le j \le N-1)$$

An expression for the derivative of $P_N f(x)$ can be obtained by differentiating the above equation:

$$\frac{\partial P_N f}{\partial x} = \sum_{n=0}^{N} a_n T_n'(x) = \sum_{n=0}^{N} b_n T_n(x) \tag{2.3.7}$$

where $b_N = 0,\ b_{N-1} = 2Na_N$ and

$c_n = b_{n+2} + 2(n+1)a_{n+1} \quad (0 \le n \le N-2)$

More generally,

$$\frac{\partial^p (P_N f(x_k))}{\partial x^p} = \sum_{j=0}^{N} f(x_j)(D_p)_{kj} \tag{2.3.8}$$

where $(D_1)_{kj} = \dfrac{c_k}{c_j}\dfrac{(-1)^{j+k}}{x_k - x_j} \quad (k \ne j)$

$(D_1)_{jj} = -\dfrac{x_j}{2\left(1 - x_j^2\right)}\ ,\ (D_1)_{00} = \dfrac{2N^2+1}{6} = -(D_1)_{NN}$

and $D_p = (D_1)^p$

It should be noted that the matrix D_1 is not antisymmetric; also D_2 is not symmetric.

To consider the stability and convergence of the pseudospectral Chebyshev method, the following norm definitions are introduced:

$$||f||_q^2 = \left[||f||^2 + \left\| \frac{\partial f}{\partial x} \right\|^2 + \cdots + \left\| \frac{\partial^q f}{\partial x^q} \right\|^2 \right] \tag{2.3.9}$$

$$\text{where } ||f||^2 = \int_{-1}^{1} \frac{f^2(x)}{\sqrt{1-x^2}} dx$$

By these norm definitions, Gotllieb *et. al.* have summarized the main approximation results of the pseudospectral Chebyshev method in [20].

Let f(x) be a function with S continuous derivatives and let $P_N f$ be defined as in (2.3.6), then there is a constant C independent of f(x) and N such that

$$\|f - P_N f\|_q \leq C N^{2q-s} \|f\|_s \left(0 \leq q \leq \frac{s}{2}\right) \tag{2.3.10}$$

The accuracy of this approximation still depends on the smoothness of f(x). Similarly, one can obtain a good approximation for the derivative "far away" from the discontinuity by somehow smoothing the function. In [24, 20], it is demonstrated by the energy estimate method that the pseudospectral Chebyshev method has similar stability properties for the three major PDEs as the Fourier method. The hyperbolic equation is still a problem when the wave velocity changes sign. Fortunately, in physical applications, wave velocities are always strictly positive.

2.4 Pseudospectral Domain Decomposition Techniques

In order to avoid the need for global mappings required by spectral methods in problems with complicated geometries, domain decomposition techniques have been developed for elliptic equations. They are also effective methods for overcoming some non-smoothness or jumping conditions in global domains. A complicated domain can be subdivided into several subdomains and individual spectral methods can be applied to each subdomain. These techniques seems naturally suited for the parallel computer architectures except at the subdomain interfaces. Gottlieb *et. al.* [19] investigated in depth the influence of interface boundary conditions on the possibility of parallelizing pseudospectral multidomain algorithms.

The subdomains are connected by "PATCHING" if the solutions in different elements are matched along their common boundary by requiring that U(x) and a finite number of its derivatives are equal along the interface [5]. Since the explicit numerical boundary conditions must be used, the Chebyshev method is the most common choice.

The number of matching conditions is the number that is necessary to give a unique solution in each subdomain. Thus, one must match both $U(x)$ and $\frac{\partial U}{\partial x}$ at the interface for a second order differential equation. For elliptic equations or other steady state equations, patching is usually straightforward. Boyd [5] has given a very efficient way to generate the weak coupling of element solutions in the following example:

$$\text{Equation } \mathcal{L}u = f(x) \tag{2.4.1}$$

$$u(a) = \alpha,\ u(b) = \beta$$

$$\mathcal{L} = q_2(x)\frac{\partial^2}{\partial x^2} + q_1(x)\frac{\partial}{\partial x} + q_0(x)$$

Instead of just two subintervals, divide the interval $x \in [a, b]$ into M subdomains. Let d_j denote the boundary between element (j-1) and element j and define:

$$U_j \equiv u(d_j)$$

The solution on the *j*th element, $U_j(x)$, can always be written as the sum of a particular integral $P_j(x)$ plus two homogeneous solutions, defined by:

$$\mathcal{L}P_j(x) = f \text{ and } P_j(d_{j-1}) = P(d_j) = 0 \tag{2.4.2}$$

$$\mathcal{L}h_{Lj} = 0 \text{ and } h_{Lj}(d_{j-1}) = 1,\ h_{Lj}(d_j) = 0 \tag{2.4.3}$$

$$\mathcal{L}h_{Rj} = 0 \text{ and } h_{Rj}(d_{j-1}) = 0,\ h_{Rj}(d_j) = 1$$

It follows that the general solution to the differential equation can be written as :

$$u_j(x) = P_j(x) + U_{j-1}h_{Lj}(x) + U_j h_{Rj}(x) \tag{2.4.4}$$

Thus, all particular integrals and homogeneous solutions can be computed independently of the solutions on all the other elements. The boundary value problems defined by (2.4.2), (2.4.3) are completely uncoupled.

The final element solutions $u_j(x)$ should be determined by (2.4.2), (2.4.3) plus the following continuity condition of the first derivative at each of the subdomain interfaces. (The (2.4.2), (2.4.3) only consider the $u(x)$ continuity)

$$h'_{Lj}U_{j-1} + \left(h'_{Rj} - h'_{L,j+1}\right)U_j - h'_{R,j+1}U_{j+1} = P'_{j+1} - P'_j \qquad (2.4.5)$$
$$j = 1, \cdots, M-1$$
$$U_0 = \alpha,\ U_M = \beta$$

The matrix expressed above should be tridiagonal because only four different homogeneous solutions, two for element (j-1) and two for element j, enter the continuity condition at each row of (2.4.5). Therefore, the cost of solving this continuity condition is not great.

Chapter 3 Pseudospectral Polynomial Time-Marching Techniques

The pseudospectral methods provide a very useful tool for the solution of time-dependent differential equations. A standard scheme uses spectral methods to approximate the space derivatives and a finite difference approach to march the solution in time. This tactic results in an unbalanced scheme. Some alternative time-marching techniques, which approximate the time dependence of PDE solutions by using orthogonal polynomials or Faber polynomials involving powers of the spatial differential operator, have been introduced. These time-marching techniques can be used to evaluate the numerical solutions to spectral accuracy in both the space and time dimensions.

1 Evolution of the Time Dependence Using Chebyshev Polynomials

3.1.1 Algorithm

Consider the equation

$$\begin{aligned} &U_t - GU = 0, \;\; 0 \le x \le 2\pi \\ &U(x,0) = U_0(x) \end{aligned} \tag{3.1.1}$$

where G is a linear spatial differential operator (for example, $G = a\frac{\partial}{\partial x}$). Here, one can assume the periodic boundary conditions for the problem so that (3.1.1) can be discretized in space by using the pseudospectral Fourier methods outlined below:[39]

$$\begin{aligned} &\frac{\partial U_N}{\partial t} - P_N G P_N U_N = 0 \\ &U_N(x,0) = P_N U_0(x) \end{aligned} \tag{3.1.2}$$

where for any function *f(x)*, *P_Nf(x)* is its trigonometric interpolant at the collocation points

$$x_j = j\frac{\pi}{N},\; j = 0, 1, \cdots, 2N-1 \tag{3.1.3}$$

More precisely (the same as (2.2.4))

$$P_N f(x) = \sum_{l=-N}^{N} a_l e^{ilx} \tag{3.1.4}$$

where a_n is defined in (2.2.3). The solution of (3.1.2) is given by

$$U_N(x,t) = \exp(tP_N G P_N) U_0(x) \tag{3.1.5}$$

Except for a very simple operator G, it is impossible to construct the exponential matrix $\exp(tP_N G P_N)$ explicitly. Usually an approximation to the exponent is used.

According to the discussion in Chapter 2, the matrix $G_N = P_N G P_N$ is an anti-symmetric matrix and therefore has a complete set of 2N eigenvectors. Let $H_m(G_N t)$ be a polynomial approximation of $\exp(G_N t)$ of the form

$$H_m(G_N t) = \sum_{l=0}^{m} a_l (G_N t)^l \tag{3.1.6}$$

Tal-Ezer [39] has shown that

$$\begin{aligned} \left\| e^{G_N t} - H_m(G_N t) \right\|^2 &= \max_k \left| e^{\lambda_k t} - H_m(\lambda_k t) \right|^2 \\ &\leq \max_z \left| e^{zt} - H_m(zt) \right|^2 \\ &\text{where } \lambda_k,\ k = 1, 2, \cdots, 2N \text{ are the eigenvalues of } \mathrm{G}_N, \\ &z \in [-ia(N-1),\ ia(N-1)] \end{aligned} \tag{3.1.7}$$

We therefore seek a polynomial approximation with real coefficients to the function e^{zt} that will minimize the expression on the right hand of (3.1.7). Define

$$R = |a(N-1)| \tag{3.1.8}$$

Then one can find the Chebyshev polynomial approximation

$$H_m(zt) = \sum_{k=0}^{m} C_k J_k(Rt) Q_k\left(\frac{z}{R}\right) \tag{3.1.9}$$

where $C_0 = 1,\ C_k = 2$ for $k \geq 1,\ J_k(Rt)$ is the Bessel function of order k and the Q_ks are the modified Chebyshev polynomials in *z/R* defined as below:

$$Q_k(w) = (i)^k T_k(-iw), \qquad (3.1.10)$$
$$w = \frac{z}{R},\ w \in [-i,\ i]$$

Thus, using the recurrence relation satisfied by the Chebyshev polynomials

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x), \qquad (3.1.11)$$
$$T_0(x) = 1,\ T_1(x) = x,$$

it is easily verified that $Q_k(w)$ satisfies the following recurrence relation:

$$Q_{k+1}(w) = 2wQ_k(w) + Q_{k-1}(w), \qquad (3.1.12)$$
$$Q_0(w) = 1,\ Q_1(w) = w$$

3.1.2 Accuracy, Resolution And Convergence

The accuracy of a polynomial is defined by its asymptotic rate of convergence as m (the degree of the polynomial) tends to infinity. Let the interval of asymptotic behavior of the polynomial be denoted by $[m_0,\ \infty)$. Then the degree of the polynomial has to be greater or equal to m_0 in order to have proper time resolution. In [39], Tal-Ezer defined the condition of having meaningful resolution as one in which $m \geq m_0$ and the relative error norm is less than 10%. precisely, assume that for $m \in [m_0, \infty)$, the minimal m which achieves this accuracy is $\widetilde{m_0}$, Tal-Ezer defined the necessary and sufficient condition for resolution is $m \geq \widetilde{m_0}$. Applying this condition to my approximation problem means that one has to apply the spatial operator tG_N m_0 times in order to resolve N modes of the exact solution of (3.1.1) at time level *t*:

$$\begin{aligned} U_N(x,t) &= \exp(t\mathbf{G}_N)U^0(x) \\ &= \sum_{k=0}^{m_0} C_k J_k(tR) Q_k\left(\frac{G_N}{R}\right)U^0(x) \end{aligned} \tag{3.1.13}$$

It is known that the Bessel function $J_k(tR)$ converges to zero exponentially when *k* increases beyond *tR*. This implies that the interval of asymptotic behavior is $[m_0,\ \infty)$ where $m_0 = [tR]$. Hence, to resolve N modes for the problem of (3.1.2), one has to use the spatial operator at least $|at(N\text{-}1)|$ times. In contrast, in order to get a resolution of N modes by the leap-frog central-differencing time-marching scheme,

$$U^{n+1} = U^{n-1} + 2\Delta t G U^n \tag{3.1.14}$$

one has to operate with tG_N at least $\left(\frac{5}{3}\right)^{1/2}|taN|^{3/2}$ times [39]. A detailed analysis of the time resolution in standard finite difference time-marching scheme can be found in [26]

It is clear from the resolution discussion that resolution implies stability. In fact, since

$$\left|H_m(t\mathbf{G}_N) - e^{t\mathbf{G}_N}\right| \le const.$$

$H_m(tG_N)$ must be bounded independently of *m* and *N*.

Obviously, due to the stability limitation, the leap-frog scheme has to be iterated many times over a small time step Δt to get the resolution at time level *t*. In contrast, the Chebyshev polynomial approximation has no such kind of limitation. All the $|ta(N\text{-}1)|$ times operations can be evaluated in one time step $\Delta t = t$. According to the property of the Bessel function:

$$\begin{aligned} &J_m(m\phi) \le \alpha^m,\ (|\phi| < 1) \\ &\alpha \equiv \left|\frac{\phi \exp\left(\sqrt{1-\phi^2}\right)}{1+\sqrt{1-\phi^2}}\right| \end{aligned}$$

Tal-Ezer has proven in [39] that the error of Chebyshev approximation scheme in time converges to zero as

$$\alpha^m \underset{m\to\infty}{\longrightarrow} 0,\ 0 \le \alpha < 1 \tag{3.1.15}$$

Hence, this scheme has spectral accuracy both in time and space. In practice, if m is too large, the coefficients of Q_k in the polynomial may exceed machine precision, thus imposing a practical limitation.

In the scalar variable coefficient (still the one-dimensional) case, the operator G is

$$G = a(x)\frac{\partial}{\partial x} \tag{3.1.16}$$

Now the new discretized operator matrix

$$\mathbf{G}_N = \mathbf{A}_N \cdot \mathbf{D}_N \tag{3.1.17}$$

where $\mathbf{A}_N$ is a diagonal matrix

$$(\mathbf{A}_N)_{ij} = a(x_j)\delta_{ij}$$

and D_N approximates the derivative spectrally. It is clear that G_N is no longer a normal matrix. By a technique different from the constant coefficient case, it has been proven in [39] that the main results of the Chebyshev scheme are still valid as long as the wave velocity *a(x)* does not change its sign in the interval. Now the $\max|a(x)|$, $x \in [0,\ 2\pi]$ should replace the constant a in the previous case.

3.1.3 Second-order Case

Etgen [12] has extended Tal-Ezer's results to the second-order wave equation :

$$\frac{\partial^2 U}{\partial t^2} = -\mathcal{L}^2 U \tag{3.1.18}$$

where $-\mathcal{L}^2$ is a second-order spatial differential operator. The standard second-order finite difference approximation is expressed as the sum of two Taylor series in [12]:

$$\frac{\partial^2 U}{\partial t^2} \approx \frac{1}{\Delta t^2}[U(t+\Delta t) - 2U(t) + U(t-\Delta t)] \tag{3.1.19}$$

$$
\begin{aligned}
&= 2\left[\frac{1}{2}\frac{\partial^2 U}{\partial t^2} + \frac{\Delta t^2}{4!}\frac{\partial^4 U}{\partial t^4} + \frac{\Delta t^6}{6!}\frac{\partial^6 U}{\partial t^6} + \cdots\right] \\
&= 2[\cos(\Delta t\mathcal{L}) - 1]U
\end{aligned}
$$

By expressing (3.1.7) in terms of the real and imaginary parts, one has:

$$
\begin{aligned}
\left|e^{zt} - H_m(zt)\right|^2 &= \left|\cos(-izt) - H_m^R(-izt)\right|^2 \\
&\quad + \left|\sin(-izt) + H_m^I(-izt)\right|^2 \\
&\quad \text{where } H_m(zt) = H_m^R(-izt) + H_m^I(-izt)
\end{aligned} \tag{3.1.20}
$$

According to (3.8) in [39]:

$$
\begin{aligned}
H_m^R(t\mathcal{L}) &= J_0(Rt) + 2\sum_{k=1}^{\infty}(-1)^k J_{2k}(Rt)T_{2k}\left(\frac{i\mathcal{L}}{R}\right) \\
H_m^I(t\mathcal{L}) &= 2\sum_{k=1}^{\infty}(-1)^k J_{2k+1}(Rt)T_{2k+1}\left(\frac{i\mathcal{L}}{R}\right)
\end{aligned} \tag{3.1.21}
$$

The cosine of a differential operator acting on a function can be represented by an orthogonal polynomial series expansion as follows:

$$
\cos(t\mathcal{L}) = \sum_{k=0}^{\infty} C_{2k}J_{2k}(tR)Q_{2k}\left(\frac{\mathcal{L}}{R}\right) \tag{3.1.22}
$$

where C_{2k} , J_{2k} and Q_{2k} are defined the same as the previous definitions in (3.1.9). Particularly, based on (3.1.12), Q_{2k} satisfy the following recurrence relation:

$$
\begin{aligned}
&Q_0\left(\frac{\mathcal{L}}{R}\right) = I;\ Q_2\left(\frac{\mathcal{L}}{R}\right) = I - \frac{2\mathcal{L}^2}{R^2}; \\
&Q_{2k+2}\left(\frac{\mathcal{L}}{R}\right) = \left(-\frac{4\mathcal{L}^2}{R^2} + 2I\right)Q_{2k} - Q_{2k-2}
\end{aligned} \tag{3.1.23}
$$

where I is the identity operator. In [12], Etgen also gives the formal solution to the wave equation (3.1.18) for an initial value problem as:

$$U(t) = \cos(t\mathcal{L})U(0) + \frac{\sin(t\mathcal{L})}{\mathcal{L}}\frac{\partial U(0)}{\partial t} \tag{3.1.24}$$

This equation can be easily used to advance a solution to the wave equation in time. The time updating operator for any time level can be found by shifting the initial conditions to other values of t:

$$U(t+\Delta t) = -U(t-\Delta t) + 2\cos(\Delta t\mathcal{L})U(t) \tag{3.1.25}$$

Theoretically, in this scheme there's no limitation on the time step Δt. However, due to the resolution requirement, one has to use the spatial operator at least $R\Delta t$ times. In other words, the highest degree of the evaluated polynomial should be at least $(R\Delta t)$th order. Therefore, the largest Δt practically is limited by the highest order polynomial for given machine precision.

3.2 Approximating The Time Dependence By Complex Polynomials

The main limitation to the orthogonal polynomial time-marching scheme is that it requires a normal or skew-symmetric derivative matrix and its eigenvalues have to be purely real or imaginary. Therefore, a more general polynomial time-marching algorithm needs to be introduced.

3.2.1 Polynomial Interpolation Algorithm

Basically, the method described in the last section is the approximation of the time dependence of the solution, which can be represented as $f(A)$ where A is a finite operator. Approximating $f(A)$ can be reduced to a problem of approximating $f(z)$ where z belongs to a domain D in the complex plane. This domain includes all the eigenvalues of A [40, 41]. A possible approach is to expand the function as a sum of orthogonal polynomials. In the last section, it has been shown that this approach can be efficiently used for the function *f(z)=exp(tz)* where the domain is part of the imaginary (or the real) axis. For the more complicated domains, this method fails to demonstrate its effectiveness and efficiency. Thus, Tal-Ezer [40, 41] suggests an alternative way which is based on the complex polynomial interpolation for the topologically complex eigenvalue domains.

Consider a domain D in the complex plane $\mathbf{C}$. The D is a bounded continuum and its complement is simply connected in the extended plane and contains the point at ∞. As concluded in [40, 41, 10], a general approximation to a function *f(z)* in the arbitrary domain D is given by a Faber polynomial expansion. Let $\phi(z)$ be a conformal mapping from z to ω that maps the complement of D to the complement of a disk of radius ρ such that

$$\lim_{z\to\infty} \frac{\phi(z)}{z} = 1 \tag{3.2.1}$$

Here ρ is called the logarithmic capacity or the transfinite diameter of D. The conformal mapping $\phi(z)$ and its inverse mapping $\psi(\omega)$ are illustrated in Figure 3.1.

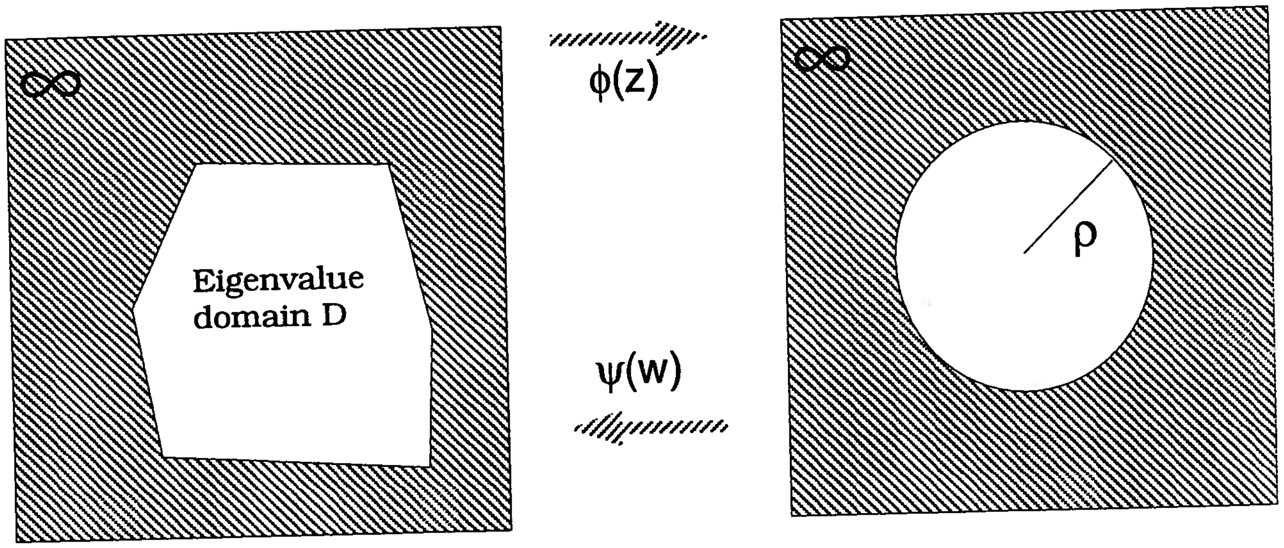


Figure 3.1 Conformal Mappings for the Newton-form Polynomial Interpolation

If the Laurent expansion at ∞ of $[\phi(z)]^m$ is

$$[\phi(z)]^m = z^m + C_{m-1}z^{m-1} + \cdots + C_1 z + C_{-1}z^{-1} + \cdots \tag{3.2.2}$$

then the Faber polynomial of degree m, $F_m(z)$, which corresponds to the domain D is the polynomial part of (3.2.2):

$$F_m(z) = z^m + C_{m-1}z^{m-1} + \cdots + C_0 \tag{3.2.3}$$

with

$$C_j = \frac{1}{2\pi j} \int\limits_{|z|=R} \frac{[\phi(z)]^m}{z^{j+1}} dz \tag{3.2.4}$$

where R is chosen sufficiently large so that D is contained in the interior of the region bounded by the circle $|z|=R$. If f is a function that is analytic at every point of D, then f can be approximated as:

$$f(z) = \sum_{k=0}^{\infty} a_k F_k(z) \tag{3.2.5}$$

The coefficients a_k are

$$a_k = \frac{1}{2\pi j} \int\limits_{|\omega|=r} \frac{f(\psi(\omega))}{\omega^{k+1}} d\omega \tag{3.2.6}$$

where $\psi(\omega)$ is the inverse of $\phi(z)$ and $r>\rho$ is sufficiently small so that f can be extended analytically to I_r (I_r is the closed Jordan region with boundary Γ_r, where Γ_r is the image under ψ of the circle $|\omega|=r$). Based on (3.2.5), the matrix function $f(A)$ can be approximated by the Faber polynomial expansion:

$$f(\mathbf{A}) = \sum_{k=0}^{\infty} a_k F_k(\mathbf{A}) \tag{3.2.7}$$

When D is an ellipse or the special case, a segment of the real or imaginary axis, F_k is the translated and scaled Chebyshev polynomial (Markushevich in [32]).

When D is a more complicated domain, the generalized Faber polynomial satisfies a k term recurrence relation instead of the 3–term recurrence for the Chebyshev polynomial. Thus it is more difficult to compute F_k's. In order to overcome this major drawback, Tal-Ezer has proposed the Newton-form interpolation in [40, 41] to simplify the problem as outlined below:

First, some proper interpolation points for a general domain D are chosen in two feasible ways:

a. The m zeros of $F_m(z)$;

b. $z_j = \psi(\omega_j),\ j = 1, \cdots, m+1$ where ω_j are the m+1 roots of the equation $\omega^{m+1} = \rho$. These points are called Fejér points.

Since, interpolating at $z_j = \psi(\omega_j),\ j = 0, \cdots, m$ does not involve computing Faber polynomials, it is a simpler and more efficient approach to the problem of approximation. The interpolating polynomial in Newton-form is

$$P_m(z) = \sum_{k=0}^{m} a_k R_k(z) \tag{3.2.8}$$

where a_k is the divided difference of order k: [1]

$$\begin{aligned} a_k &= f[z_0, \cdots, z_k] \\ &= \sum_{j=0}^{k} \frac{f_j}{\pi_k'(z_j)}\ \ k = 0, \cdots, m \\ \pi_k'(z_j) &= (z_j - z_0)\cdots(z_j - z_{j-1})(z_j - z_{j+1})\cdots(z_j - z_k) \\ f_j &= f(z_j) \end{aligned} \tag{3.2.9}$$

$$\begin{aligned} R_0(z) &= 1, \\ R_k(z) &= \prod_{i=0}^{k-1}(z - z_i),\ k = 1, \cdots, m \end{aligned} \tag{3.2.10}$$

In order to prevent overflow or underflow while computing the divided differences, the domain should be scaled such that its logarithmic capacity ρ will be equal to 1:

$$\hat{z} = \frac{z}{\rho} \tag{3.2.11}$$

Hence

$$\hat{f}(\hat{z}) = f(z) = f(\rho \cdot \hat{z}) \text{ and } P_m(\hat{z}) \approx \hat{f}(\hat{z}) \tag{3.2.12}$$

where

$$P_m(\hat{z}) = \sum_{k=0}^{m} b_k \hat{R}_k(\hat{z}) \tag{3.2.13}$$
$$b_k = \hat{f}[\hat{z}_0, \cdots, \hat{z}_k],\ k = 0, \cdots, m$$
$$\hat{R}_0(\hat{z}) = 1,$$
$$\text{and } \hat{R}_k(\hat{z}) = \prod_{i=0}^{k-1} (\hat{z} - \hat{z}_i),\ k = 1, \cdots, m$$

Then one can approximate the operator *f(A)* by $P_m\left(\hat{\mathbf{A}}\right)$, where

$$P_m\left(\hat{\mathbf{A}}\right) = b_0 I + b_1\left(\hat{\mathbf{A}} - \hat{z}_0 I\right) + b_2\left(\hat{\mathbf{A}} - \hat{z}_0 I\right)\left(\hat{\mathbf{A}} - \hat{z}_1 I\right) \tag{3.2.14}$$
$$+ \cdots + b_m\left(\hat{\mathbf{A}} - \hat{z}_0 I\right) \cdots \left(\hat{\mathbf{A}} - \hat{z}_{m-1} I\right)$$
$$\text{with } \hat{\mathbf{A}} = \left(\frac{1}{\rho}\right)\mathbf{A}$$

and ρ is the logarithmic capacity of the domain D.

3.2.2 Rate of Convergence

Before considering the convergence of the Newton-form interpolation approximation, one needs the following definitions: [40, 41]

"**Definition 1:** Let Γ_R be the image under ψ of the circle $|\omega|$=R (R>ρ), and I_R the closed Jordan region whose boundary is Γ_R. If *f(z)* is single-valued and analytic on I_R, then the sequence of polynomials $P_m(z)$ is said to convergence to *f(z)* on D maximally if

$$|f(z) - P_m(z)| \le C(\rho/R)^m,\ z \in D \tag{3.2.15}$$

where C depends on ρ and R but not on m or *z*."

"**Definition 2:** A set of points $z_i^{[m]}$ is said to be uniformly distributed on Γ_D (the boundary of D) if

$$\lim_{m\to\infty} |R_m(z)|^{1/m} = \rho \tag{3.2.16}$$

$$\text{where } R_m(z) = \prod_{i=1}^{m} (z - z_i)$$

It has been shown in [32] that either the m zeros of $F_m(z)$ or the m Fejér points are uniformly distributed on Γ_D."

According to the theorem in [58], if one chooses these uniformly distributed points $z_i^{[m]}$ as the interpolation points, the $P_m(z)$ converges maximally to $f(z)$ on D. Thus, one has

$$\frac{\rho}{R} = \text{the asymptotic rate of convergence} \tag{3.2.17}$$

Based on complex variable theory, Tal-Ezer has also shown that if $f(z)$ is an entire function, 3.2.15 is satisfied for arbitrary R.

Consequently, the degree m of the polynomial $P_m(z)$ can be determined in the following ways:

1. Getting the parameters R *(the disc radius)* and ρ (the logarithmic capacity) (analytically or numerically) and choosing m such that

$$\left(\frac{\rho}{R}\right)^m \simeq \epsilon, \tag{3.2.18}$$

where ϵ is the desired accuracy.

2. Computing the error numerically on a set of check points on the boundary for different *m's* and choosing m that will satisfy the desired accuracy.

This method can provide an accurate value of the m only when A is a normal matrix [40]. When A is "far" from normality, m should be increased by an amount that is not known before hand. Therefore, it is preferable, in this case, to use the infinite set of points generated by the second algorithm which will be described later.

3.2.3 Computational Considerations

In the Newton-form of polynomial interpolation, computing the divided differences is very vulnerable to round off errors and overflow. Besides scaling the logarithmic capacity to 1, the interpolation points should be arranged in a proper order. If the successive points are very close to each other, there exist severe round off errors or overflow for computing the divided differences.

Tal-Ezer gives two possible algorithms in [40, 41] to generate a set of interpolation points which don't cause severe round off errors and overflow.

Fejér point generating Algorithm 1: θ_1=0, j=l=1, $\delta\theta$=2π/m, find the largest k such that k is power of two and $k\delta\theta<\pi\leq 2k\delta\theta$. Then the algorithm proceeds as follows:

```
(1) For $i = 1$ until $l$ do
    $\eta = \theta_i + k\delta\theta$
    if $(\eta \geq \pi)$ goto (2)
    $j = j + 1$
    $\theta_i = \eta$
(2) end do
$l = j$
$k = k/2$
if $(k < 1)$ stop
goto (1)
```

This part of the algorithm produces arguments of points on the upper part of the unit circle. Thus

$$z_0^{(m)} = \psi(1),\ z_1^{(m)} = \psi(-1) \tag{3.2.19}$$
$$z_{2j}^{(m)} = \psi\left(e^{i\theta_{j+1}}\right),\ z_{2j+1}^{(m)} = \bar{z}_{2j}^{(m)}$$

$$1 \leq j \leq (m-1)/2$$

Here, $\psi()$ is the conformal mapping from the complement of the unit disc to the complement of the domain D. The set of points $z_j^{(m)}$ described above is equally distributed on the boundary of the domain D for any degree of m. It has the disadvantage of having to decide on m before starting the iterations. The next algorithm generates an infinite set of points free of this disadvantage.

Fejér point generating Algorithm 2:

```
$\delta\theta = \pi$
$k = 1$
$z_0 = \psi(1)$
$\theta_1 = 0$
(1) For $i = 1$ until $k$ do
    $\theta_{k+i} = \theta_i + \delta\theta$
    $z_{k+i-1} = \psi\left(e^{i\theta_{k+i}}\right)$
end do
$\delta\theta = \delta\theta/2$
$k = 2k$
goto (1)
```

Those points generated by Algorithm 2 are asymptotically equally distributed. The even distribution is achieved only when m is power of two. The uneven distribution of points in Algorithm 2 is the price one has to pay for the ability to add additional points. Therefore, in practice, one stops the iterations of Algorithm 2 at some proper degree m which satisfies the desired accuracy. Otherwise, one can just add some more points to the existing set of points by this algorithm without recomputing all the points until the accuracy is satisfied. In general, one can expect the interpolation based on Algorithm 1 to be more accurate.

When the operator matrix A is real, $f(z)$ accepts real values for z real, and also $f(\bar{z}) = \overline{f(z)}$, Tal-Ezer [40] has proven that the approximation algorithm can be modified to eliminate complex arithmetic.

Approximation algorithm with real arithmetic: $w_m \approx f(A)v$:

```
$u = [a_0 + a_1(A - z_0 I)]v$                                    (3.2.21)
$r = (A - z_0 I)(A - z_1 I)v$
For  $i = 1, \cdots,$ do
    $\bar{r} = \left(A - z_{2i}^R I\right) r$
    $u = u + a_{2i}^R r + a_{2i+1}^R \bar{r}$
    if $\frac{\|a_{2i}^R r + a_{2i+1}^R \bar{r}\|}{\|u\|} < \delta$ goto (1)
    $r = \left(A - z_{2i}^R I\right)\bar{r} + \left(z_{2i}^I\right)^2 r$
end do
(1)  $w_m = u$
```

Where z_k^R, z_k^I are respectively the real and imaginary part of the interpolation points z_k.

The only difficulty in finding the interpolation points is to get the conformal mapping $\psi(\omega)$ and its inverse $\phi(z)$. For few simple domains, it is possible to find this function analytically. For more complicated domains, one has to resort to a numerical approach. When the domain D is a polygon, the mapping function is a Schwarz-Christoffel transformation. The numerical routines described by Trefethen in [51] map the interior of the unit disk on the interior of the polygon. In order to map the exterior of the unit disk to the exterior of the polygon, when it is symmetric about the real axis, Tal-Ezer [40, 41] has modified the Trefethen's routines as outlined below.

One can treat the upper half of the exterior of the polygon as an interior of a polygon with vertex at infinity. Then, the desired mapping function is an interior Schwarz-Christoffel transformation composed with *f(u)* which maps the upper half of the exterior of the unit disk on the interior of the unit disk. The *f(u)* can be written as $f(u)=f_2(f_1(u))$, where

$$f_1 = \frac{1}{2}\left(u + \frac{1}{u}\right) \qquad (3.2.21)$$
$$f_2(u) = u_a \frac{u - \bar{u}_b}{u - u_b}$$

Here, u_a is a point on the unit circle and $\bar{u}_b$ is a point in the upper half plane. These points determine the orientation of the mapping.

Therefore, one construct the composed conformal mapping to map the complement of the unit disk to the complement of the domain *D*. In general, one can use this idea for any domain that can be divided into two symmetric parts, simply by rotating it such that the line of symmetry will match the real axis.

Since the Faber series expansion deals with more general complex domain approximation to the time dependence operator *f(A)* of the solution, one can expect that this method can solve the problems with a more complicated operator matrix *A*. For example, as discussed in the previous chapter, for the non-periodic case, Chebyshev collocation rather than Fourier pseudospectral has to be used. According to (2.3.8), the Chebyshev derivative operator may be neither normal nor skew-symmetric. Therefore, the approximation of the time-dependent symbolic solution has to be based on the Newton-form interpolation, rather than the Chebyshev expansion. The Newton-form interpolation is just one kind of computational implementation based on the Faber series expansion theory. By these new complex domain approximation methods, the time dependence of the solutions to the PDEs can be efficiently approximated as long as the eigenvalue distributions of spatial differential operators are available. Thus, the new time-marching technique based on the complex domain polynomial expansion can be applied to many time-evolution problems.

3.3 Numerical Results for Time-Marching in the Periodic Domain

Some simulations have been done to demonstrate the accuracy of the new polynomial time-marching techniques in the periodic domain. One-dimensional wave propagation in a slowly-varying medium with damping has been simulated by the Fourier pseudospectral time-marching and the results are compared with the corresponding analytical solution. The well-known d'Alembert solution is used here to check the one-dimensional homogeneous wave propagation simulation. A simple-shape two-dimensional scattering simulation is also performed via the Fourier pseudospectral time-marching algorithm.

3.3.1 Variable Wave Velocity and Damping in the Pseudospectral Time-Marching Scheme

In the equation (3.1.18), the operator $\mathcal{L}$ actually includes not only a spatial differential operator, but also the wave-velocity distribution in the medium. It has been proven (in [39]) that when the wave-velocity distribution does not change sign in the computational domain, the eigenvalues of $\mathcal{L}$ are also purely imaginary. Therefore, all the foregoing conclusions about the polynomial expansion remain valid.

For the cases in medium with constant damping, let us consider the following 1st order hyperbolic equation model:

$$\frac{\partial U}{\partial t} + \mathcal{L}U = -\alpha U \tag{3.3.1.1}$$

Here α is damping factor.

The symbolic solution to (3.3.1.1) is :

$$U(t) = e^{-(\alpha I + \mathcal{L})t} U(0) \tag{3.3.1.2}$$

where I is identity operator. Because α is constant, and I is commutative with $\mathcal{L}$, according to [33], one has:

$$
\begin{aligned}
e^I e^{\mathcal{L}} = e^{(I+\mathcal{L})} &= \sum_{n=0}^{\infty} \frac{(I+\mathcal{L})^n}{n!} \\
&= \sum_{n=0}^{\infty} \sum_{k=0}^{\infty} \frac{1}{k!(n-k)!} I^k \mathcal{L}^{n-k} \\
&= \sum_{j=0}^{\infty} \sum_{k=0}^{\infty} I^j \mathcal{L}^k \\
&= \sum_{j=0}^{\infty} \sum_{k=0}^{\infty} \mathcal{L}^k I^j \\
&= \sum_{n=0}^{\infty} \sum_{k=0}^{\infty} \frac{1}{k!(n-k)!} \mathcal{L}^k I^{n-k} \\
&= \sum_{n=0}^{\infty} \frac{(\mathcal{L}+I)^n}{n!} \\
&= e^{(\mathcal{L}+I)} = e^{\mathcal{L}} e^I
\end{aligned}
\qquad (3.3.1.3)
$$

Therefore, the (3.3.1.2) can be written as:

$$
U(t) = e^{-\alpha t} e^{-\mathcal{L}t} U(0) \qquad (3.3.1.4)
$$

So one can separate the damping factor term from the differential operator if α is a constant.

For non-constant damping and complicated wave-velocity distribution cases, since the eigenvalues of $\alpha + \mathcal{L}$ are no longer purely imaginary or real, the corresponding polynomial expansion has to be done in the complex plane by the Faber series.

3.3.2 Chebyshev polynomial expansion results

Some typical one-dimensional wave propagation and two-dimensional simple shape scattering problem simulations have also been implemented.

The first problem is a simple one-dimensional wave equation in a homogeneous medium (with periodic boundary condition):

$$\frac{\partial^2 U}{\partial t^2} - C^2 \frac{\partial^2 U}{\partial x^2} = 0, U(0,x) = e^{-\frac{(x-x_0)^2}{\sigma^2}}, U_t(0,x) = 0 \tag{3.3.1}$$

where C is a constant, and the initial distribution is a Gaussian pulse.

The corresponding analytical solution is :

$$U(t,x) = \frac{1}{2}\left[exp\left(-\frac{(x+ct-x_0)^2}{\sigma^2}\right) + exp\left(-\frac{(x-ct-x_0)^2}{\sigma^2}\right)\right] \tag{3.3.2}$$

The following computation is based on the parameters: wave velocity C= 1, time step Δt=0.1, number of grid points=256 or 128, and axis length 6.

Obviously, Figure 3.2 shows that the computation error will increase as time increases due to the recursive use of spectral time-marching. Theoretically, it is possible to compute the solution in one time step of any size. But since the highest Chebyshev polynomial to be evaluated is governed byΔtR, a time step which is too large requires very high order Chebyshev polynomials whose coefficients need more digits than the machine precision. In the above results, it seems that the errors are reduced by reducing the sampling grid number (from 256 to 128). Actually, it is the highest order of the truncated polynomial which directly affects the computational errors. When the grid number N is 256,ΔtR=14 and the computation uses up to the 20th order Chebyshev polynomial, i.e. 43% higher than ΔtR. As previously discussed, when k is beyond ΔtR, the Bessel function $J_k(\Delta tR)$ converges to zero exponentially. When the grid number N is 128 , ΔtR=7 and the computation uses up to the 14th order polynomial, i.e. 100% higher thanΔtR. The results clearly show a higher accuracy for the latter, even though it has a lower spatial sampling frequency (i.e. fewer frequency modes).

The second problem is a one-dimensional equation with a linear wave velocity distribution and constant damping:

$$\frac{\partial U}{\partial t} + C(x)\frac{\partial U}{\partial x} = -\alpha U \tag{3.3.3}$$

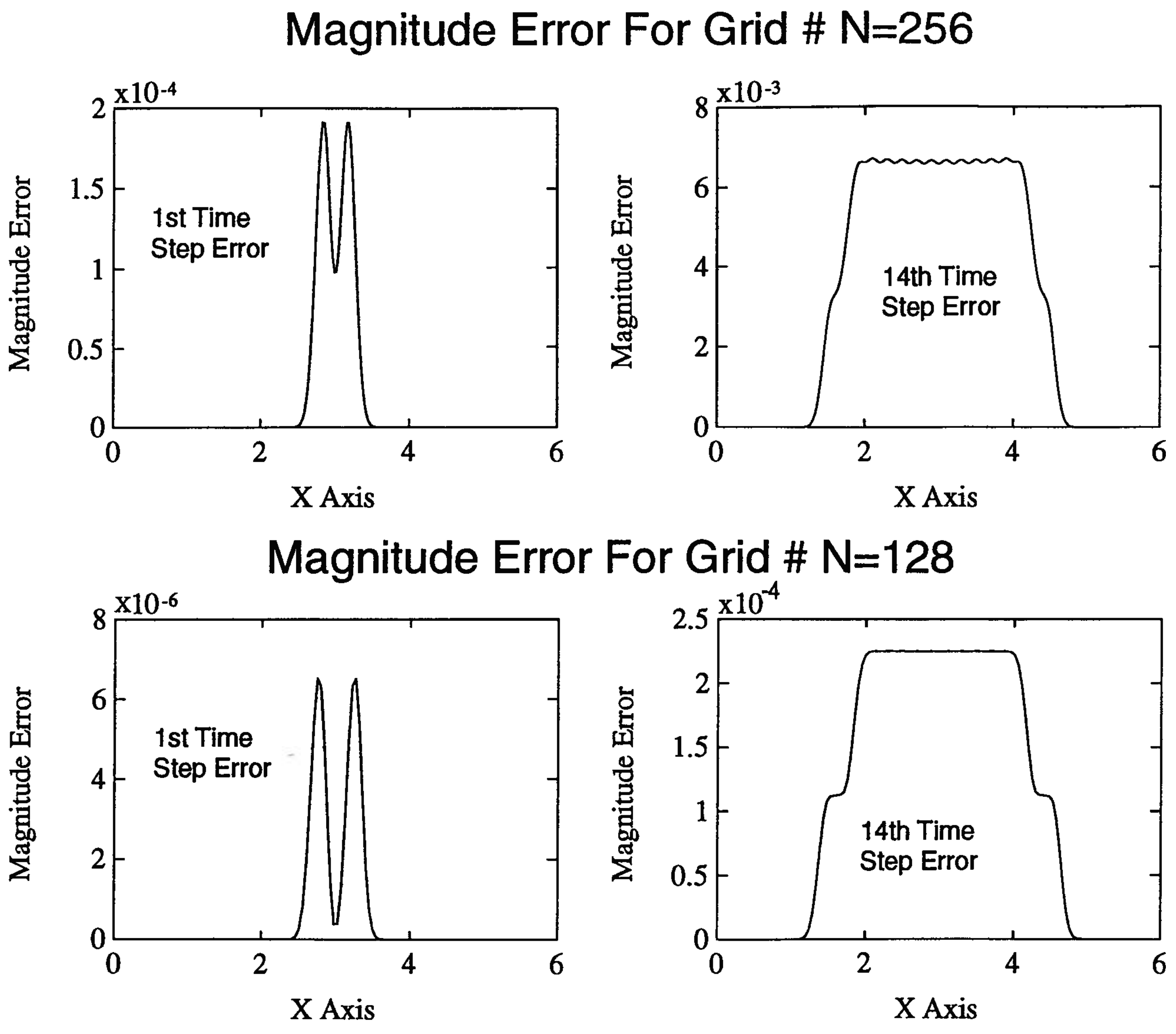


Figure 3.2 Computation Error for the One-dimensional Homogeneous Wave Equation

Let $C(x) = \frac{\partial x}{\partial t}$, then one has

$$\frac{\partial U(t,x(t))}{\partial t} = \frac{\partial U}{\partial t} + \frac{\partial U}{\partial x}\frac{\partial x}{\partial t} = -\alpha U \tag{3.3.4}$$

$$i.e.\frac{\partial U}{\partial t} = -\alpha U$$

that is:

$$\therefore U(t,x) = e^{-\alpha t}U(0,x) = e^{-\alpha t}f(\xi) \tag{3.3.5}$$

If *C(x)=ax+b* (linear wave velocity distribution),

$$C(x) = \frac{\partial x}{\partial t} = ax + b \tag{3.3.6}$$

the solution to (3.3.6) is: *(1/a)ln(ax+b)+constant=t*

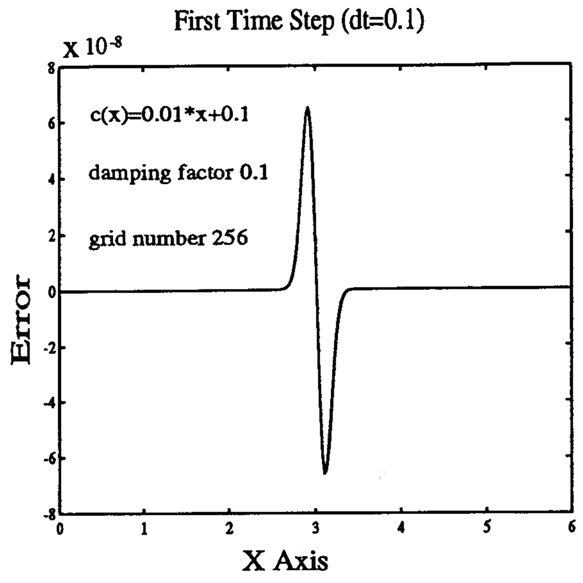
When t=0, *U(0,x)=f(ξ),i.e. x(0)=ξ*

$$\therefore t = \frac{1}{a}\ln\left(\frac{ax+b}{a\xi+b}\right) \text{ and} \tag{3.3.7}$$
$$\xi = \frac{(ax+b)e^{-at} - b}{a}$$

Substituting ξ in (3.3.5) by (3.3.7), one can get the final analytical solution to (3.3.3):

$$U(t,x) = e^{-\alpha t} f\left[\frac{(ax+b)e^{-at} - b}{a}\right] \tag{3.3.8}$$

Using (3.3.1.4) one can compute the numerical solution of (3.3.3). The following figures illustrate the comparisons and errors between the spectral numerical results by (3.3.1.4) and the analytical solution in (3.3.8). In Fig. 3.3 and the left part of Fig. 3.4, the polynomial series are truncated over 150% more than ΔtR, which governs the least order of the truncated series, but in the right part of Fig. 3.4 only 32% more than ΔtR polynomials are evaluated. It is similar to the homogeneous case. If one evaluates more polynomials, one can achieve higher computational accuracy. Comparing the left parts of the two figures, one can see the only difference between them is that the Nyquist rates of their wave velocity distributions are different (different linear rates). Thus, due to the aliasing problem, the same spatial sampling rate (grid number) results in different numerical accuracies. The third model simulated by the new time-marching algorithm is a simple two-dimensional scattering problem without damping. The following Fig. 3.5 and Fig. 3.6 are the two-dimensional initial value distribution and the wave velocity distribution. The medium wave velocity is variable in the domain with wave velocity lower inside a disc region simulating a cylinder. Another appended figure (in Appendix



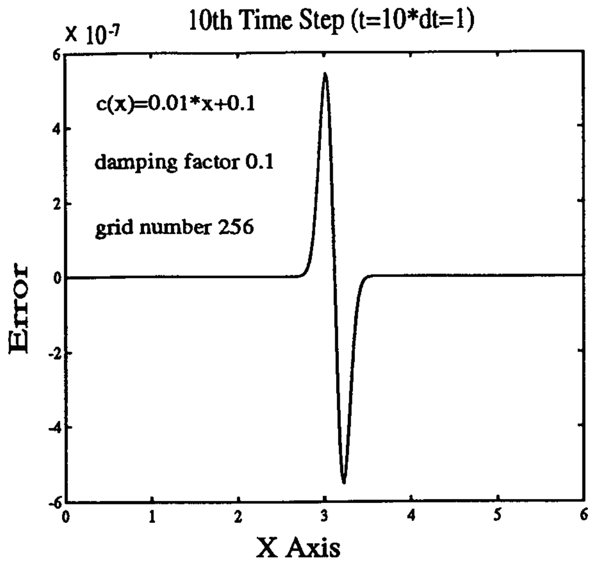
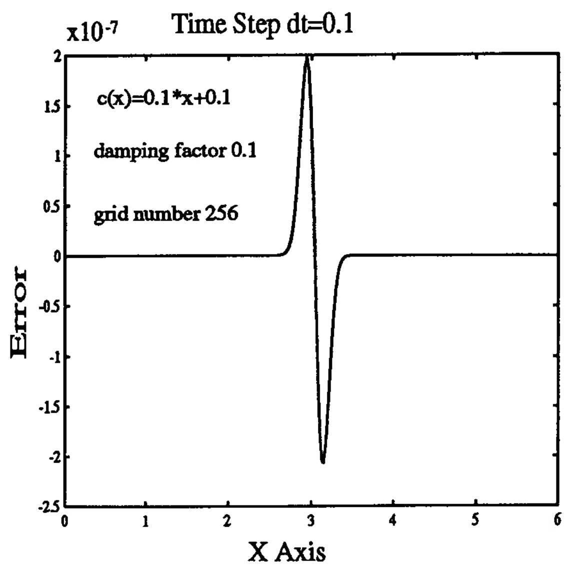


Figure 3.3 Magnitude of Errors At Different Time Steps



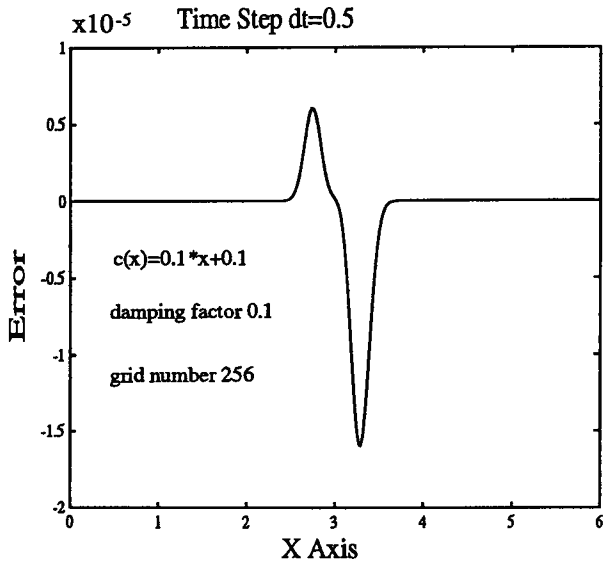


Figure 3.4 Magnitude of Errors For Different Time Step Sizes (at 1st time step)

A) illustrates one snapshot of the wave propagation and scattering procedure, in which the diffraction and reflection are obvious for this model. The refraction is relatively trivial because the wave velocity inside the low-velocity region (0.2 inside the disc, 1.0 outside).

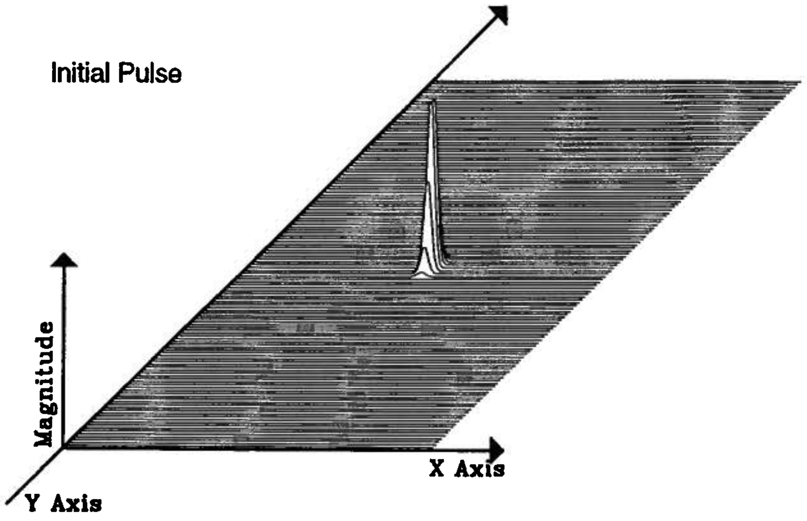


Figure 3.5 Initial Wave Distribution Simulating an Impulse

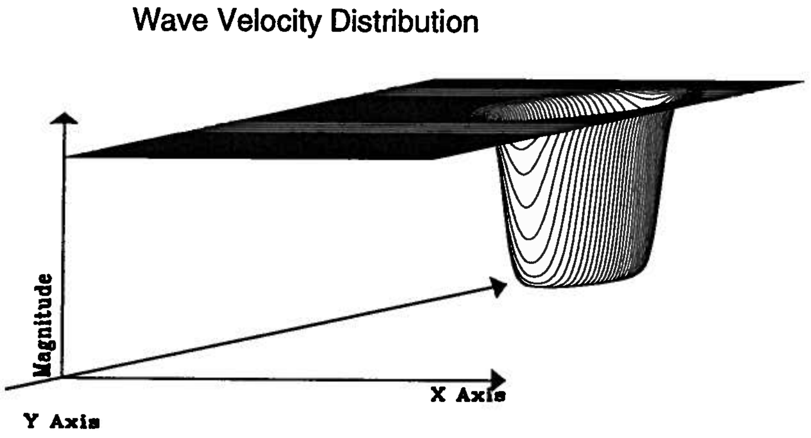


Figure 3.6 Wave-Velocity Distribution Simulating a Low Velocity Cylinder (The interface is created by a Rapid Velicity Transition)

Chapter 4 Pseudospectral Polynomial Time-marching for Non-periodic Boundary Value Problems

For periodic problems or the other "behavior boundary" value problems, due to the same boundary behavior of the corresponding spectral basis functions, the spectral operator automatically includes the boundary behavior. The Tal-Ezer time-marching technique, which is based on the approximation of the spectral operator function, can be directly applied. However, for non-periodic "numerical boundary" value problems, explicit boundary condition expressions are decidedly needed. In the conventional finite-difference time-marching scheme, a so-called "boundary-bordering" technique (Boyd, 1986) [5] can be used to impose explicitly the boundary conditions on the spectral operator matrix. In this thesis, a technique is developed to incorporate implicitly the homogeneous Dirichlet or Neumann boundary conditions into the Chebyshev collocated spatial operator. The eigenvalue distribution of the operator, which determines the time-marching polynomial, automatically includes the effect of the boundary conditions. Imposing homogeneous boundary conditions in this way makes it possible to apply directly Tal-Ezer's advanced time-marching technique to homogeneous Dirichlet or Neumann boundary value problems [31].

4.1 Homogeneous Dirichlet Boundary Conditions

First, let us consider the Chebyshev collocated second order derivative matrix with homogeneous Dirichlet boundary conditions. The second-order pseudospectral differentiation process can be described as follows:

(1) Interpolate the data at *Gauss-Lobatto* grid points

$$x_j = \cos\left(\frac{j\pi}{N}\right), \quad j = 1, \cdots, N-1 \tag{4.1.1}$$

by a Chebyshev polynomial *p(x)*, which should be constrained to satisfy homogeneous Dirichlet boundary conditions at the end points $x_0 = +1$ and $x_N = -1$

$$p(x_0) = p(x_N) = 0 \tag{4.1.2}$$

(2) Differentiate the interpolant twice to obtain estimates $p''(x_j)$ of the second derivative of the data at each grid point.

Since the differentiation process is linear it can be described by an (N-1)×(N-1) matrix D^2_{sp} (Weideman *et al* 1988) [59]. This matrix is typically neither sparse nor symmetric, in contrast to the situation with finite differences. Once the boundary conditions (4.1.2) and collocation points x_j are fixed, the differentiation matrix D^2_{sp} is implicitly determined. According to Gottlieb *et al* [20], the explicit formulas for the Chebyshev collocated second-order differentiation matrix entries (without boundary conditions) can be calculated as follows:

$$\frac{\partial^p (P_N f(x_k))}{\partial x^p} = \sum_{j=0}^{N} f(x_j)(D_p)_{kj} \tag{4.1.3}$$

$$\text{where } (D_1)_{kj} = \frac{c_k}{c_j}\frac{(-1)^{j+k}}{x_k - x_j} \quad (k \neq j)$$

$$c_0 = c_N = 2, \quad c_j = 1, \ (1 \leq j \leq N-1)$$

$$(D_1)_{jj} = -\frac{x_j}{2\left(1 - x_j^2\right)}, \ (D_1)_{00} = \frac{2N^2+1}{6} = -(D_1)_{NN}$$

$$\text{and } D_p = (D_1)^p$$

Under homogeneous Dirichlet boundary constraints, one has

$$\begin{bmatrix} 0 \\ u''_1 \\ \vdots \\ u''_{N-1} \\ 0 \end{bmatrix}_{(N+1)\times 1} = [D_2]_{(N+1)^2} \begin{bmatrix} u_0 = 0 \\ u_1 \\ \vdots \\ u_{N-1} \\ u_N = 0 \end{bmatrix}_{(N+1)\times 1} \tag{4.1.4}$$

Since u_0 and u_N are always fixed as zero, one has no need to evaluate their derivatives. Due to Trefethen *et al* in [54], similar to the description of the first-order differentiation matrix entries, one can easily obtain

$$\begin{bmatrix} u_1'' \\ \vdots \\ u_{N-1}'' \end{bmatrix}_{(N-1)\times 1} = \left[D^2_{sp}\right]_{(N-1)^2} \begin{bmatrix} u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}_{(N-1)\times 1} \tag{4.1.5}$$

where ${D^2}_{sp}$ is the same matrix as D_2, but with the first, last rows and columns deleted (masked by the zero boundary conditions). Thus, it is two ranks lower than the original D_2, in contrast to the situation of two ranks higher as in the "boundary bordering" scheme.

Even though it has been proven by Gottlieb *et al* [21] that all eigenvalues of ${D^2}_{sp}$ for collocation at *Gauss-Lobatto* points are real, distinct, and negative, Weideman *et al* [59] discovered that only a proportion, $2/\pi$, of the numerically computed eigenvalues are accurate approximations to those of the continuous problem. However, it is fortunate that all one needs in the new time-marching scheme are just the eigenvalue bounds in the complex plane, rather than the accurately computed eigenvalues. As described before, the domain D which determines the time-marching polynomials should include all the eigenvalues of the spatial differentiation matrix. In many applications, the time dependence of the solution (or the symbolic solution for time-marching) is usually an entire function, such as an exponential, *sin* or *cos*. Thus, as long as all eigenvalues are located within D, the time-marching can be done properly no matter what the actual shape of the domain D. In practice, in order to efficiently compute the conformal mapping $\phi(z)$ and $\psi(\omega)$, sometimes the original eigenvalue domain needs to be re-scaled so that it is easier to construct the conformal mapping. However, all eigenvalues are still included inside.

4.2 Homogeneous Neumann Boundary Conditions

For homogeneous Neumann boundary conditions, since there are no fixed function values, but the fixed first order derivatives at the end points, imposing the boundary conditions into the second-order differentiation matrix is not as straightforward as for Dirichlet boundary conditions.

According to equation (4.1.3), one has

$$\begin{bmatrix} u''_0 \\ \vdots \\ u''_N \end{bmatrix} = D_1 D_1 \begin{bmatrix} u_0 \\ \vdots \\ u_N \end{bmatrix} \tag{4.2.1}$$

for the no-boundary constraint case. The homogeneous Neumann boundary conditions at Gauss-Lobatto grid points are

$$u'(x_0) = u'(x_N) = 0 \tag{4.2.2}$$

Then equation (4.2.1) can be re-written as

$$\begin{bmatrix} u''_0 \\ \vdots \\ u''_N \end{bmatrix} = D_1 \begin{bmatrix} 0 \\ u'_1 \\ \vdots \\ u'_{N-1} \\ 0 \end{bmatrix} = \begin{bmatrix} d_{00} & \cdots & d_{1N} \\ \vdots & \ddots & \vdots \\ d_{N0} & \cdots & d_{NN} \end{bmatrix} \begin{bmatrix} 0 \\ u'_1 \\ \vdots \\ u'_{N-1} \\ 0 \end{bmatrix}, \tag{4.2.3a}$$

$$\text{and} \begin{bmatrix} 0 \\ u'_1 \\ \vdots \\ u'_{N-1} \\ 0 \end{bmatrix} = D_1 \begin{bmatrix} u_0 \\ \vdots \\ u_N \end{bmatrix} = \begin{bmatrix} d_{00} & \cdots & d_{1N} \\ \vdots & \ddots & \vdots \\ d_{N0} & \cdots & d_{NN} \end{bmatrix} \begin{bmatrix} u_0 \\ \vdots \\ u_N \end{bmatrix} \tag{4.2.3b}$$

$$\text{where } d_{ij} = (D_1)_{ij} \tag{4.2.3c}$$

Obviously, when one calculates the first-order derivative, one only needs to evaluate $[u'_1, \cdots, u'_{N-1}]$. The first and the last rows of the differentiation matrix D_1 in (4.2.3a) can be deleted. Thus, the first-order derivative evaluation becomes

$$\begin{bmatrix} u'_1 \\ \vdots \\ u'_{N-1} \end{bmatrix}_{(N-1)\times 1} = \begin{bmatrix} d_{1,0} & \cdots & d_{1,N} \\ \vdots & \ddots & \vdots \\ d_{N-1,0} & \cdots & d_{N-1,N} \end{bmatrix}_{(N-1)\times(N+1)} \begin{bmatrix} u_0 \\ \vdots \\ u_N \end{bmatrix}_{(N+1)\times 1} \tag{4.2.4}$$

Furthermore, the fixed zero first-order derivatives at end points x_0=+1 and x_N=−1 will mask out the first and the last columns of the D_1 matrix in (4.2.3a). Therefore the second-order differentiation process becomes

$$\begin{bmatrix} u''_0 \\ \vdots \\ u''_N \end{bmatrix}_{(N+1)\times 1} = \begin{bmatrix} d_{0,1} & \cdots & d_{0,N-1} \\ \vdots & \ddots & \vdots \\ d_{N,1} & \cdots & d_{N,N-1} \end{bmatrix}_{(N+1)\times(N-1)} \begin{bmatrix} u'_1 \\ \vdots \\ u'_{N-1} \end{bmatrix}_{(N-1)\times 1} \tag{4.2.5}$$

From (4.2.1) — (4.2.5) one can note that the Chebyshev collocated second-order derivative matrix with homogeneous Neumann boundary conditions at the end grid points can be expressed as:

$$[D_{sp}^2]_{(N+1)^2} = \begin{bmatrix} d_{0,1} & \cdots & d_{0,N-1} \\ \vdots & \ddots & \vdots \\ d_{N,1} & \cdots & d_{N,N-1} \end{bmatrix}_{(N+1)\times(N-1)} \cdot \begin{bmatrix} d_{1,0} & \cdots & d_{1,N} \\ \vdots & \ddots & \vdots \\ d_{N-1,0} & \cdots & d_{N-1,N} \end{bmatrix}_{(N-1)\times(N+1)} \qquad (4.2.6)$$

This (N+1)×(N+1) matrix D_{sp}^2 implicitly reflects both the Chebyshev collocated space differentiation at interior Gauss-Lobatto points and the homogeneous Neumann boundary conditions at end points.

By the foregoing manipulations, the homogeneous Dirichlet or Neumann boundary conditions are incorporated into the spectral operator. Therefore, the polynomial time-marching based on the eigenvalue characteristics of the spectral operator will automatically generate the boundary-constrained solution. For the Dirichlet boundary case, since the time-marching is evaluated only at interior points, at every time step the end point values should be updated separately by the boundary conditions. But for the Neumann boundary conditions, the function value time-marching is evaluated at all grid points (D_{sp}^2 is an (N+1)×(N+1) matrix) but with the incorporated first derivative constraints. Therefore, all grid point values will be updated.

4.3 Numerical Examples

The first numerical example used to demonstrate the accuracy and efficiency of the polynomial time-marching for boundary value problems is a homogeneous wave equation with Dirichlet boundary conditions on the interval [-1,1].

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} = 0, & -1 \le x \le 1, \ \ t > 0 \\ u(x,0) = u_0(x), \ \ u_t(x,0) = 0 & \\ u(-1,t) = u(+1,t) = 0, & t \ge 0 \end{cases} \qquad (4.3.1)$$

In order to get a symbolic solution as an entire function (so that the eigenvalue domain can be regularized), one can make some variable substitutions. Set $\frac{\partial u}{\partial t} = v$ and the above equation becomes

$$\begin{bmatrix} u_t \\ v_t \end{bmatrix} = \begin{bmatrix} 0 & I \\ \frac{\partial^2}{\partial x^2} & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \tag{4.3.2}$$

$$\text{with } \begin{bmatrix} u \\ v \end{bmatrix}_{t=0} = \begin{bmatrix} u_0 \\ 0 \end{bmatrix}, \quad u(\pm 1, t) = 0$$

Chebyshev collocation at Gauss-Lobatto points (4.1.1) is represented as

$$\begin{bmatrix} u_j \\ v_j \end{bmatrix}_t = G_N \begin{bmatrix} u_j \\ v_j \end{bmatrix}, \quad j = 1, \cdots, N-1 \tag{4.3.3}$$

$$u_0 = u_N = 0$$

$$\text{where } G_N = \begin{bmatrix} 0 & I_{N-1} \\ D_{sp}^2 & 0 \end{bmatrix}$$

where I_{N-1} is an identity matrix of rank (N-1), D_{sp}^2 is defined in Section 4.1, a Chebyshev collocation differentiation operator with homogeneous Dirichlet boundary conditions. The symbolic solution of equation (4.3.3) is

$$\begin{bmatrix} \bar{u} \\ \bar{v} \end{bmatrix} = \exp(G_N t) \begin{bmatrix} \bar{u}_0 \\ 0 \end{bmatrix} \tag{4.3.4}$$

where $\bar{u} = [u_1, \cdots, u_{N-1}]$ and $\bar{v} = [v_1, \cdots, v_{N-1}]$ are (N-1)×1 vectors for the interior grid points.

At this stage, the solution of equation (4.3.1) can be computed by the Newton-form interpolation time-marching technique. Even though the matrix G_N is twice as large as the D_{sp}^2, the actual computation will still be concentrated on evaluating D_{sp}^2 due to the sparse characteristics of the matrix G_N. This operator matrix is far from normal. Therefore, it is preferable to use the time-marching scheme based on an infinite set of interpolation points in the domain D. Because of the special structure of the operator

G_N, the eigenvalues of G_N can be easily obtained from those of D^2_{sp}. Suppose one of the eigenvalues of G_N is g, then one can get the following:

$$G_N \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} 0 & I_N \\ D^2_{sp} & 0 \end{bmatrix}_{(2N)^2} \begin{bmatrix} X \\ Y \end{bmatrix} = g \begin{bmatrix} X \\ Y \end{bmatrix}, \tag{4.3.5}$$

$$\therefore \begin{cases} Y = gX \\ D^2_{sp} X = gY \end{cases}$$

$$\text{Finally, } D^2_{sp} X = g^2 X$$

Thus, it can be concluded that the eigenvalues of the operator G_N are the square roots of those of D^2_{sp}. Therefore, the eigenvalue distribution bounds of G_N can also be directly obtained from the eigenvalues distribution of D^2_{sp}. An eigenvalue distribution domain D can then be accordingly configured based on these bounds.

According to the definition of D^2_{sp} in the previous section, the elements of the matrix operator G_N are purely real, and the exponential function satisfies the condition of $f(\overline{z}) = \overline{f(z)}$. Due to the proof in [40](Tal-Ezer 1989), one can use the algorithm designed to carry out only real arithmetic operations. Thus, the time-marching algorithm in (3.2.21) should be preferred.

In Fig. 4.1 and 4.2, which show the numerical results of the time-marching of the problem (4.3.1), the initial pulse is a smooth polynomial pulse. The time-marching error is less than 10^{-5}. The corresponding analytical solution is based on the d'Alembert solution combined with the method of images. The numerical results in the following figures show that under this time-marching scheme and error control factor, the space discretization error due to spectral sampling dominates in the solution. This can be seen by comparing the maximum errors at each time step for the case N=64 in Figure 4.1 and N=31 in Figure 4.2. In both figures, the wave distribution at t=1 should be a flat line of zero, therefore the wave amplitude at t=1 is identical to the numerical computational errors at t=1.

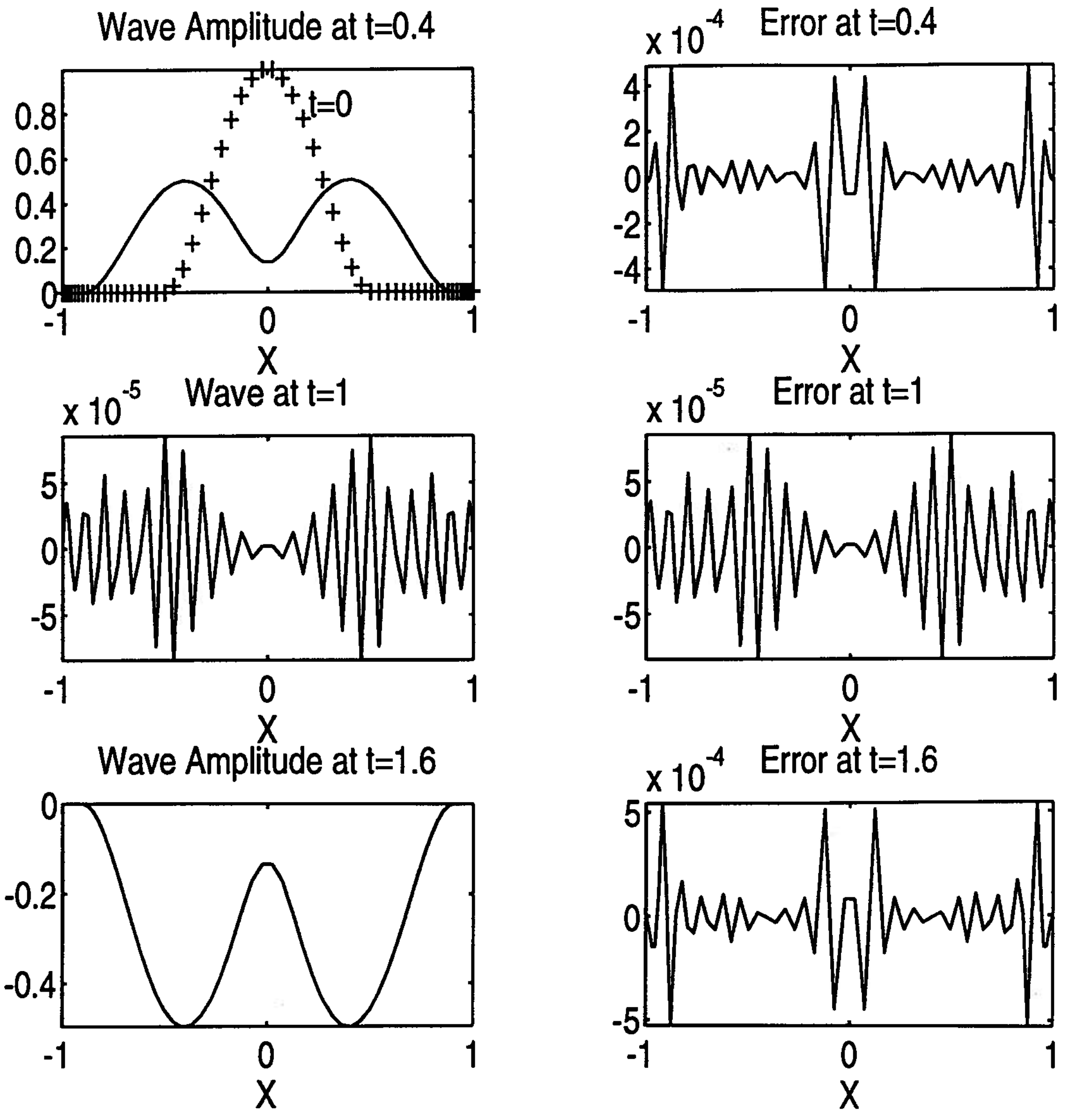


Figure 4.1 Homogeneous Dirichlet Initial-Boundary Value Problem N=64 (time step 0.005)

The numerical evidence listed in Table 4.1 shows that the stability limitation for the time step in the polynomial time-marching scheme is approximately *$O(1/N^2)$*. The experimental data in Table 4.1 are based on the polynomial or Gaussian pulse initial

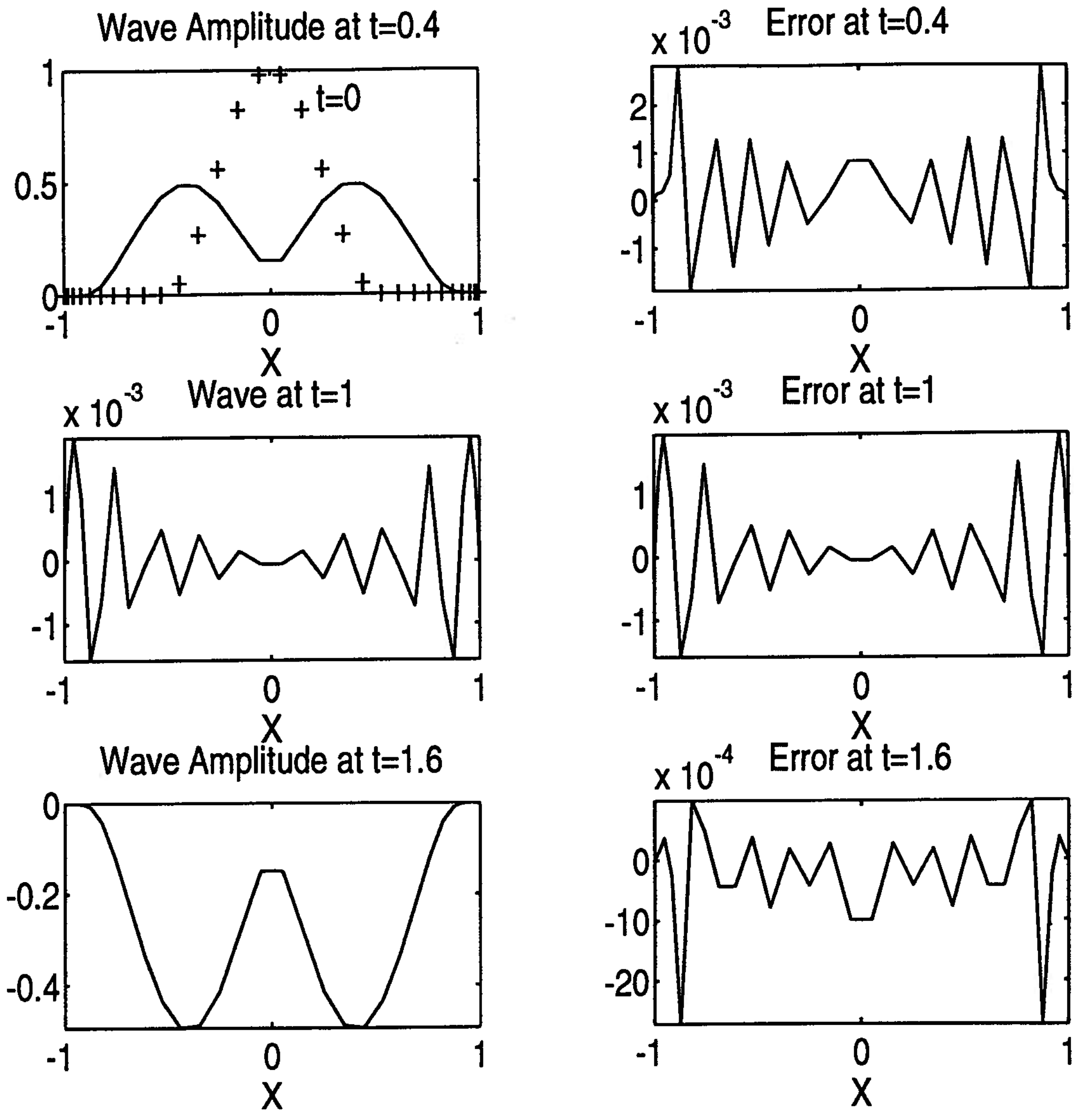


Figure 4.2 Homogeneous Dirichlet Initial-Boundary Value Problem N=31 (time step 0.005)

distribution and 20 time step calculations. The time-marching error control factor δ is less than 10^{-3}.

Table 4.1 Numerical Stability Limitation for the Polynomial Time-marching Step Size

	N=32	N=64	N=128
Dirichlet boundary case	Δt_{max}=0.25	Δt_{max}=0.0625	Δt_{max}=0.0156
Neuman boundary case	Δt_{max}=0.25	Δt_{max}=0.0625	Δt_{max}=0.0156

Since there is no theoretical stability analysis available for the new polynomial time-marching algorithm, a rough explanation and qualitative analysis can be presented here. Numerically, the major cause of the instability is from the round-off errors of the Newton-form interpolation points z_i. Since the maximum eigenvalue of the operator $G_N\Delta t$ is affected by the time step Δt, the interpolation points z_i, which are distributed along the eigenvalue domain boundary, are also affected by the time step size. If the time step is too large, the interpolation points, and thereby the divided difference coefficients a_k, could be large enough to cause significant round-off errors. A similar situation occurs in Chebyshev polynomial time-marching (Luo and Yedlin 1993)[30].

The second numerical example is a homogeneous initial boundary value problem with homogeneous Neumann boundary conditions

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} = 0, & -1 \leq x \leq 1, \;\; t > 0 \\ u(x,0) = \cos(\pi x), \;\; u_t(x,0) = 0 & \\ u_x(-1,t) = u_x(+1,t) = 0, & t \geq 0 \end{cases} \tag{4.3.6}$$

The analytical solution to the above equation is

$$u(x,t) = \cos(\pi t) \cdot \cos(\pi x) \tag{4.3.7}$$

The solution procedure is the same as for Dirichlet boundary problem described before, except that the operator G_N now is defined as

$$G_N = \begin{bmatrix} 0 & I_{N+1} \\ D_{sp}^2 & 0 \end{bmatrix}_{(2N+2)^2} \tag{4.3.8}$$

where D^2_{sp} is defined in (4.2.6). Under the same parameter conditions (N=31, Δt=0.01 and time-marching error control factor < 10^{-5}), the numerical results of this problem show that an *O(10*$^{-6}$*)* magnitude accuracy can be achieved. In this numerical example, since the initial distribution is *cos* which can be precisely represented by Chebyshev collocation, the major error only comes from the time-marching approximation.

4.4 General Homogeneous Boundary Conditions

The boundary condition manipulation technique can be easily extended to the representation of more general cases. Suppose the wave equation is given with the general homogeneous boundary conditions as shown below:

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} = 0, & -1 \le x \le 1, \ t > 0 \\ u(x,0) = U_0(x), \ u_t(x,0) = 0, & -1 \le x \le 1 \\ (\frac{\partial u}{\partial x} - \alpha u)|_{x=-1} = 0, & t \ge 0 \\ (\frac{\partial u}{\partial x} + \beta u)|_{x=+1} = 0, & t \ge 0\,. \end{cases} \tag{4.4.1}$$

Where α and β are constants. Obviously, when $\alpha \gg 1$, $\beta \gg 1$, the boundary conditions approximate the homogeneous Dirichlet case; when $\alpha \ll 1$, $\beta \ll 1$, the boundary conditions approximate the homogeneous Neumann case.

Collocated at the *Gauss-Lobatto* points (4.1.1), the constraints at the end points are

$$\begin{cases} \frac{\partial u_N}{\partial x} = \alpha u_N, & \text{at } x_N = -1 \\ \frac{\partial u_0}{\partial x} = -\beta u_0, & \text{at } x_0 = 1 \end{cases} \tag{4.4.2}$$

Consequently, one has the 1st order Chebyshev collocated derivative as shown below:

$$\begin{bmatrix} u'_0 \\ u'_1 \\ \vdots \\ u'_{N-1} \\ u'_N \end{bmatrix} = \begin{bmatrix} -\beta & 0 & \cdots & 0 & 0 \\ d_{1,0} & d_{1,1} & \cdots & d_{1,N-1} & d_{1,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d_{N-1,0} & d_{N-1,1} & \cdots & d_{N-1,N-1} & d_{N-1,N} \\ 0 & 0 & \cdots & 0 & \alpha \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \\ u_N \end{bmatrix} \tag{4.4.3}$$

Where d_{ij}s are defined in (4.1.3). Then the 2nd order derivative can be represented as

$$\begin{bmatrix} u''_0 \\ u''_1 \\ \vdots \\ u''_{N-1} \\ u''_N \end{bmatrix} = [D_1]_{(N+1)^2} \begin{bmatrix} -\beta & 0 & \cdots & 0 & 0 \\ d_{1,0} & d_{1,1} & \cdots & d_{1,N-1} & d_{1,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d_{N-1,0} & d_{N-1,1} & \cdots & d_{N-1,N-1} & d_{N-1,N} \\ 0 & 0 & \cdots & 0 & \alpha \end{bmatrix}_{(N+1)^2} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \\ u_N \end{bmatrix} \tag{4.4.4}$$

Here the matrix D_1 is also defined in (4.1.3).

Thus, it can be concluded that the 2nd order Chebyshev collocation differential operator, including the general homogeneous boundary conditions in (4.4.1), is given by:

$$[D_{sp}^2]_{(N+1)^2} = [D_1]_{(N+1)^2} \begin{bmatrix} -\beta & 0 & \cdots & 0 & 0 \\ d_{1,0} & d_{1,1} & \cdots & d_{1,N-1} & d_{1,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d_{N-1,0} & d_{N-1,1} & \cdots & d_{N-1,N-1} & d_{N-1,N} \\ 0 & 0 & \cdots & 0 & \alpha \end{bmatrix}_{(N+1)^2} \tag{4.4.5}$$

This general mixed boundary condition is extremely useful in simulating some wave scattering problems in a domain with leaking walls, e.g. the wave propagation in a wave guide.

4.5 The Two-Dimensional Case

In the two-dimensional case, the Chebyshev collocation is done along *x, y* directions. The eigenvalue analysis for the collocated derivative operator is more complicated. Suppose one can represent the two-dimensional *Laplace* operator as below:

$$\begin{aligned} \mathcal{L}^2 U &= \frac{\partial^2 U}{\partial x^2} + \frac{\partial U}{\partial y^2} \\ &= D_x^{(2)} U + U D_y^{(2)} \end{aligned} \tag{4.5.1}$$

Here, $D_x^{(2)}$, $D_y^{(2)}$ include homogeneous boundary conditions and can be obtained through the procedure described in the previous sections. Instead of a one-dimensional vector, here U is a two-dimensional matrix with x-row-y-column orientation:

$$U_{(N+1)\times(M+1)} = \begin{bmatrix} u_{00} & \cdots & u_{0M} \\ \vdots & \cdots & \vdots \\ u_{N0} & \cdots & u_{NM} \end{bmatrix}_{(N+1)\times(M+1)} \tag{4.5.2}$$

For this orientation, one can have the following relationships for the operator matrices :

$$D_x^{(2)} = D_{2sp}^N \tag{4.5.3}$$

$$D_y^{(2)} = \left(D_{2sp}^M\right)^T$$

where D_{2sp}^N, D_{2sp}^M are the corresponding one-dimensional Chebyshev collocated 2nd order derivative operators (including homogeneous boundary conditions) for *N+1* and *M+1* grid points, respectively.

To get the eigenvalue bounds of the operator $\mathcal{L}^2$, one needs to use *Kronecker product* and to stack the matrix $U_{(N+1)\times(M+1)}$ to a vector V

$$V = \begin{bmatrix} u_{00} \\ \vdots \\ u_{0M} \\ u_{10} \\ \vdots \\ u_{1M} \\ \vdots \\ \vdots \\ u_{NM} \end{bmatrix}_{(N+1)(M+1)\times 1} \tag{4.5.4}$$

The *Kronecker product* (sometimes called the *direct product*) is defined as: [4]

$$A_{m\times n} \otimes B_{p\times q} = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix} \tag{4.5.5}$$

Each submatrix in the above *(mp)*×*(nq)* matrix has dimensions $p\times q$. The *Kronecker product* is distributive and associative [4]. Based on the description of Barnett [4], the matrix equation

$$A_{n\times n}X_{n\times m} + X_{n\times m}B_{m\times m} = C_{n\times m} \tag{4.5.6}$$

is equivalent to

$$\left([A_{n\times n} \otimes I_m]_{nm\times nm} + \left[I_n \otimes B^T_{m\times m}\right]_{nm\times nm}\right)[\text{vec}(X)]_{nm\times 1} = [\text{vec}(C)]_{nm\times 1} \tag{4.5.7}$$

Here, vec() is used to represent the re-stacking procedure along the column, as described in (4.5.4). Hence, one can re-write (4.5.1) as:

$$\left(D_x^{(2)} \otimes I_{M+1} + I_{N+1} \otimes \left(D_y^{(2)}\right)^T\right)[V]_{(N+1)(M+1)\times 1} \tag{4.5.8}$$

According to the property of *Kronecker product* [4], if $D_x^{(2)}$ has eigenvalues $\lambda_i, \quad i = 0, 1, \cdots, N$, and $D_y^{(2)}$ has eigenvalues $\mu_j, \quad j = 0, 1, \cdots, M$, then the *Kronecker product* operator in (4.5.8) has *(M+1)(N+1)* eigenvalues $l_k = \lambda_i + \mu_j, \quad k = 0, 1, \cdots, (M+1)(N+1)$. Thus, the bounds of the two-dimensional Chebyshev collocation operator can be directly obtained from the maximum bound of the corresponding one-dimensional operator bounds along X and Y. The eigenvalue bounds of the operator determine the size and the shape of the domain D, thereby determining the logarithmic capacity and the distribution of the Fejèr points used in (3.2.14).

It has been shown that the Chebyshev collocation approximation for spatial derivatives can be combined with the polynomial time-marching technique, provided that the corresponding boundary conditions are properly incorporated into the spectral operator. This technique extends the applicability of the new spectrally accurate polynomial time-marching technique to many non-periodic problems. Homogeneous boundary conditions have been successfully included within the corresponding Chebyshev collocation operator. Under this time-marching scheme, besides the significant improvement of the accuracy in one dimension, the largest time step size required by stability is *O(1/N^2)*. In the two-dimensional case, the eigenvalue bounds of the derivative operator incorporated with boundary conditions can be obtained through the *Kronecker product* properties. Compared to some implicit finite-difference time-marching schemes, which may allow larger time steps, this algorithm also avoids matrix inversion required by some implicit time-marching schemes.

Chapter 5 Pseudospectral Polynomial Time-marching for Non-reflecting Boundary Problems

When simulating wave propagation, usually it is essential to introduce artificial boundaries to limit the area of computation, due to the finite memory limitation of the computer. The boundary conditions at these artificial boundaries are used to guarantee a unique and well-posed solution to the PDE problems within the computational domain. These artificial boundary conditions are expected to affect the solution in a manner such that it closely approximates the free space solution which exists in the absence of these boundaries. Thus, the amplitudes of waves reflected from these artificial boundaries are expected to be minimized.

In order to avoid (or reduce) the edge reflection contamination from the computational domain boundaries, one may make the model sufficiently large so that the arrival times of these edge reflections are out of the time window of interest. Although it would be completely free of any edge contamination, this option is very costly in terms of CPU time and memory. Another type of more practical choices is to implement non-reflecting and/or absorbing boundary conditions at the computational domain edges. The "non-reflecting" here means that a set of equations are imposed only at those grid points on the edges of the computational domain that mathematically absorb almost all the outgoing energy [11, 38]. Unfortunately, according to Engquist *et al* in [11], the perfectly absorbing conditions necessarily have to be nonlocal in both space and time and thus are not useful for practical calculations. Hence, in practice, some approximations have to be derived to approach the theoretical nonlocal boundary condition, at the cost of some energy being reflected back into the computational domain. Often an absorbing region is used to dampen the outgoing energy by surrounding the main domain with a narrow damping strip which can drastically dampen the traversing waves. The damping factor should go from zero in the interior of the strip to a maximum value at the edge of the whole model. This technique is more efficient than the large "free-space" one, but still involves a non-trivial amount of extra wave propagation. The third alternative to absorb the edge reflection is to set some boundary conditions which can be used to cancel the edge reflection thoroughly [18]. As described in [18], Dirichlet and Neumann boundary conditions can be used to simulate rigid and free surface conditions respectively. Their combination can be used to effectively and completely cancel the first order edge reflections. Unlike other schemes, it is independent of the incident angle. However, in order to obtain the combined solutions of both boundary conditions, 2^i wave propagation problems, (i is the number of boundaries involved) one for each combination of Dirichlet and Neumann conditions applied, have to be solved and linearly superimposed. Therefore, this perfectly-absorbing scheme is also very costly in computing time, although it's simpler.

In this chapter, the general approximation methods for the non-reflecting boundary conditions will be briefly described and the method of incorporating the 2nd order absorbing approximation equations at the edges into the new polynomial time-marching scheme will be introduced.

5.1 Non-reflecting Boundary Condition Approximation

Consider the scalar wave equation,

$$w_{tt} = w_{xx} + w_{yy}, \quad t, x \geq 0 \tag{5.1.1}$$

According to Engquist *et al* [11], the perfectly absorbing boundary condition for the above equation at *x=0* can be written in the form

$$\left(\frac{\partial}{\partial x} - \rho\left(\frac{\partial}{\partial y}, \frac{\partial}{\partial t}\right)\sqrt{\frac{\partial^2}{\partial t^2} - \frac{\partial^2}{\partial y^2}}\right) w|_{x=0} = 0 \tag{5.1.2}$$

Here $\rho(\xi, \omega)$ is a smooth function homogeneous of degree zero for $|\xi| + |\omega|$ large with support in $\xi^2 > \omega^2$ for (ξ, ω) large and identically one on a neighborhood of the support of $\hat{w}(0, \xi, \omega)$ (the Fourier transform of $w(x, y, t)$ at $x = 0$).

Unfortunately, the perfectly absorbing boundary condition above in (5.1.2) is necessarily *nonlocal in both space and time.* This boundary condition is impractical from a computational point of view since to advance one time level at a single point requires information from all previous times over the entire boundary. Thus, many highly absorbing local approximations to (5.1.2) have been developed. Necessarily, the boundary condition approximations need to satisfy the following two criteria [11]:

1. These boundary conditions are local. This is essential for reasonable control of the operation count.
2. The boundary conditions lead to a well-posed mixed boundary value problem for the wave equation. This condition guarantees the stability of the solution.

The first and second order approximations based on Taylor or Padé expansion are [11]:

1st Order Approximation :

$$\left(\frac{\partial}{\partial x} - \frac{\partial}{\partial t}\right) w|_{x=0} = 0 \tag{5.1.3}$$

2nd Order Approximation :

$$\left(\frac{\partial^2}{\partial x \partial t} - \frac{\partial^2}{\partial t^2} + \frac{1}{2}\frac{\partial^2}{\partial y^2}\right) w|_{x=0} = 0 \tag{5.1.4}$$

Both of the above two approximations have been verified to be well-posed in [11, 53]. The first order one approximates absorption at the normal incidence and it's perfect in the one-dimensional space case. Actually, (5.1.3) is the standard left-traveling one-way wave equation, which represents the left-traveling-only propagation at the boundary *x=0.*

In [38] (Renaut 1992), Renaut reviewed some of the higher order methods currently used for solving the absorbing boundary conditions for the two-dimensional scalar wave equation, such as Lindman [27] and Engquist and Majda [11]. She showed that the extension to higher orders is neither immediately obvious nor unique, as there are many different ways one can discretize the derived absorbing boundary condition. She also proposed a new high-order absorbing boundary condition approximation, which reserves the 2nd order spatial derivative as the highest order derivative in the formula. For the two-dimensional scalar wave equation ,

$$u_{tt} = c^2(u_{xx} + u_{yy}) \ , \tag{5.1.5}$$

the absorbing boundary condition at *x=0* can be approximated by a set of equations (as opposed to one)

$$u_x - \frac{p_0}{c} u_t = \frac{p_0}{c} \sum_{i=1}^{N} h_i, \tag{5.1.6}$$

where

$$\frac{\partial^2 h_i}{\partial t^2} - \beta_i c^2 \frac{\partial^2 h_i}{\partial y_i^2} = c^2 \alpha_i \frac{\partial^3 u}{\partial y^2 \partial t} \ , \qquad i = 1, \cdots, N$$

to be solved at the boundary. Here $p_0, \ \alpha_i, \ \beta_i$ are approximation coefficients. The derivation of $p_0, \ \alpha_i, \ \beta_i$ is described by Renaut in [38]. In the above equation, to increase the degree of approximation, one only needs to add some new coefficients and increase the loop index (i.e. increase the degree of interpolation, but not the highest order derivative). This property is quite useful in the finite-difference implementations.

In the polynomial time-marching scheme, due to the complication of matrix manipulations, only the standard 2nd-order absorbing approximation (5.1.4) will be considered as an example for the absorbing boundary condition implementation in the new time-marching scheme.

5.2 Polynomial Time-marching With Non-reflecting Boundary Conditions in One-dimensional Case

In the polynomial time-marching, as discussed in Chapter 4, one needs to incorporate the time-dependent absorbing boundary conditions into the spatial operator. Here, one can utilize the 1st order time derivative term in vector form (4.3.2)-(4.3.4). Let's consider a one-dimensional scalar wave equation with perfectly absorbing boundaries at the two ends:

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}\ , & -1 < x < +1, t > 0 \\ \frac{\partial u}{\partial t} = c\frac{\partial u}{\partial x}|_{x=-1}\ ,\ \ \frac{\partial u}{\partial t} = -c\frac{\partial u}{\partial x}|_{x=+1}\ , & t \geq 0 \\ u(x,t=0) = U_0(x),\ \ u_t(x,t=0) = 0\ , & -1 \leq x \leq +1 \end{cases} \tag{5.2.1}$$

Let $v = u_t$, collocating the data at *Gauss-Lobatto* points (same as the manipulation in (4.3.1)). Then the boundary conditions can be exactly represented at the end points *x_0=-1, x_N=+1*:

$$\begin{cases} (u_0)_x = \frac{1}{c}(u_0)_t = \frac{1}{c}v_0 \\ (u_N)_x = -\frac{1}{c}(u_N)_t = -\frac{1}{c}v_N \end{cases} \tag{5.2.2}$$

where u_0, u_N, v_0, v_N are collocated data at the end points *x_0=-1, x_N=+1*. Therefore, the derivatives can be represented as

$$\left[\frac{\partial u}{\partial x}\right]_{(N+1)\times 1} = \begin{bmatrix} u'_0 \\ \vdots \\ u'_N \end{bmatrix} = \begin{bmatrix} \frac{1}{c}v_0 \\ u'_1 \\ \vdots \\ u'_{N-1} \\ -\frac{1}{c}v_N \end{bmatrix} \tag{5.2.3}$$

$$= \begin{bmatrix} 0 & 0 & \cdots & 0 \\ d_{1,0} & d_{1,1} & \cdots & d_{1,N} \\ \vdots & \vdots & \cdots & \vdots \\ d_{N-1,0} & d_{N-1,1} & \cdots & d_{N-1,N} \\ 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \\ u_N \end{bmatrix} + \begin{bmatrix} \frac{1}{c} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & -\frac{1}{c} \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{N-1} \\ v_N \end{bmatrix}$$

$$\left[\frac{\partial^2 u}{\partial x^2}\right]_{(N+1)\times 1} = [D_1]_{(N+1)^2} \begin{bmatrix} u'_0 \\ \vdots \\ u'_N \end{bmatrix} = [D_1]_{(N+1)^2}[\widetilde{D}_1 \quad B]\begin{bmatrix} \bar{u} \\ \bar{v} \end{bmatrix}$$

$$\text{where } \widetilde{D}_1 = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ d_{1,0} & d_{1,1} & \cdots & d_{1,N} \\ \vdots & \vdots & \cdots & \vdots \\ d_{N-1,0} & d_{N-1,1} & \cdots & d_{N-1,N} \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{1}{c} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & -\frac{1}{c} \end{bmatrix}$$

The matrix D_1 and the elements d_{ij} are defined in (4.1.3). The vectors $\bar{u}$, $\bar{v}$ are the same as in Section 4.3. Following the same procedure as described in (4.3.2)-(4.3.4), one can obtain the general symbolic operator solution including the absorbing boundary condition as below:

$$\begin{bmatrix}\bar{u}\\ \bar{v}\end{bmatrix} = \exp\left(G_N t\right)\begin{bmatrix}\bar{u}(0)\\ \bar{v}(0)\end{bmatrix}, \tag{5.2.4}$$

$$\text{Here , } [G_N]_{(2N+2)^2} = \begin{bmatrix} 0 & I_{N+1} \\ c^2 D_1 \cdot \widetilde{D}_1 & c^2 D_1 \cdot B\end{bmatrix}_{(2N+2)^2}$$

The matrices $\widetilde{D_1}$ and B are defined in (5.2.3). By a method similar to that described in (4.3.5), one can also get the eigenvalue bounds of $[G_N]_{(2N+2)^2}$ directly from those of $[D_1 \cdot B]$ and $\left[D_1 \cdot \widetilde{D}_1\right]$. Let l is an eigenvalue of the operator $[G_N]_{(2N+2)^2}$, then one can obtain

$$[G_N]\begin{pmatrix}X\\Y\end{pmatrix} = \begin{bmatrix}0 & I\\ D_1 \cdot \widetilde{D}_1 & D_1 \cdot B\end{bmatrix}\begin{pmatrix}X\\Y\end{pmatrix} = l\begin{pmatrix}X\\Y\end{pmatrix} \tag{5.2.5}$$

$$\begin{cases} IY = lX \\ \left[D_1 \cdot \widetilde{D}_1\right]X + [D_1 \cdot B]Y = lY \end{cases}$$

$$\therefore l = \frac{\hat{d} \pm \sqrt{\hat{d}^2 + 4\hat{g}}}{2}$$

Here $\hat{d}$, $\hat{g}$ respectively are the eigenvalues of $\left[D_1 \cdot \widetilde{D}_1\right]$ and $[D_1 \cdot B]$. Then the eigenvalue bound of $[G_N]_{(2N+2)^2}$ $|l_{max}|$ can be obtained from (5.2.5). Thus, the maximum eigenvalue (spectral radius) of the operator $[G_N]_{(2N+2)^2}$ can be calculated from the eigenvalues of $\left[D_1 \cdot \widetilde{D}_1\right]$ and $[D_1 \cdot B]$. Direct eigenvalue computation for the bigger matrix$[G_N]_{(2N+2)^2}$ is unnecessary.

Figure 5.1 and Figure 5.2 show some numerical results for the one-dimensional absorbing boundary condition approximation when N=32 and N=64. The initial pulses are all Gaussian, the time step is 0.2 and the wave velocity c=1. The time-marching relative error control factor is less than 10^{-5}. The analytical reference can be derived from

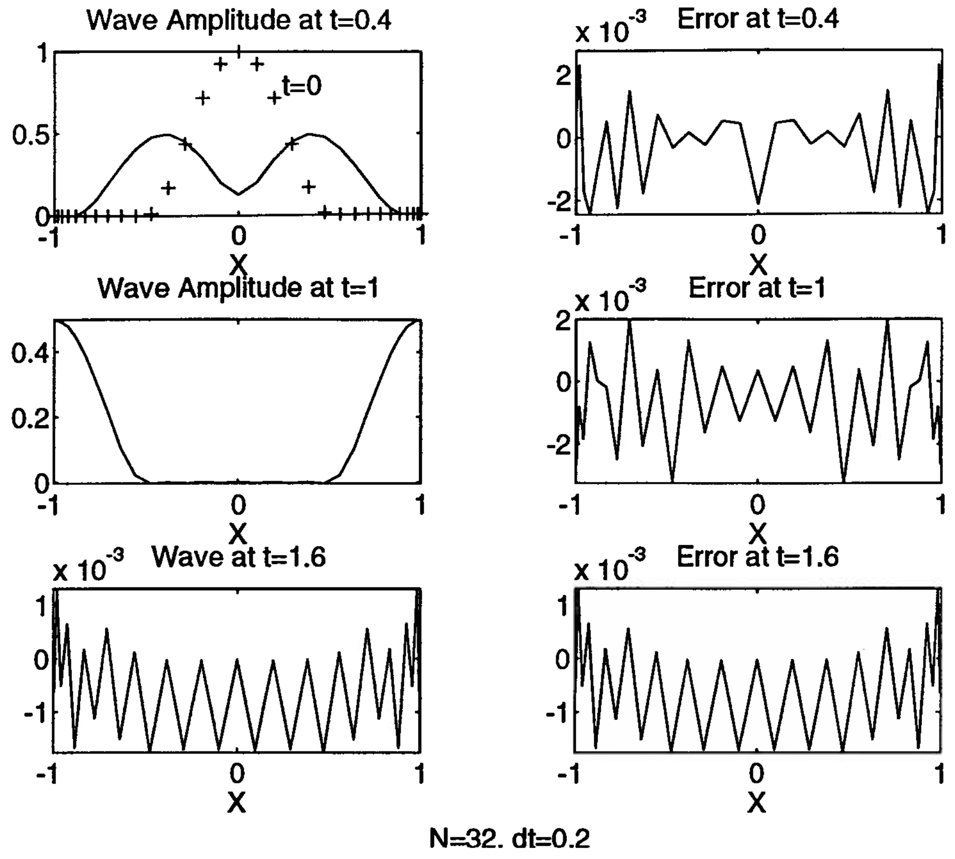


Figure 5.1 One-dimension Absorbing Boundary Condition Numerical Results (N=32)

the physical interpretation of one-dimensional scalar wave propagation. The analytical solution of the wave at t=1.6 in both figures should be a flat zero line because the pulse has transmitted out of the computational domain. Therefore, what is shown in the left column plots (t=1.6) in Figure 5.1 and Figure 5.2 is just computational error, which is equivalent to the corresponding right column plot.

5.3 Spectral Accuracy Discussions of the Polynomial Time-marching Scheme in One Dimension

As Fornberg stated in [14], the pseudospectral method performs in many situations far better than present theory would suggest. As for the spectral accuracy for general initial-boundary value problems, no general theory is available that is as readily applicable as those for finite-differences. In Chapter 2, I have shown some previously existing

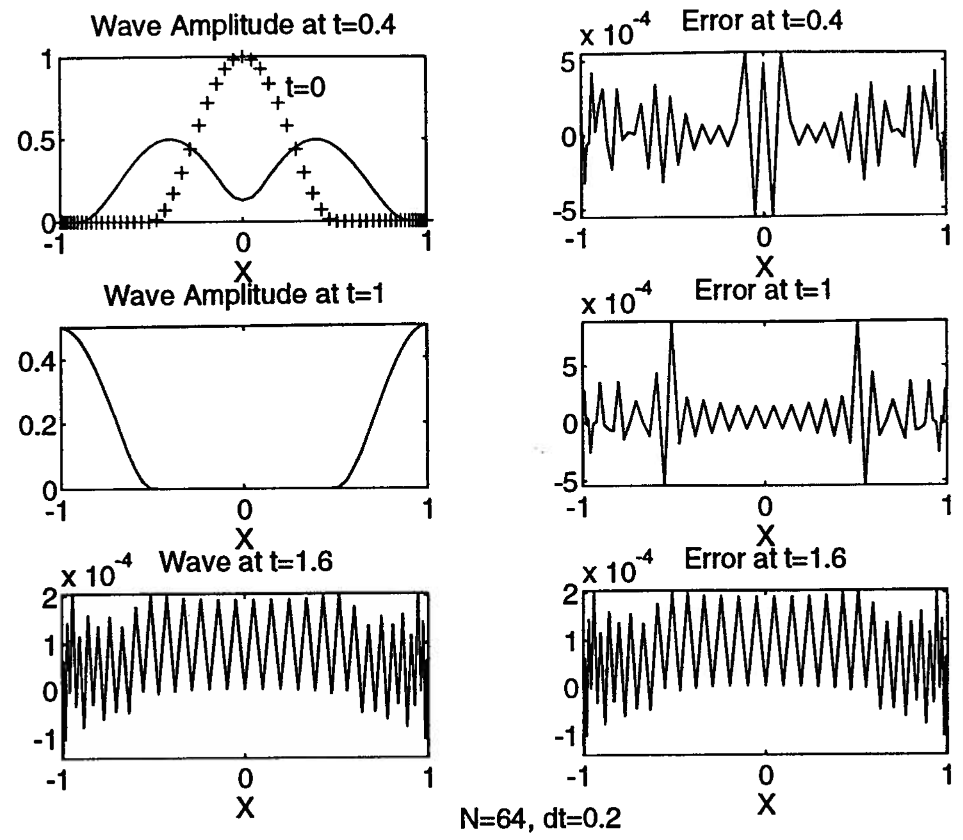


Figure 5.2 One-dimension Absorbing Boundary Condition Numerical Results (N=64)

spectral accuracy results for Fourier pseudospectral method in elastic wave equations, presented by Fornberg in [14, 15]. To investigate the spectral accuracy of the Chebyshev pseudospectral in polynomial time-marching scheme, some one-dimensional experimental calculations have been performed to obtain the data in the following tables.

Table 5.1 Time-marching Errors for the One-dimensional Wave Equation with Homogeneous Neumann Boundary Conditions (homogeneous medium)

N	Mean Error	Max. Error
4	6.041E-1	7.010E-1
8	2.788E-3	3.048E-3
12	1.552E-6	1.593E-6
16	2.270E-10	2.297E-10
20	1.099E-13	1.634E-13

Table 5.2 Time-marching Errors for the One-dimensional Wave Equation with Homogeneous Dirichlet Boundary Conditions (homogeneous medium)

N	Mean Error	Max. Error
4	1.641E-2	4.102E-2
8	5.568E-5	1.050E-4
12	1.841E-8	6.150E-8
16	1.423E-12	4.535E-12
20	2.806E-14	5.684E-14

Here, the computation is based on Equation (4.3.1) with $U_0(x) = \sin(\pi x)$ for the Dirichlet case and with Equation (4.3.6) for the Neumann case. A time step of Δt*=0.01* is used. The errors are computed from the 100th time step wave distribution, compared with the corresponding analytical solution. From the error data listed in the tables, it is clear that the error converges to zero fairly fast. In the Neumann case (Table 5.1), the mean error ratio of N=8 to N=4 is about 0.0046, which roughly shows an error at the order of $O\left(\left(\frac{1}{N}\right)^8\right)$, i.e. 8th order finite-difference scheme accuracy. The mean error ratio of N=16 to N=8 is about 8.14e-8, which roughly shows an error at the order of $O\left(\left(\frac{1}{N}\right)^{24}\right)$, i.e. 25th order finite-difference scheme accuracy. In Dirichlet case (Table 5.2), the mean error ratio of N=8 to N=4 is approximately 0.0034. It also shows an 8th finite-difference order of error convergence. The mean error ratio of N=16 to N=8 is 2.56e-8, which is roughly equivalent to an error $O\left(\left(\frac{1}{N}\right)^{25}\right)$, i.e. 25th order finite-difference accuracy. Both Dirichlet and Neumann case results show that when the number of grid points N increases, the order of error convergence also increases. Although it is claimed by J.P. Boyd in [5] that in elliptic equation case the spectral method has an error of $O\left(\left(\frac{1}{N}\right)^N\right)$, in this experiment, probably due to the boundary condition interaction and also due to the relatively low accuracy of computing Schwarz-Christoffel at small *N*s, the results do not illustrate such kind of "exponential order" convergence. In fact, due to the limitation of computer double precision format, in practice it is impossible to get this order of convergence for N>10. However, the experimental results do show that the degree of error convergence increases with N.

Table 5.3 Time-marching Errors for the One-dimensional Wave Equation with Absorbing Boundary Conditions (homogeneous medium)

N	Mean Error	Max. Error
64	5.629E-5	2.038E-4
128	5.374E-8	1.401E-7

In Table 5.3, the mean error ratio from N=128 to N=64 is 9.55e-4, which shows an error convergence order of $O\left(\left(\frac{1}{N}\right)^{10}\right)$. This error convergent degree is approximately about the same as that for 10th order of finite-difference. Probably due to the operator complexity in this case, the minimum error caused by Schwarz-Christoffel conformal mapping is around $O(10^{-9})$. Thus, the overall accuracy is certainly limited by this order. A time step of $\Delta t=0.01$ is used in this experiment. The errors are computed from the wave at 240th time step. A cosine initial distribution is used.

Table 5.4 Time-marching Errors for the One-dimensional Wave Equation with Absorbing Boundary Conditions (varying medium)

N	Mean Error	Max. Error
32	1.514E-3	5.215E-3
64	6.053E-4	2.186E-3
128	2.936E-4	1.123E-3

Finally, a step discontinuity with the absorbing boundary condition is analyzed to see the effect of the medium variance. The experimental data listed in Table 5.4 show that in varying medium (wave velocity jumping from 0.5 to 1.0 with one middle point transition) the performance is quite poor ($O\left(\frac{1}{N}\right)$) due to the varying medium coefficient's affect. Severe accuracy degradation is caused by the step discontinuity. In the polynomial time-marching scheme, since the differential operator is applied repeatedly over the medium space, some high order derivatives of the medium variation contaminate the overall solution. In the computation of Table 5.4 data, the time step remains as Δt=*0.01* and the errors are computed from the wave at 380th time step. Also, a cosine initial pulse is used here. For two-dimensional equations with absorbing boundary conditions, due to the limitation of the 2nd order absorbing approximation, no result with accuracy higher than 2nd order can be expected.

5.4 Polynomial Time-marching With Non-reflecting Boundary Conditions in Two Dimension

In the two-dimensional case, in order to get a good absorbing approximation, one needs to use at least the 2nd order absorbing boundary condition approximation (5.1.4), rather than just the 1st order one for normal incidence. Consider a two-dimensional scalar wave equation within the rectangular domain $x, y \in [-1, +1]$. The boundary condition at the edge of *x=-1* is a 2nd order absorbing approximation. The other 3 edges (*x=+1, y=+/-1*) are all homogeneous Dirichlet.

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} = c^2\frac{\partial^2 u}{\partial x^2} + c^2\frac{\partial^2 u}{\partial y^2}\,, & x, y \in (-1, +1),\ t > 0 \\ u(x,y,0) = U_0(x,y),\ u_t(x,y,0) = 0, & x, y \in [-1, +1] \\ u(x=1,y,t) = 0,\ u(x,y=\pm 1,t) = 0, & t > 0 \\ u_{tt} = cu_{xt} + \frac{1}{2}c^2 u_{yy} & x = -1, y \in (-1,1) \end{cases} \tag{5.4.1}$$

To examine and compare the results, another two simulation results need to be linearly superimposed to get a perfectly absorbing at the edge of *x=-1*. One is with all 4 edges of homogeneous Dirichlet, and the second is with homogeneous Neumann at the edge of *x=-1* but the rest all homogeneous Dirichlet. According to [18], their linear combination will generate the completely absorbing boundary at the edge of *x=-1*. Discretizing *N, M* respectively along *x, y* at *Gauss-Lobatto* points, I have

$$\begin{cases} x_i = \cos\left(\frac{\pi i}{N}\right)\,, & i = 0, \cdots, N \\ y_j = \cos\left(\frac{\pi j}{M}\right)\,, & j = 0, \cdots, M \end{cases} \tag{5.4.2}$$

Similarly, setting v=u_t, based on the two-dimensional results in (4.5.1), one has the following matrix expression:

$$\begin{bmatrix} U \\ V \end{bmatrix}_t = \begin{bmatrix} 0 & I \\ c^2 G & c\tilde{D}_x^{(1)} \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix} \tag{5.4.3}$$

$$\text{where } GU = \left[\widetilde{D}_x^{(2)}\right]_{N\times N} U_{N\times(M-1)} + B_{N^2} U_{N\times(M-1)} \left[\widetilde{D}_y^{(2)}\right]_{(M-1)^2}$$

$$\left[\widetilde{D}_x^{(1)}\right]_{N\times N} = \begin{bmatrix} & 0_{(N-1)\times N} & & \\ d_{N,1}^{(1)} & d_{N,2}^{(1)} & \cdots & d_{N,N}^{(1)} \end{bmatrix}_{N\times N}$$

$$\left[\widetilde{D}_x^{(2)}\right]_{N\times N} = \begin{bmatrix} d_{1,1}^{(2)} & d_{1,2}^{(2)} & \cdots & d_{1,N}^{(2)} \\ \vdots & \vdots & \cdots & \vdots \\ d_{N-1,1}^{(2)} & d_{N-1,2}^{(2)} & \cdots & d_{N-1,N}^{(2)} \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{N\times N}$$

$$\left[\widetilde{D}_y^{(2)}\right]_{(M-1)^2} = \begin{bmatrix} \bar{d}_{1,1} & \cdots & \bar{d}_{1,M-1} \\ \vdots & \cdots & \vdots \\ \bar{d}_{M-1,1} & \cdots & \bar{d}_{M-1,M-1} \end{bmatrix}_{(M-1)^2}$$

$$B_{N\times N} = \begin{bmatrix} \frac{1}{2} & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}_{N\times N}$$

Here, $d_{i,j}^{(1)}$, $d_{i,j}^{(2)}$ are respectively the elements of D_1 , D_2 defined in (4.1.3) and $\bar{d}_{i,j} = \left[D_2^T\right]_{i,j}$. The first row, first and last columns of the solution matrix $u_{0,j}$, $u_{i,0}$, $u_{i,M}$ are all zero, due to the 3–edge homogeneous Dirichlet conditions.

In order to obtain the absorbing effect in a few time steps, and to avoid the reflection interference from the other full-reflection edges, the initial Gaussian pulse center is located very close to the *x=-1* boundary. The initial pulse is tapered to zero at the near-edge elements. The initial pulse is generated from the expression

$$u(x,y,0) = 0.1 * \exp\left(-\left((x-x0)^2 + y^2\right)/\Delta\right) \tag{5.4.4}$$

In this numerical experiment, the wave velocity c is 1.0, the time step Δt is 0.01, which corresponds to a time step 3.33×10^{-11} second if the wave propagation speed is light speed c=3×10^8 m/sec. This time step is comparable to 2.5×10^{-11} second time step used in [50]. In this experiment, the number of collocation grid points in both X and Y is 32, x0=-0.5290, that is, the peak of the initial pulse will pass the x=-1 boundary at 47th time step. The global error and normalized reflection are also defined in the same way as in

[50]. At each time step the difference is defined to be

$$D(i,j) = u_{\text{appr.}}(i,j) - 0.5 * \left(u_{\text{dirich}}(i,j) + u_{\text{neum}}(i,j)\right) \tag{5.4.5}$$

The global error is defined as

$$E = \sum_i \sum_j D^2(i,j) \tag{5.4.6}$$

The normalized reflection is the reflection occurring at one cell above the *x=-1* absorbing boundary when the pulse peak passes the boundary. It is normalized by the maximum value of the pulse along the boundary at that moment (47th time step).

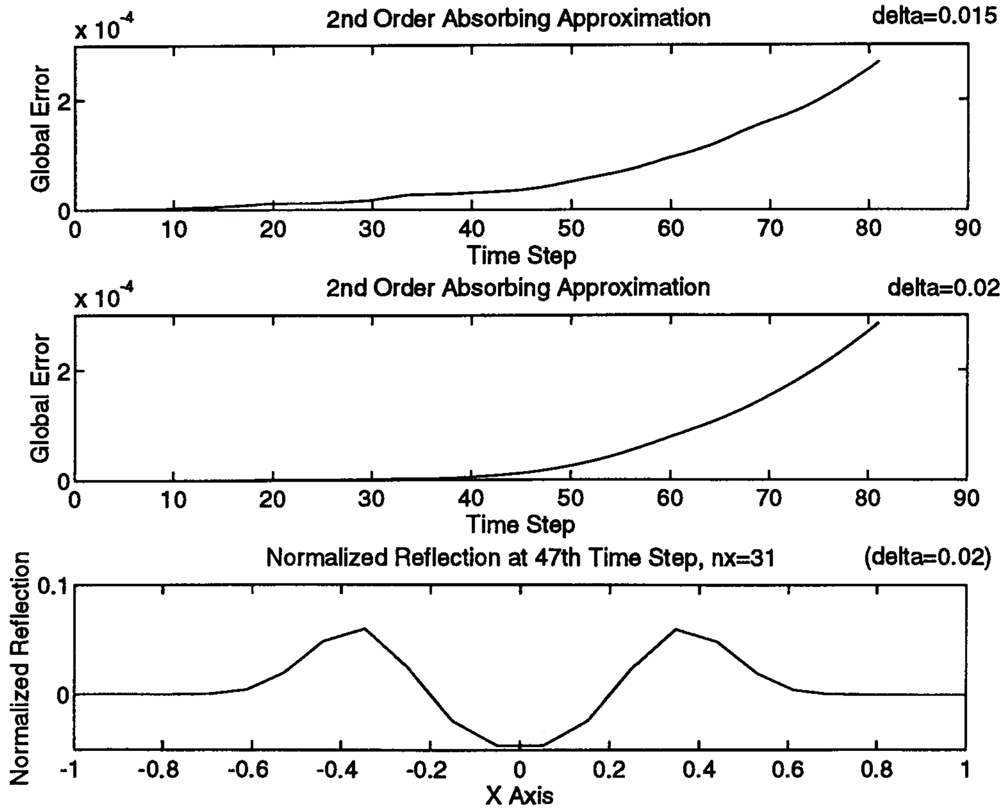


Figure 5.3 Two-dimension Absorbing Approximation Numerical Results (Grid points 32 in both X and Y, dt=0.01)

The global errors of both cases in this experiment are comparable to those reported in reference [50] (less than 0.0003 within 80 time steps). When Δ=0.015, the initial pulse is steeper and more aliasing may occur, thus the results are worse. Since the same polynomial time-marching and Chebyshev collocation scheme are used here for both the approximated and the perfect absorber, the global error only demonstrates the accuracy of the absorbing approximation, which is limited by the order of the absorber accuracy, in this case second order, and is determined principally by the magnitude of the reflected waves propagating into the computational domain.

Chapter 6 Simulation and Visualization of Model Wave Scattering

Numerically simulating wave field propagation in heterogeneous media has become one of the major tools for studying seismograms. Many different numerical algorithms have been applied in this area. Particularly, finite-difference time-domain algorithms have been intensively implemented on supercomputers for seismic forward modeling [25, 57, 55], which starts with an assumed earth model to generate the wave field by solving the elastic wave equations.

6.1 2–D Seismic Reflection Modeling

The two-dimensional *SH*-wave (horizontal shear wave) propagation in a heterogeneous medium can be described by the following elastic equation:

$$\rho(x,z)\frac{\partial^2 v}{\partial t^2} = \frac{\partial}{\partial x}\left[\mu(x,z)\frac{\partial v}{\partial x}\right] + \frac{\partial}{\partial z}\left[\mu(x,z)\frac{\partial v}{\partial z}\right] \tag{6.1.1}$$

where *ρ(x, z)* is the density and *μ(x, z)* is the shear modulus at the point (x, z) of the medium and $v(x,z)$ is the horizontal displacement along the *y* axis. The medium is supposed to be in equilibrium at time t=0. Hence, all the initial conditions are zero everywhere in the medium. The wave propagation is caused by a point source excitation (a short pulse under the earth surface) in the medium. The distribution geometry of the medium is a rather simplified salt dome model used by J. Virieux in [57], illustrated in Figure 6.1. The positions *S1, S2, S3* are three possible source locations (one source

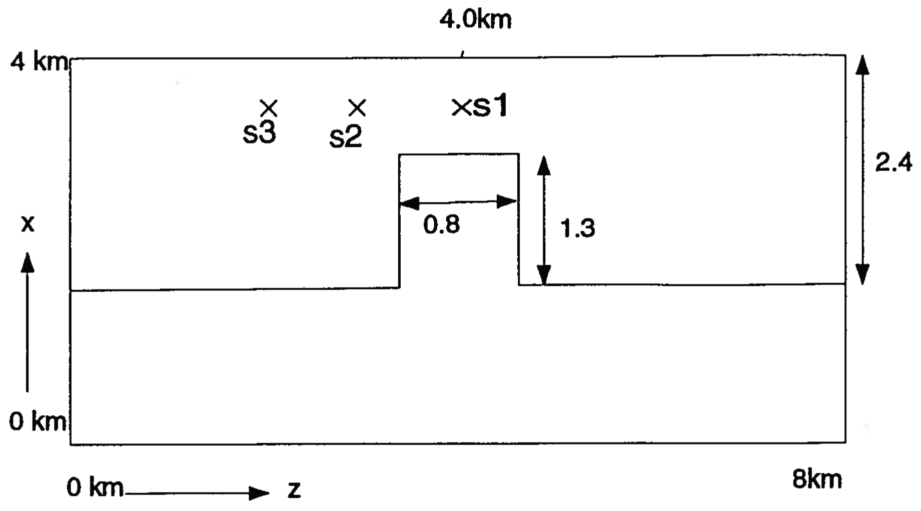


Figure 6.1 Geometry of Salt Dome for Polynomial Time-marching Modeling

at a time). The top edge is free earth surface, so the boundary condition is stress-free, i.e. Neumann condition for the displacement v. The other three edges are all artificial domain boundaries for computational purposes. They are all set as non-reflecting and the 2nd order absorbing approximation (5.1.4) is applied there as described in Chapter 5 and [28]. For ease of implementation, when the absorbing approximation is applied to the left and the right side of the domain illustrated in Figure 6.1, a fairly homogeneous medium is always assumed along the boundaries so that the simple form of (5.1.4) can be used. The boundary conditions can be expressed in the following form:

$$\begin{cases} v_{tt} = cv_{xt} + \frac{1}{2}c^2 v_{zz} \ , & x = -1 \text{ (bottom)} \\ \frac{\partial v}{\partial x} = \frac{\partial v}{\partial z} = 0 \ , & x = +1 \text{ (top)} \\ v_{tt} = cv_{tz} + \frac{1}{2}c^2 v_{xx} \ , & z = -1 \text{ (left)} \\ v_{tt} = -cv_{tz} + \frac{1}{2}c^2 v_{xx} \ , & z = +1 \text{ (right)} \end{cases} \tag{6.1.2}$$

$$\text{where } c = \sqrt{\frac{\mu}{\rho}} \text{ is wave velocity}$$

Similar to the manipulations in the previous chapters by introducing the 1st order time derivative of v, the equation (6.1.1) with the above boundary conditions incorporated can be configured in the following matrix operator form:

$$\begin{pmatrix} v \\ v_t \end{pmatrix}_t = \begin{pmatrix} 0 & I \\ \text{op1} & \text{op2} \end{pmatrix} \begin{pmatrix} v \\ v_t \end{pmatrix} \tag{6.1.3}$$

where I is an identity operator. Under the discretization of the scaled *Gauss-Lobatto* grid points [61], the node positions are given by

$$\begin{cases} x_i = \frac{X_{max}+X_{min}}{2} + \frac{X_{max}-X_{min}}{2}\cos\left(\frac{\pi i}{N}\right), & i = 0,\cdots,N \\ z_j = \frac{Z_{max}+Z_{min}}{2} + \frac{Z_{max}-Z_{min}}{2}\cos\left(\frac{\pi j}{M}\right), & j = 0,\cdots,M \end{cases} \tag{6.1.4}$$
$$\text{where } Z_{min} = X_{min} = 0\ m,\ \ Z_{max} = 8000\ m,\ \ X_{max} = 4000\ m$$

The operator 1 *op1* and operator 2 *op2* can respectively represented as

$$\begin{aligned} \text{op1}\{v\} &= \frac{1}{\rho}\{r_x^2[D_0](\mu)[D_1][v][A_2] + r_z^2[A_1][v][D_z](\mu)[D_2]\}, \\ \text{op2}\{v_t\} &= (c)\{r_x[D_4][v_t][B_1] + r_z[v_t][D_5]\} \end{aligned} \tag{6.1.5}$$

The detailed derivation procedure is described in Appendix C. Here, ( )s are all two-dimensional arrays and [ ]s are matrices. The operator 1 reflects the 2nd-order equation (6.1.1), Neumann boundary condition at x=-1 and the 2nd-order derivatives in (6.1.2). The operator 2 reflects the 1st order derivatives in (6.1.2). The corresponding matrices are defined below as:

$$D_0 = \begin{bmatrix} d^x_{0,1} & \cdots & d^x_{0,N} \\ \vdots & \cdots & \vdots \\ d^x_{N-1,1} & \cdots & d^x_{N-1,N} \\ 0 & \cdots & 0 \end{bmatrix}_{(N+1)\times N}, \quad D_1 = \begin{bmatrix} d^x_{1,0} & \cdots & d^x_{1,N} \\ \vdots & \cdots & \vdots \\ d^x_{N,0} & \cdots & d^x_{N,N} \end{bmatrix}_{N\times(N+1)} \tag{6.1.6}$$

$$D_2 = \begin{bmatrix} 0 & d^z_{0,1} & \cdots & d^z_{0,M-1} & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & d^z_{M,1} & \cdots & d^z_{M,M-1} & 0 \end{bmatrix}_{(M+1)^2}, \quad A_1 = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & 0 & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0.5 \end{bmatrix}_{(N+1)^2}$$

$$D_4 = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \cdots & \vdots \\ 0 & \cdots & 0 \\ d^x_{N,0} & \cdots & d^x_{N,N} \end{bmatrix}_{(N+1)^2}, \quad D_5 = \begin{bmatrix} -d^z_{0,0} & 0 & \cdots & 0 & +d^z_{0,M} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ -d^z_{M,0} & 0 & \cdots & 0 & +d^z_{M,M} \end{bmatrix}_{(M+1)^2}$$

$$A_2 = \begin{bmatrix} 0.5 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & 0 & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0.5 \end{bmatrix}_{(M+1)^2} , \quad B_1 = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & I_{M-1} & \vdots \\ 0 & \cdots & 0 \end{bmatrix}_{(M+1)^2}$$

where d^x_{ij} , d^z_{ij} are the elements of derivative matrices along X and Z, defined in the formula (2.3.8) or (4.1.3). The D_z is simply the full 1st order derivative matrix in Chebyshev collocation. The matrices D_0 and D_1 are configured in the same way as described in (4.2.1) — (4.2.6). In addition, the last row of D_0 is also masked by zeros so that the last row (corresponding to x=-1 in the solution array) is left for the absorbing approximation. The matrix D_2 simply is the 1st order Chebyshev collocation derivative matrix with its 1st and last column masked by zeros due to the absorbing approximation at these two columns (corresponding to z=+1 and z=-1 respectively). The matrices A_1 and A_2 are respectively used to add the absorbing boundary coefficient 0.5 to the related row (the last row) and columns (the 1st and the last columns). The matrix A_1 also masks the 1st row of the solution array to zeros to satisfy the $\frac{\partial v}{\partial z}|_{x=1} = 0$ boundary condition. The matrices D_4 and D_5 are used to represent the first order derivatives in the absorbing approximation. The matrix B_1 is used to avoid the repeat of the two lower corners which have been counted twice at both the horizontal and the vertical edges. The factors r_x, r_z and their squares come from the scaled coordinate. According to Zhao in [61], the Chebyshev collocated derivative under the scaled coordinate (6.1.4) has the following relationship with the standard [-1, +1] interval expression

$$\frac{\partial v}{\partial x} = r\frac{\partial v}{\partial \xi} , \quad x \in [X_{min}, X_{max}] , \ \xi \in [-1, +1] \qquad (6.1.7)$$
$$r = \frac{2}{X_{max} - X_{min}}$$

Consequently, the symbolic solution of the 1st order operator equation system (6.1.3) can be written as :

$$\begin{pmatrix} v \\ v_t \end{pmatrix} = \exp(t\mathbf{A}) \begin{pmatrix} v(0) \\ v_t(0) \end{pmatrix} \tag{6.1.8}$$

Here, the general operator **A** is defined as

$$\mathbf{A} = \begin{pmatrix} 0 & I \\ \text{op1} & \text{op2} \end{pmatrix} \tag{6.1.9}$$

Then, the polynomial time-marching discussed in Chapter 3 can be applied.

In order to obtain the initial distribution, a similar method as used by Alford et al in [3] can be applied here. Assuming the point source is located in a small homogeneous region, and the source excitation is a short pulse, which is the usual case in most seismic applications, before the source term diminishes, the wave propagation resulting from the source excitation is always limited within the homogeneous region. Therefore, the wave distribution within the homogeneous region can be computed by an infinite-space Green's function method. After the source pulse disappears, the wave distribution obtained by the Green's function method can be used as the initial condition for the time marching in (6.1.8).

The medium is composed of two layers. A dome rising from the lower medium (salt) intrudes the upper overburden medium. The medium parameters used here are :

Upper layer : velocity 1500 m/sec, density 2300 kg/m^3 ;

Lower layer : velocity 2250 m/sec, density 2100 kg/m^3 .

The source is an impulsive point source located at 500 m below the free surface and right at the horizontal middle line of the domain (xs = 3500 m , zs = 4000 m). The source is defined as

$$f(t) = (t - t_0) \exp\left(-\alpha(t - t_0)^2\right) \tag{6.1.10}$$

with α=1000 and t_0=0.15. The main advantage of this pulse source is that its bandwidth is limited while its time-period is also short. The negative part caused by this source is especially suited for identifying wave fronts in block diagrams, according to Virieux (1984) [57] and Alford (1974) [3]. The constant factor α governs the time interval w from negative to positive peaks of the function as shown in the following figure This choice of

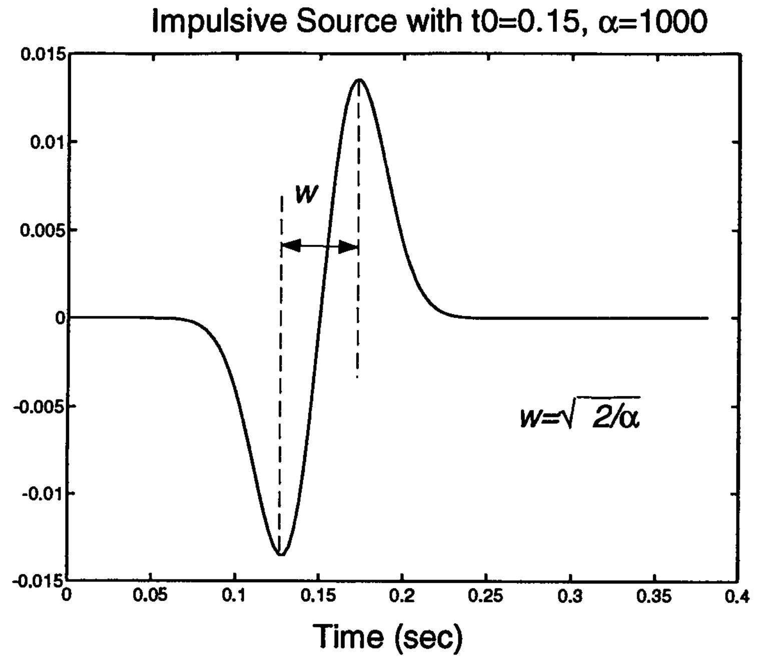


Figure 6.2 Impulsive Source Distribution Time Function

the source function was made as a good compromise between short time duration (regional homogeneous propagation) and a narrow spectrum (reasonable sampling in time). The Fourier transform of (6.1.10) is

$$F(\omega) = \frac{-i\omega}{\alpha}\sqrt{\left(\frac{\pi}{4\alpha}\right)}e^{-\omega^2/4\alpha}e^{-i\omega t_0} \tag{6.1.11}$$

This spectrum provides sufficient damping to control the bandwidth of the source and also limits the bandwidth of the convolution with the infinite-space Green's function due to the following formula

$$v_s(x,z,t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty}\left\{-i\pi H_0^{(2)}\left(\frac{\omega}{c}r\right)F(\omega)\right\}e^{+i\omega t}d\omega \tag{6.1.12}$$

where r is the distance between the observation point and the source point, c is the wave velocity $c = \sqrt{\frac{\mu}{\rho}}$, $H_0^{(2)}$ is the second Hankel function of 0th order. In practice, the inverse Fourier transform integral is performed numerically to get the time-domain wave distribution $v_x(x, z, t)$. Its time derivative can obtained by the 4th order finite-difference approximation. Then they are used as the initial conditions for the time marching. The t_0=0.5 sec and the homogeneous propagation time period [0, 0.3] sec are carefully chosen so that the source values at both *t=0* sec and $t \geq 0.3$ sec are small enough ($\tilde{}10^{-9}$*peak). Therefore, the zero initial condition for the domain can be satisfied and the source excitation virtually disappears when the time-marching begins at t=0.3 sec. With the wave velocity of 1500 m/sec in the upper layer medium, the wavefront caused by the point source excitation (at sx=3500m , sz=4000m) is still within the upper layer homogeneous region (0.3 sec * 1500 m/sec = 450 m).

The highest frequency component can also be obtained from the source constant α as the maximum time-domain sampling interval should be no greater than the largest transition interval *w* (see Figure 6.2). In this case, 65 temporal sampling points are chosen for the 0.3 sec initial propagation period (sample interval 0.0046 sec, far less than *w=0.045 sec* for α=1000 case) for the Green's function computation.

Consequently, the shortest wavelength corresponding to the highest frequency component is given by

$$\lambda_{min} = 2wc_{min} = 2c_{min}\sqrt{\frac{2}{\alpha}} \approx 134 \ \ m \qquad (6.1.13)$$

In order to resolve all the sampled frequency components (up to $f_{max} = 0.5f_s = \frac{1}{2}\sqrt{\frac{\alpha}{2}}$) in the time-domain, the spatial discretization should have enough grid points so that the maximum spatial interval is less than half of the shortest wavelength corresponding to the highest sampled frequency component, i.e. 67 meters. Since the *Gauss-Lobatto* grid used here is unevenly distributed, its maximum interval should be counted here. From (6.1.4), it's easy to get the maximum interval for the scaled *Gauss-Lobatto* grid given by

$$(\Delta x)_{max} = \frac{X_{max} - X_{min}}{2} \sin\left(\frac{\pi}{N}\right) , \qquad (6.1.14)$$
$$(\Delta z)_{max} = \frac{Z_{max} - Z_{min}}{2} \sin\left(\frac{\pi}{M}\right)$$

Hence, N=100 and M=200 were chosen so as to satisfy the conditions of $(\Delta x)_{max} \approx (\Delta z)_{max} \approx 62.8$ m $< \frac{1}{2}\lambda_{min} \approx 67$ m.

The time-marching step size could also be limited to be no greater than w so that the minimum time sampling interval can always be guaranteed. On the other hand, in order to utilize the advantage of the looser stability limitation of the polynomial time-marching, which allows much larger time step size than that in the conventional finite-difference scheme, a time step of 0.001 sec was chosen for the time marching.

As a necessary practical requirement for numerical computation, the discontinuity of the medium parameters at the interface has to be somehow smoothed. Here, a hyperbolic tangent function is used for this purpose so as to effectively reduce the aliasing caused by the discontinuity at the interface.

6.2 Numerical Results and Discussion

In the following pages, some simulation snapshots of the previously defined seismic model are illustrated there. They demonstrate that the validity of the upper free surface boundary condition and other 2nd-order absorbing boundary conditions. The corner and the interface diffraction wavefronts can be clearly observed from these snapshots. Figure 6.6 shows a simpler half space wave scattering model, which demonstrates clearer multiple reflections, including the so-called "conical" wave occurring at the critical incident angle. More animation movies of these wave scattering phenomena are presented on a video tape as an appendix to this thesis.

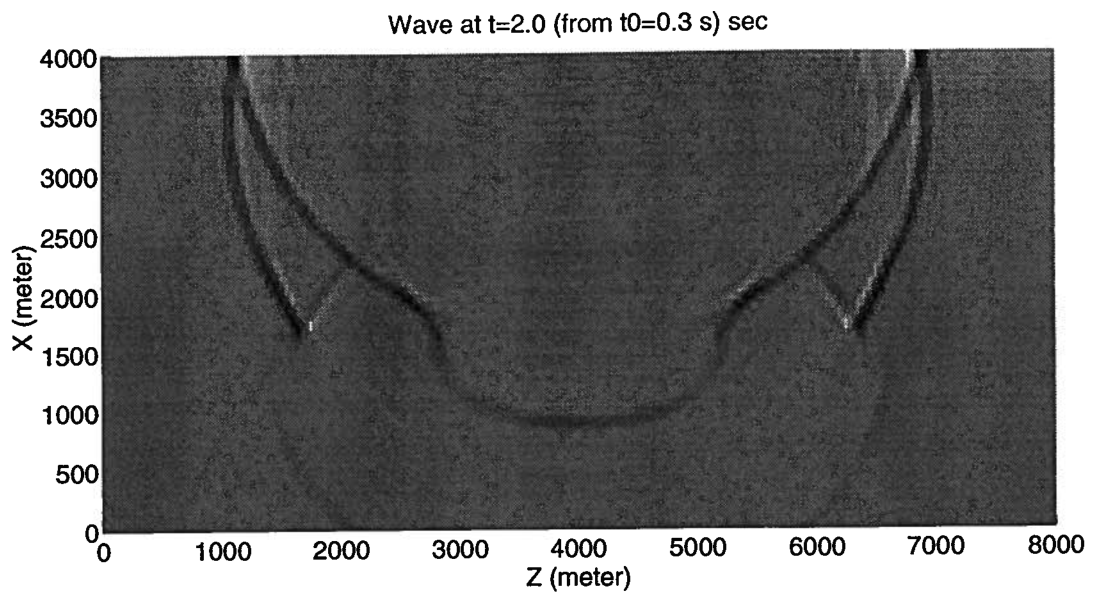


Figure 6.3 Scatter Movie Snapshots When the Source is Located at xs=3500m, zs=4000

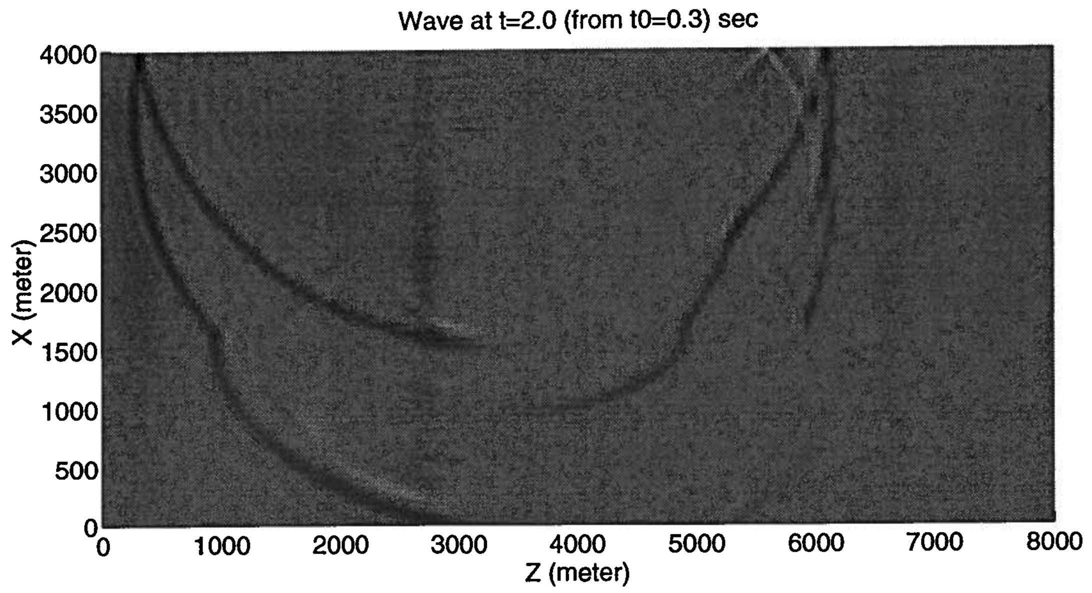


Figure 6.4 Scatter Movie Snapshots When the Source is Located at xs=3500m, zs=3200

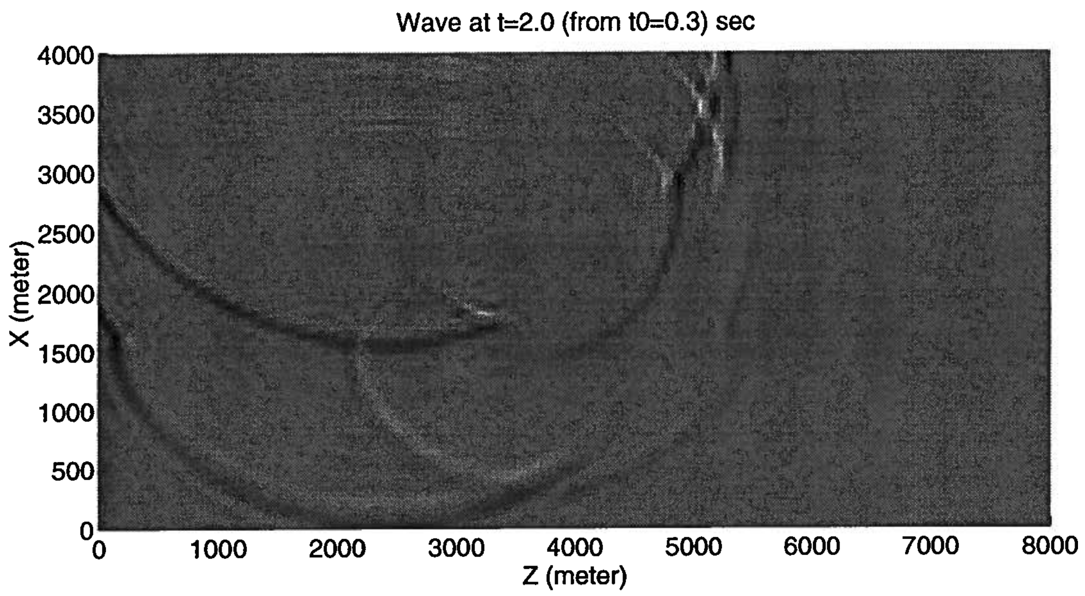


Figure 6.5 Scatter Movie Snapshots When the Source is Located at xs=3500m, zs=2400

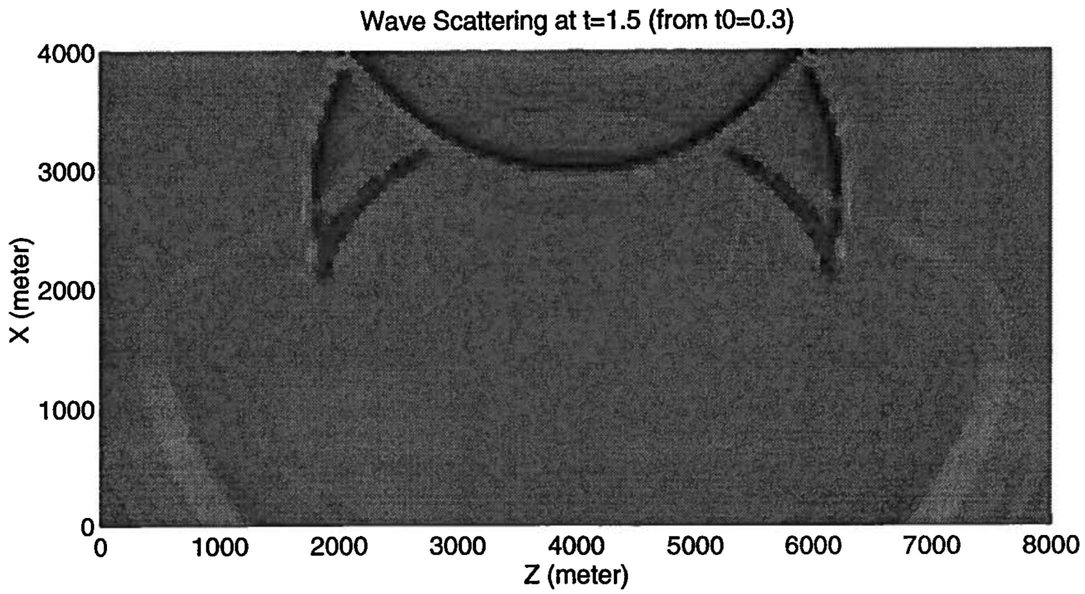


Figure 6.6 Half-space Scattering

6.3 Half-space Wave Scattering

In order to verify the two-dimensional wave scattering results, a half-space scattering example is considered. The analytical time-domain solution for half-space wave scattering is derived by Cagniard in [7] in 1962 and Aki also gives more detailed descriptions in [2]. In practice, due to the numerical implementation difficulties of a step function, it is preferable to use the frequency domain solution as shown in (6.1.12). The reflection can also be computed in the frequency domain by multiplying the incident wave by the asymptotic reflection coefficients V(θ) described by Brekhovskikh and Lysanov in [6] given by

$$V(\theta) = \frac{m\cos\theta - \sqrt{n^2 - \sin^2\theta}}{m\cos\theta + \sqrt{n^2 - \sin^2\theta}}\ , \quad \sin\theta \le n \qquad (6.3.1)$$
$$V(\theta) = \frac{m\cos\theta - i\sqrt{\sin^2\theta - n^2}}{m\cos\theta + i\sqrt{\sin^2\theta - n^2}}\ , \quad \sin\theta > n$$

Here, θ is the incidence angle from the normal at the interface (*y=0*), $m = \rho_2/\rho_1$, $n = c_1/c_2$. ρ_1, ρ_2, c_1, c_2 are the medium densities and shear velocities for the upper and lower medium layers respectively. In this analysis, we use $\rho_1 = \rho_2, c_1 < c_2$. Therefore, we have a real n and *n<1*.

The same source function (6.1.10) is applied here. Thus, the analytical solution for the half space $y < 0$ needs to be convolved with the source function to get the source-excited wave propagation. The analytical solution can be written as

$$v_s(x,z,t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \left\{ -i\pi\left(H_0^{(2)}\left(\frac{\omega}{c}R\right) + H_0^{(2)}\left(\frac{\omega}{c}R_0\right)V(\theta)\right)F(\omega)\right\} e^{+i\omega t} d\omega\ , \quad (6.3.2)$$

where R is the distance from the receiver to the source $(x_0,\ y_0)$, R_0 is the distance from the receiver to the image source at $(x_0,\ -y_0)$ (illustrated in Figure 6.7).

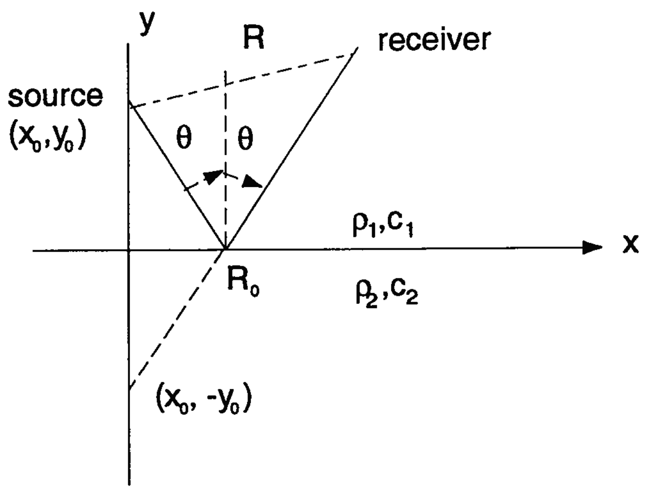


Figure 6.7 Wave Incidence and Reflection for a Half-Space

On the other hand, the band-limited source also bandlimits the analytical solution in numerical form. The "conical" wave is not included in the asymptotic reflection coefficients. The purpose of computing the analytical solution here is to verify the accuracy of the numerical results, in particular, the boundary-smoothing effect. Fortunately, from the numerical solution snapshots, one can note that there are plenty of receiver positions without the "conical" wave involved.

The comparison results are illustrated in Figure 6.9 and Figure 6.10. In the comparison, the medium densities are all set as 1 and the shear velocities are $c_1 = 1, c_2 = 3$ (m/sec). The receivers are located at (x=0, y=-0.094), (x=0, y=-0.187) and (x=0.187, y=–0.094) three positions, as shown in Figure 6.8.

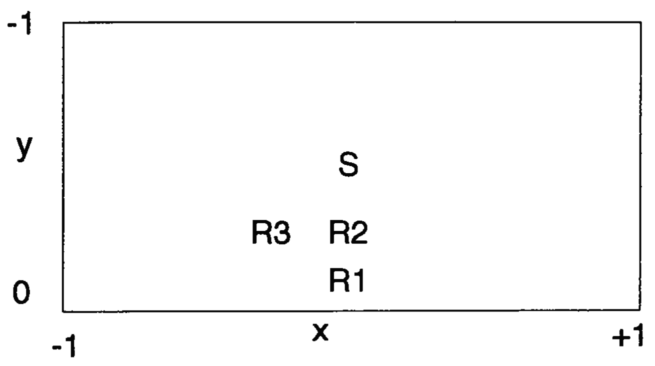


Figure 6.8 Source-Receiver Geometry

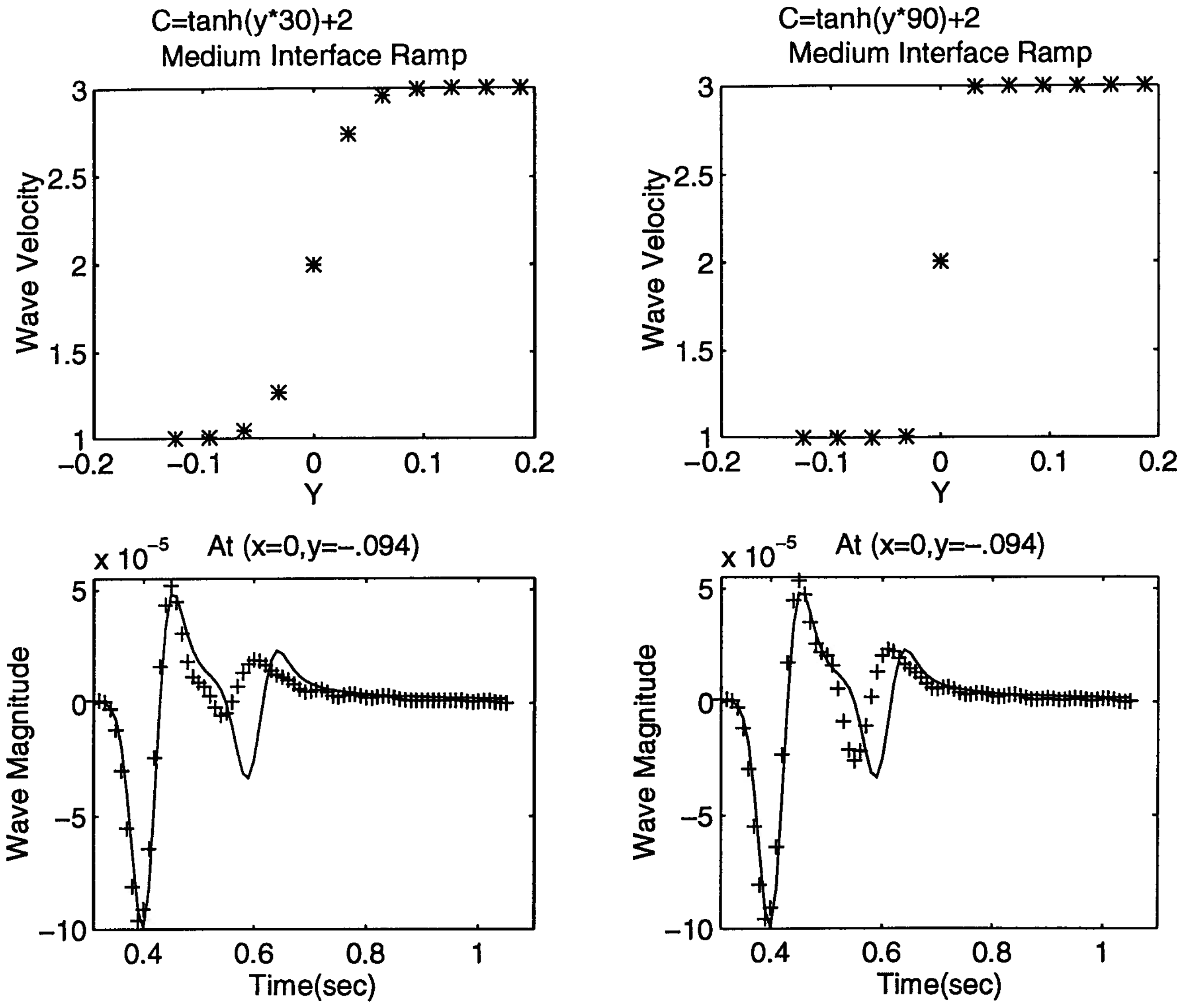


Figure 6.9 Trace Comparison Between Analytical (—) and Numerical (++) Solutions at (0, -0.094)

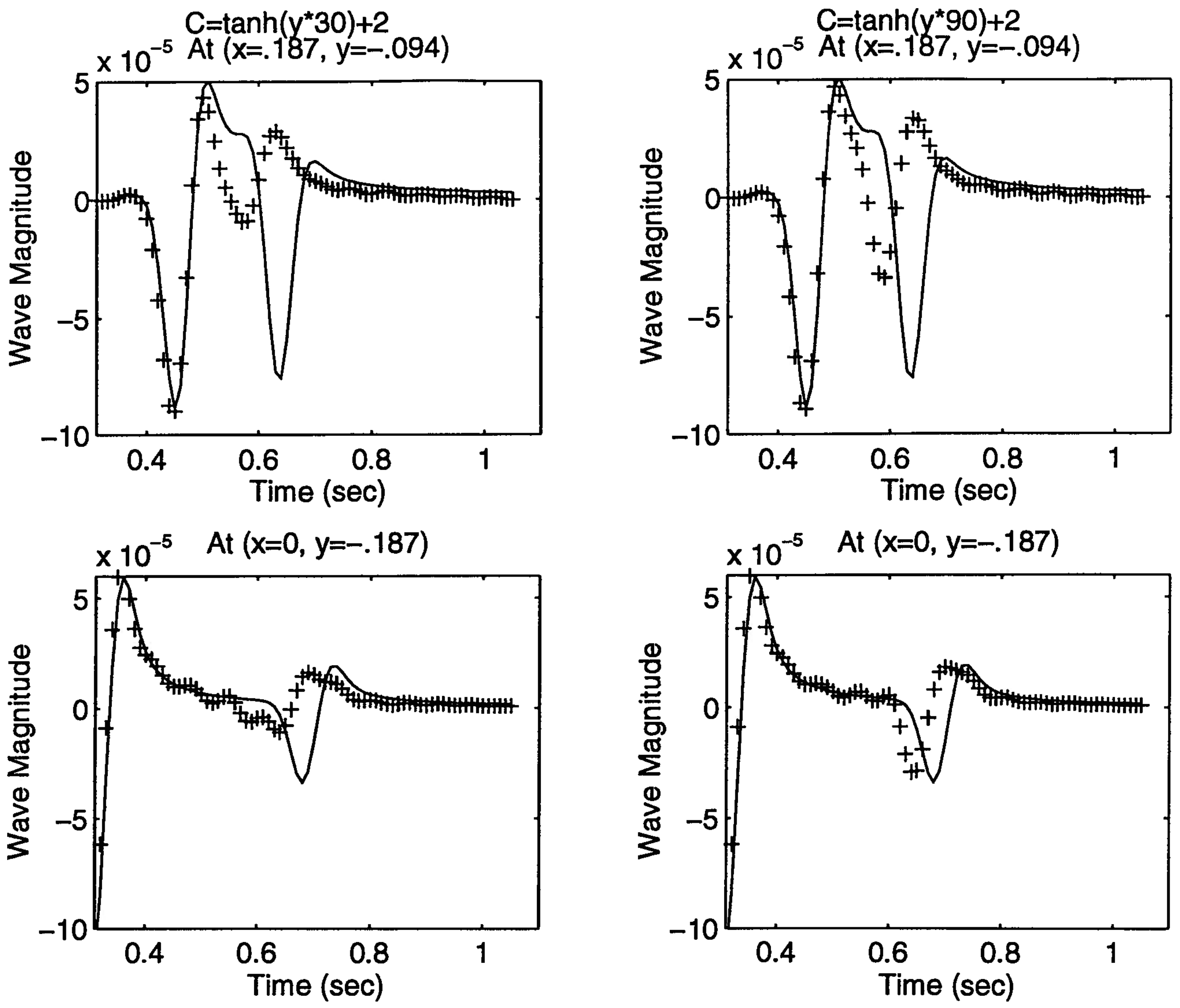


Figure 6.10 Trace Comparisons Between Analytical (—) and Numerical (++) Solutions at (0.187, -0.094) and (0, -0.187)

In the comparison graphs, the left column graphs are for numerical solutions with a smoother interface ramp (5 transition points from velocity 1 to velocity 3). The right column graphs are for numerical solutions with only one transition point for medium interface. From the trace comparison, it is clear that the direct wave is fairly accurately represented in the numerical solution. The main difference between the analytical solution and the numerical one happens in the reflection part due to the smoothed numerical interface ramp. Because of the existence of those intermediate transition points, the numerically calculated reflection always occurs earlier than the analytical one. Also, the numerical reflection wave is always weaker since the adjacent medium contrast at the smoothed interface is lower. When the number of ramp transition points is reduced to one (minimum transition for numerical simulation), the reflected wave is closer to the analytical one in both time and magnitude. On the other hand, the reflected field was obtained by an asymptotic expansion which is more accurate in the far field. This can be noted that the magnitude of the reflection wave in the trace at (0, -0.187) is a little bit closer to the analytical one than at (0, -0.094). In the traces at point (0, –0.187), one can also note that there are some ripples occurring in the numerical solution, right at the tail of the direct wave. This is mainly due to the band limitation of the Green's function.

Although time-marching is spectrally accurate due to the high accuracy of the polynomial time-marching scheme, all successive steps are based on the previously generated numerical results (with accumulated errors from the very beginning). Generally, this error accumulation effect exists for any time-marching scheme. Compared to the conventional finite-difference time-marching techniques, since a larger time-step size is allowed in this polynomial time-marching scheme, for a certain time-period, fewer time steps are required to accomplish the time-marching. Therefore, less errors are accumulated in this time-marching scheme. With the polynomial time-marching technique, one can even accomplish the time-marching for a certain time-period in one big time-step, provided machine precision is sufficient. Then the error accumulation can be avoided. However, in the illustrated trace results, after 750 time steps of time-marching, and after the interface ramp distortion, the tail of the reflected wave in the numerical solutions still fairly close to the analytical one. This phenomenon shows the high accuracy and stability of the polynomial time-marching scheme.

In conclusion, the trace comparison and the foregoing analysis show that the major error sources are the numerical modelling of the medium interface, the approximate absorbing boundary conditions and the numerical implementation of the Green's function (truncation and the asymptotic solution).

Chapter 7 Parallel Programming On the Connection Machine CM-5 and the Fujitsu VPX240/10

As mentioned in the Introduction, the pseudospectral methods have many potential parallelisms and have been implemented on the parallel computers [36, 37, 34]. On the other hand, the iterative multi-dimensional collocation evaluation in space is an extremely time-consuming procedure and it has to rely on the computational capabilities of supercomputers. However, the parallel implementation of the new time-marching technique is still a challenge even though the kernel of the new time-marching methods is based on the pseudospectral methods. Fortunately, two state-of-the-art supercomputers are available for this algorithm development and simulation: a 32–node (with vector unit) Connection Machine CM-5 at Stanford University (courtesy of Stanford Exploration Project), and a 2.5 GFLOPS peak performance single-CPU vector machine Fujitsu VPX240/10 at the Calgary High-Performance Computing Center (through HPC/Fujitsu scholarship). The general descriptions of the architecture of Fujitsu VPX240/10 vector machine and some important programming issues by the vectorized Fortran are given here. Also, Connection Machine CM-5 and some CM-Fortran programming features are introduced as a comparison. The implementation considerations of the polynomial time-marching scheme are addressed.

7.1 Fortran Programming on Fujitsu VPX240/10

The Fujitsu VPX240/10 at Calgary HPCC is a single processor, pipelined SIMD parallel computer with peak performance rate currently at 2.5 GFLOPS. The general architectural block diagram of the VPX240/10 is illustrated in Figure 7.1. The hardware summary of the VPX240/10 is also listed in Table 7.1. .

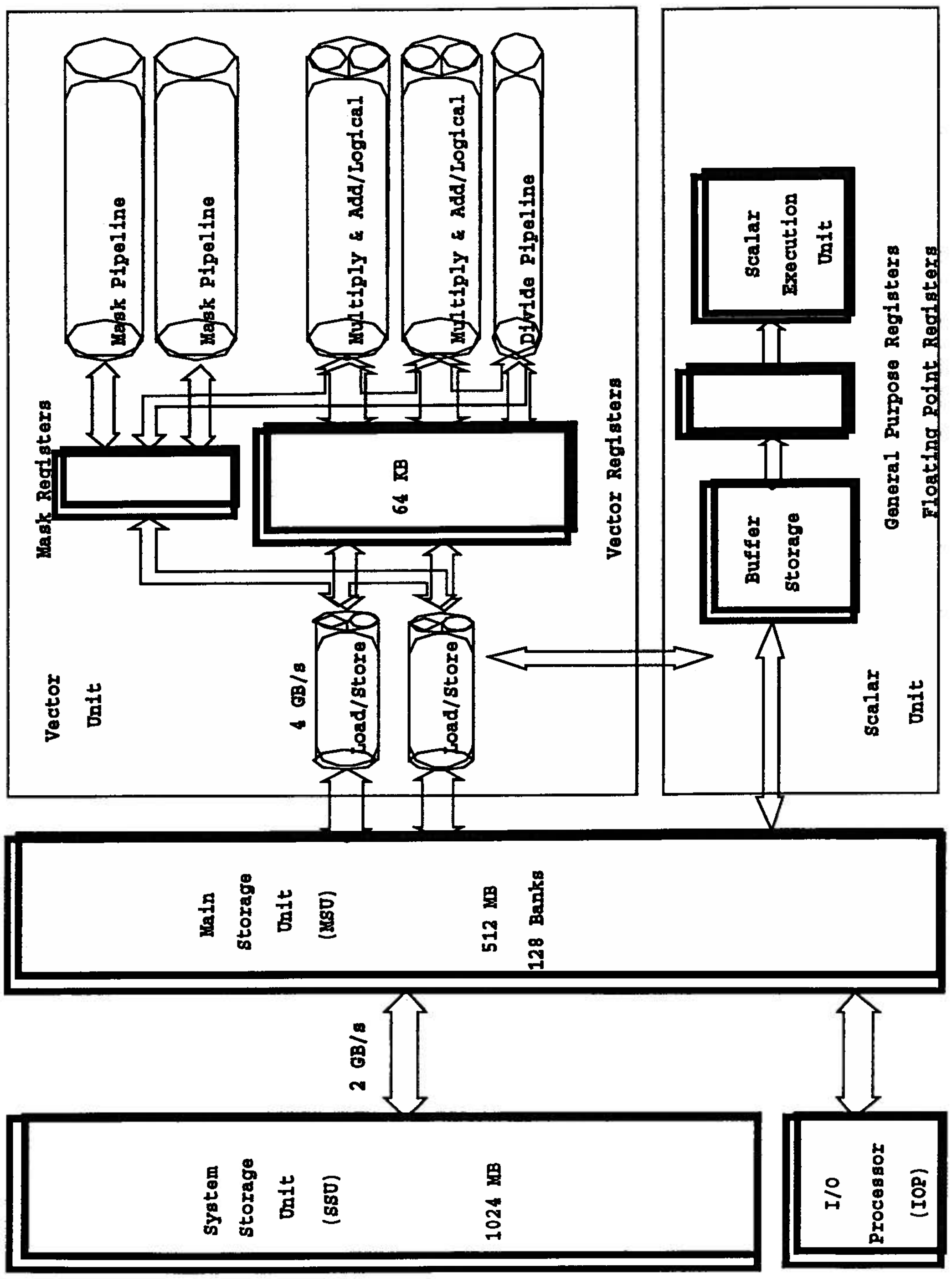


Figure 7.1 The Architectural Diagram of the Fujitsu VPX240/10 [23]

Table 7.1 VPX240/10 Hardware Summary[23]

No. of Vector Processing Unit	1
Vector Clock Period (Frequency)	3.2 nsec (312 MHz)
Peak Computation Rate	8 flops/cycle (2.5 GFLOPS)
Vector Register Size	64KB reconfigurable
Main Storage Unit (MSU)	512 MB (64 Mwords)
Memory Architecture	128-way interleaved
Load/Store Pipes	2 bidirectional
Memory Bandwidth	4 GB/sec
System Storage Unit (SSU)	1024 MB (128 Mwords)
MMU to SSU Bandwidth	1 GB/s read+1 GB/s write
No. of Scalar Processing Unit	1
Secondary Storage (Disk)	60 GB

The VPX240 vector processing unit is a SIMD (Single Instruction Multiple Data) pipelined parallel processor. The term "vector" processing originated with the 1977 vintage CRAY-1. The power of the original CRAY-1 was derived from the extensive use of pipelining, supported by segmented arithmetic units and an interleaved memory. However, the maximum speedup available from pipelining is limited by the number of operand pairs that can be in the pipe simultaneously (about 10); all further performance must come from parallelism. The VPX240 has added significantly to the peak performance of a single vector processing unit by introducing in the VPX240/10 model four independent arithmetic pipes for a peak computational rate of 8 floating point operations (FLOPS) per 3.2 nsec clock period, which translates to 2.5 GFLOPS.

The optimal computation on the VPX240 is the matrix multiply computation defined by the following kernel [23, 16]:

```
do 4 j=1, n
*VOCL LOOP,UNROLL(4)
          do 4 k=1, n
```

```
        do 4 i=1, n
    4   c(i,j)=c(i,j)+a(i,k)*b(k,j)
```

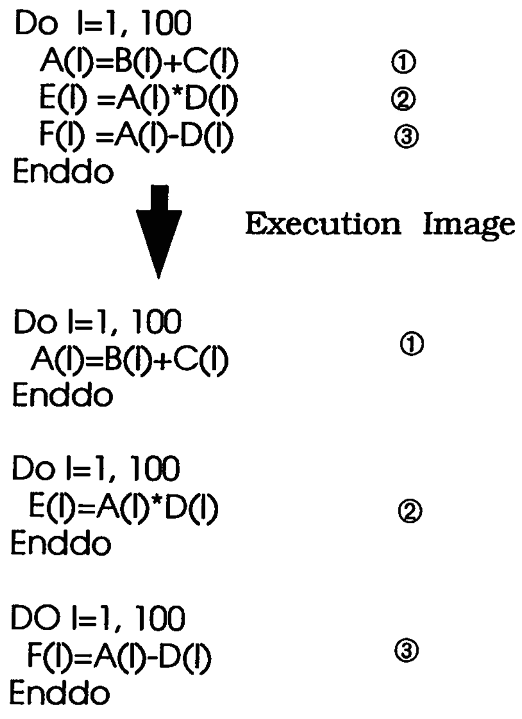
Here, the use of compiler directive (*VOCL LOOP,UNROLL(4)) to unroll the k loop to a depth of 4 increases the number of parallel floating point operations in the inner loop to 8 to match the 4–pipe architecture of the VPX240 without introducing any changes to the original source code. A 1024 by 1024 matrix multiply in standard FORTRAN code cán execute at a speed very close to the 2.5 GFLOPS peak performance. according to HPCC in [23].

The VPX240/10 used for this simulation has a 512 MB Main Memory Unit (MMU) divided into 128 memory banks. Although two 4 GB/sec bidirectional load/store pipes are provided for the vector unit memory access, the memory contention problem still exists. This problem can severely degrade the system performance if continuous memory addressing is not properly used. Generally, for a continuous real*8 array access, unit stride addressing through memory causes no memory bank conflict problems, but a 2n+1, 4n+2 and 4n stride addressing may respectively cause a performance degradation of factor 2, 4 and 8. Therefore, the easiest way to avoid memory bank contention with real*8 strides is to declare the leading dimension of arrays as an odd value (odd stride) , and try to put the leading dimension index in the innermost loop as long as it's possible (unit stride).

The VPX240/10 also has 1 GB of secondary memory, a semiconductor memory based System Storage Unit (SSU) , which can be configured as part of the file system. The transfer rate between SSU and MMU is 1GB/sec for either reads or writes, for a total bandwidth of 2 GB/sec. Users can perform I/O to the SSU using standard Fortran reads or writes at a bandwidth approaching 1 GB/sec [23]. This feature provides a very good way for temporarily swapping out some intermediate computational results.

The VPX240 Fortran compiler is a fully conforming ANSI standard Fortran 77 compiler. It is an optimized compiler with powerful auto-vectorizing capabilities. The compiler is able to vectorize a wide range of loops using a variety of advanced techniques, including inner and outer loop unrolling, nested DO loop reordering, subroutine inlining, vectorization of conditional blocks, first-order recursion and so on. The vectorization of a DO loop can be simply illustrated in Figure 7.2 [17].

When a DO loop is vectorized, the execution order of the statements within the scope of the DO loop changes. Each statement appears as if it is executed within its own DO loop. Vectorized statements are executed so that the definition and reference order of data appearing within the statement do not change. If nothing restricts the definition and reference order of the data in the statement, the execution order of the statement becomes undefined. The statement is not necessarily executed according to the order in which is appears in the source program. Optimization is performed so that the program executes at high speed, utilizing the parallel processing capabilities of the vector processor hardware. Here the loop (1) has to be executed before the loop (2) and the loop (3) due



```
Do I=1, 100
  A(I)=B(I)+C(I)        ①
  E(I) =A(I)*D(I)       ②
  F(I) =A(I)-D(I)       ③
Enddo
```

```
Do I=1, 100
 A(I)=B(I)+C(I)         ①
Enddo

Do I=1, 100
 E(I)=A(I)*D(I)         ②
Enddo

DO I=1, 100
 F(I)=A(I)-D(I)         ③
Enddo
```

Figure 7.2 Vectorizing a DO Loop

to the reference order in the original source. But the (2) and (3) loop execution order is undefined. Each of the loops will be organized into the following vector form (take the loop (1) as an example):

$$\begin{bmatrix} \mathtt{A(1)} \\ \mathtt{A(2)} \\ \vdots \\ \mathtt{A(100)} \end{bmatrix} = \begin{bmatrix} \mathtt{B(1)} \\ \mathtt{B(2)} \\ \vdots \\ \mathtt{B(100)} \end{bmatrix} + \begin{bmatrix} \mathtt{C(1)} \\ \mathtt{C(2)} \\ \vdots \\ \mathtt{C(100)} \end{bmatrix}$$

Thus, the DO loop operation becomes a vector operation that processes vector data [A], [B] and [C] in one operation.

Consequently, the following factors have been carefully considered during the Fortran implementation of the simulation on the VPX240/10 :

- Vectorizing the code as much as possible so that the real computational power of the vector unit can be properly utilized;
- Carefully dealing with some useful compiler directives such as UNROLL; whenever it's possible, an appropriate unrolling factor (or NOUNROLL because the default unrolling depth is 2) is used so that the number of the inner loop floating point operations is as close as possible to 8, and/or the number of the operation statements is 2. Then the 4 arithmetic pipes and/or the 2 load/store pipes can be fully utilized;
- Declaring the leading dimension of every array as an odd value and indexing the lead dimension in the innermost loop where possible, to minimize the memory contention;
- To avoid some unnecessary overhead, write repetitive matrix multiplication/add operations in a optimal and customized way, instead of using those general purpose matrix routines from NAG or SSL II/VP (Scientific Subroutine Libraries);
- For very large-scale simulations (for example, multi-thousand time steps), the SSU is used to temporarily store partial results so as to reduce the main memory consumption.

The following tables show some performance data of the VPX240/10 and demonstrate some powerful effects of the compiler directives and the memory contention problem.

From the above tables, it is quite clear that the optimal performance of the matrix multiply

Table 7.2 2K*2K REAL*8 Matrix Multiply

	NAG F01CRF routine	Compiler directive: LOOP,UNROLL(4)	Compiler directive: LOOP,UNROLL(2)
CPU Time (sec)	10.287	9.764	14.950
VU Time (sec)	10.206	9.740	14.928
Vectorization	99.2%	99.8%	99.9%

Table 7.3 Memory Access (2K*2K REAL*8 Array): two-dimensional assignment (3 arrays)

	Column loop inside, row loop outside	Column loop outside, row loop inside
CPU Time (microsec)	10772	10701
VU Time (microsec)	10607	10598
Vectorization	98.5 %	99.0 %

kernel, which fully utilizes the arithmetic pipes, is better than the corresponding NAG routine which may have some overhead. The 2nd table also shows that the memory access is always faster for the unit stride contiguous access.

Due to the non-IEEE internal binary format and different intrinsic function implementation, the Schwarz-Christoffel conformal mapping package, used for generating the interpolation points and coefficients in (3.2.14), can not guarantee a robust output on the VPX240/10. Since this package is obtained from NETLIB and includes many complicated numerical routines, which are hard to modify and optimize for the VPX240/10, all the necessary data through a good implementation of this SC package are precomputed on the SUN SPARCstation. Then one can let the VPX240/10 concentrate on the kernel operator evaluation operation. A preliminary Fortran implementation based on the above listed principles can achieve about 96.8% vectorization for the whole program execution. For the kernel part of the time-marching, over 98% vectorization can be achieved and each time step (14 iterations per step) takes about 0.78 sec CPU time, including 0.765 sec vector unit time (domain size 101*201). This execution time can be converted at the sustained processing rate of about 650 MFLOPS, 25% of the theoretic peak performance 2.5 GFLOPS or 37% of the achievable peak performance 1.76 GFLOPS (based on 2K*2K Real*8 matrix multiply execution time 9.76 sec).

7.2 CM-5 and CM-Fortran

The previous implementation of the pseudospectral time-marching technique was based on the Thinking Machines Corp. CM-2 SIMD supercomputer. Specially, the Fourier pseudospectral time-marching with periodic boundary conditions was mainly implemented on the hypercube networked CM-2. Since December 1992, the parallel computing platform has been upgraded to the latest TMC SIMD & MIMD supercomputer, the CM5. Since then, all later implementations, such as Chebyshev collocation, absorbing boundary approximation and the final simulation, were all based on the more powerful CM5.

The CM5 is the latest supercomputer developed by Thinking Machines Corp. (Cambridge, MA) in 1990s. It continues and extends support of the parallel programming model that has been proved successful in the CM-2 and CM-200. The major components of a typical CM-5 system are illustrated in Figure 7.3.

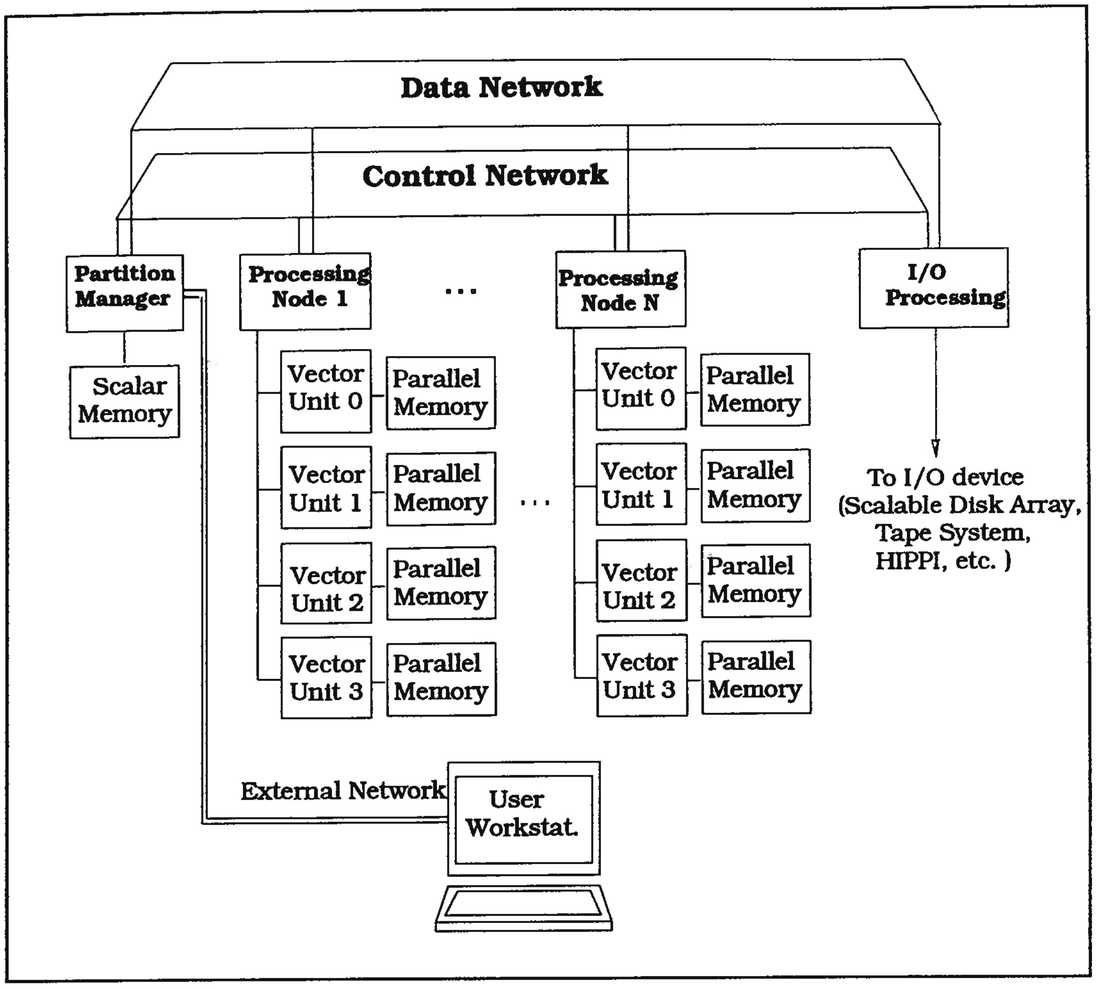


Figure 7.3 Components of a Typical CM-5 System [49]

The two network and the processing nodes are the key components of the CM-5. The control network is used for operations that require all the nodes to work together, such as the partitioning of the processing nodes, general broadcast of messages, and global synchronization. The data network is used for point-to-point exchange of messages between individual processing nodes. The topology of the CM-5 data network is a fat tree (quadric tree) and currently it transfers data at upwards of 5 Mbytes/second per node. Each CM-5 processing node contains a RISC microprocessor (currently a SPARC chip) and the optional four vector units, as shown in Figure 7.4. From Figure 7.4, it

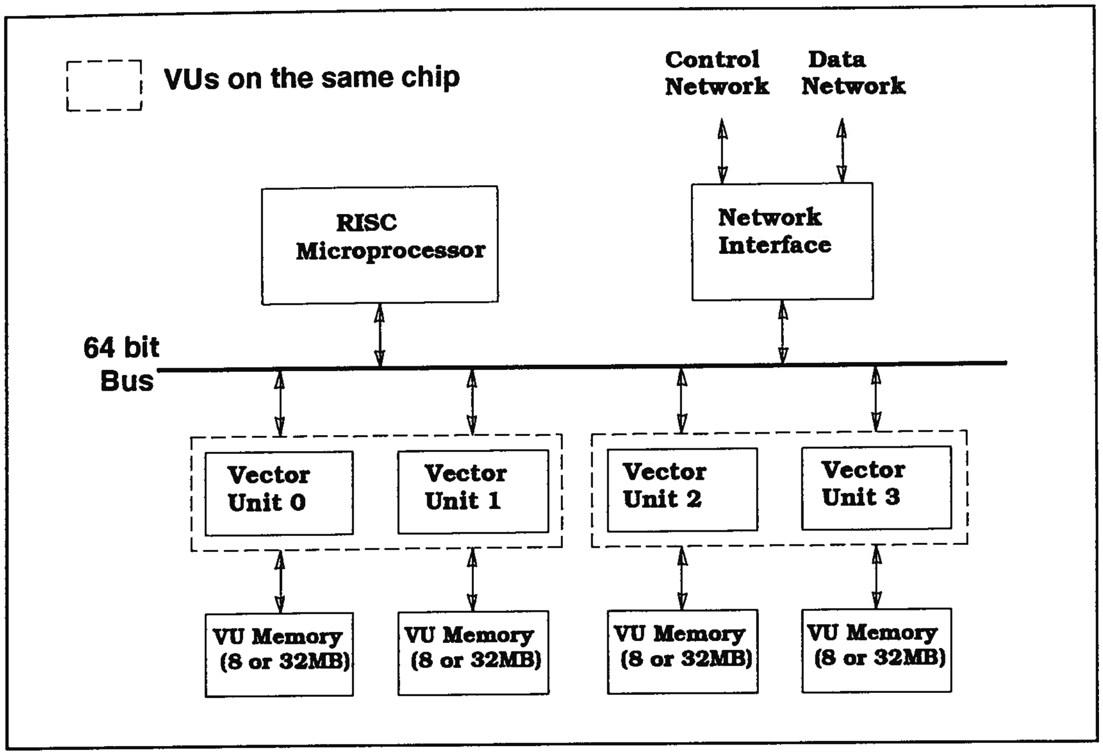


Figure 7.4 Components of a Processing Node with Vector Units

is clear that the VUs are implicitly grouped in pairs. This affects the speed of data transfer between the memory regions of the VUs. The fastest exchanges take place between the two VUs on the same chip; the next fastest between VUs on the different chips on the same node. Transfers between different processing nodes require network communication, which takes longer.

Here CM-Fortran is used as the programming language because it's an implementation of Fortran 77 supplemented with array processing extensions from the ANSI Fortran 90, and it is also suitable for scientific computing. In Fortran 90, an array object can be referenced by name in an expression or passed as an argument to an intrinsic function, and the operation is performed on every element of the array. This feature matches the parallel architecture of the CM system, which can process all the thousands of elements in unison. If the number of array elements is greater than the number of the physical processors, the CM system supports Virtual Processing which lets each physical processor behave as many Virtual Processors (VPs) so that the Connection Machine programs are completely scalable among different physically-sized CM systems. The CM system architecture and interconnection network provide three different VP set layouts in CM Fortran: NEWS, SEND and SERIAL, respectively corresponding to the NEWS grid interconnection, the data network router connection, and all elements along the axis local (within one processing node).

CM Fortran's execution model refers to the way a program makes use of the hardware. The CM-Fortran on CM-5 systems supports three kinds of execution models: [43]

- A *global* model, where a single program operates on arrays of data spread across all the parallel processors
- A *nodal* model, where multiple copies of a program operate independently on subsections of the data and communicate in message passing style
- A *global/local* model, a combination of data parallel and message passing for maximum flexibility in programming

The Fortran 90 features of CM-Fortran refer to the *global* model, which is data parallel (corresponding to SIMD).

In the polynomial time-marching applications, a large number of operations are matrix/array operations in a standard SIMD data parallel form. Therefore, the *global* model of CM-Fortran is chosen for the simulations.

The global CM Fortran programs executes in a master/slave style between a partition manager (PM) and the processing nodes in its partition. When the PM comes to a program that requires a parallel operation (that is, requires the processing nodes, PNs to execute some operation on their data), it broadcasts a call to a parallel routine, and the PNs immediately execute that routine. The PM handles all scalar instructions and CM Run-Time Systems (CMRTS), thereby controlling all communication involved as shown in Figure 7.5 It is clear that all use (explicit or implicit) of parallel communication

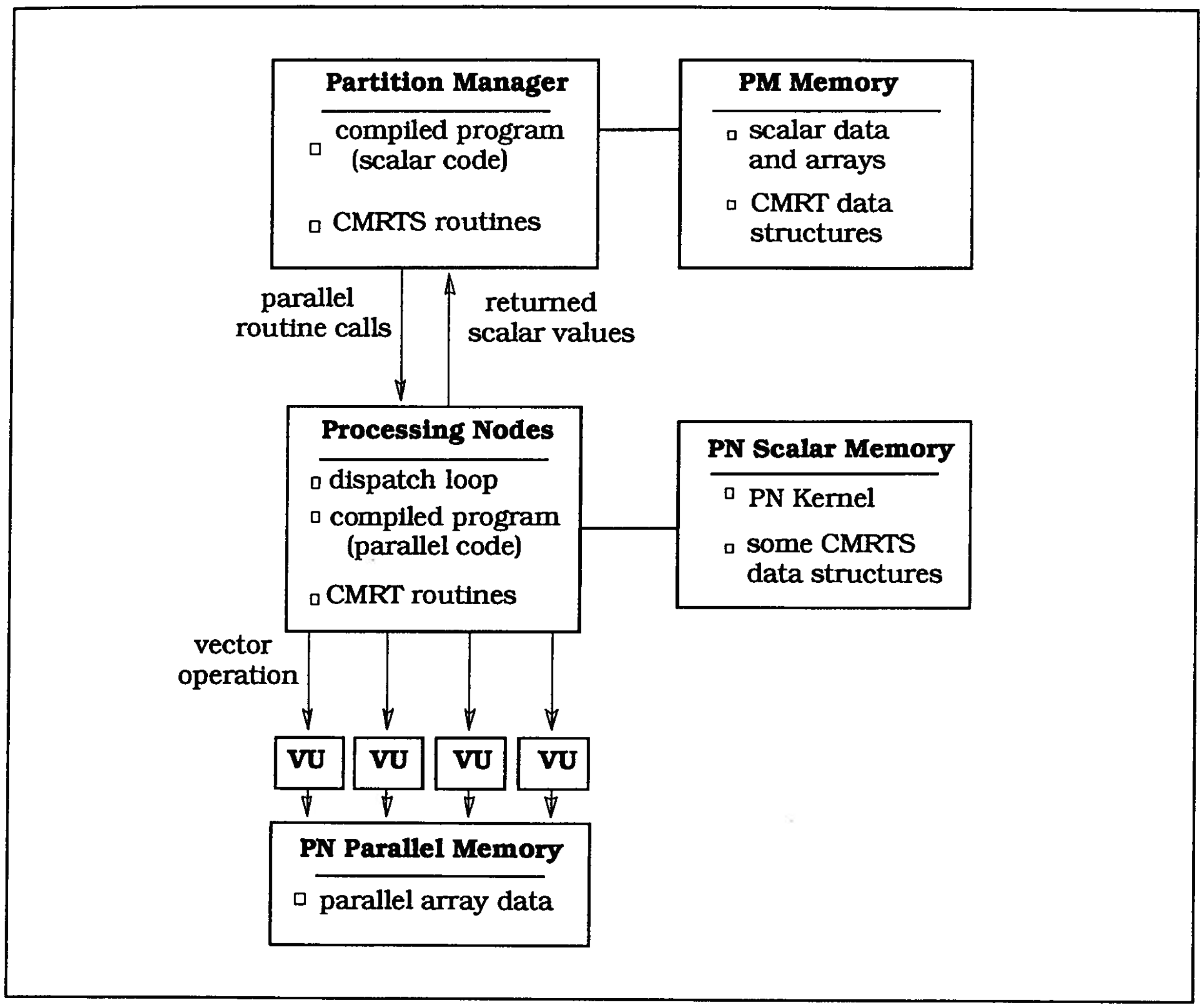


Figure 7.5 Distribution of Code and Data in a CM Program [49]

involves not only the data movement (which is the bottleneck in any parallel computing) but also the PM. Therefore, a significant amount of overhead occurs with the parallel communication. As a general rule for multiprocessor parallel programming, the parallel communication should be minimized so as to increase the purely parallel computational portion thereby increasing the general efficiency. CM Fortran provides some compiler directives, command line switches and some utility libraries to control the detailed parallel array layout. Through these tools, a programmer can flexibly control the distribution of the parallel array elements so that the inter-array operations involve a minimal amount of parallel communication. On the CM-5 systems, basically the parallel distributed **:NEWS** layout and the locally **:SERIAL** layout are in effect.

As discussed in the previous chapters, in contrast to the FFT kernel operation for the Fourier pseudospectral time-marching, the kernel operation of the Chebyshev collocation polynomial time-marching is the matrix multiply. Therefore, among the comprehensive scientific utilities provided by the Connection Machine Scientific Software Library (CMSSL), the matrix multiply utility is addressed because it is extensively used as the kernel operation in the final simulations described in Section 6.1.

There are two routines available on the CM-5 for matrix multiply: CM Fortran intrinsic function MATMUL and the CMSSL routine gen_matrix_mult_noadd. When the CM Fortran program is linked with CMSSL Version 3.1 library, the MATMUL performance is specially optimized and improved. The performance data in the following table show how the parallel array layout affects the matrix multiply performance. The CM

Table 7.4 2K*2K Double Precision Matrix Multiply C=A*B by CMSSL Routine gen_matrix_mult_noadd

	CM Elapsed Time (sec)	CM Busy Time (sec)
All arrays laid out as (:news, :news)	9.636	9.622
A, B: (:news, :serial), C: (:news, :news)	9.759	9.578
All arrays laid out as (:news, :serial)	16.555	16.338

elapsed time here means the elapsed execution time while the program is not swapped out by the OS on the partition manager. The CM busy time means the time spent actually executing parallel computation on the processing nodes. Due to the default axis reorder action taken by the CM Fortran compiler, the **:NEWS** axis always varies faster than the **:SERIAL** one. Therefore, a layout of **(:NEWS, :SERIAL)** is virtually equivalent to

(**:SERIAL, :NEWS**) layout here for these 2K*2K arrays. From the performance data listed in the table, making one axis of **A** and **B** local (**:SERIAL**) can cause a marginal performance improvement and making one axis of all arrays local even degraded the performance. On the other hand, since a series of matrix/array operations will be involved, the incompatible layout in the 2nd case in Table 7.4 may cause further performance degradation due to the different layouts between arrays. Therefore, unanimous **:NEWS** layout is used through the entire time-marching implementation.

7.3 Parallel Implementation of the Polynomial Time-Marching by CM-Fortran

In this section, the preliminary optimized implementation issues on the CM-5 of the polynomial time-marching simulation will be briefly discussed. For the model discussed in Section 6.1, the following implementation issues should be carefully handled:

1. Based on the layout discussion in the previous section, **:NEWS** layout is used throughout the entire program;
2. In order to minimize the interprocessor communication, some row, column masks for incorporating boundary conditions (e.g. matrices $[A_1]$, $[A_2]$, $[B_1]$ in (6.1.5)) should be handled via CM Fortran **WHERE** mask block, rather than **FORALL** which may cause some unnecessary communication;
3. Due to some complicated non-linear numerical approximation routines involved, currently it's better to run the Schwarz-Christoffel conformal mapping package in Fortran 77 serially on the front-end. However, this may cause a significant amount of PN idle waiting while the SC package is running on the front-end. Thus, a more efficient alternative is to pre-compute the necessary SC parameters on the workstation (e.g. in MATLAB) and save these data for the final CM parallel processing;
4. To guarantee the minimum amount of communication involvement, the **-list** compiler option can be used to generate an additional listing file. It contains a complete line-numbered source-code listing of the program, as well as detailed information about

the array homes, a list of all CM interprocessor communication operators used by the program, with line number showing where they are used. This information is particularly useful in helping to identify all the communication, to eliminate redundances.

By applying the above points to the CM Fortran program, the 2–D seismic simulation program kernel with virtually no communication can be obtained. This is confirmed by the generated listing files. The following table briefly illustrates the performance upgrading. Since the communication caused by the **FORALL** operation is eliminated in the kernel

Table 7.5 Per Time Step Kernel Operation Execution Time (model discussed in Section 6.1)

	CM Elapsed Time (sec) per time step	CM Busy Time (sec) per time step
FORALL used for the boundary manipulations	9.262	8.656
WHERE mask used for the boundary manipulations	8.954	8.341

operation (6.1.5), the execution time is reduced. Furthermore, the difference between CM elapsed time and busy time is also reduced since the PM involvement for the parallel communication is reduced.

The polynomial time-marching algorithms evaluate the time dependent solution of the wave equation by extensively and recursively using the pseudospectral methods. The computational requirement therefore is much higher than the conventional time-marching techniques. Thus, the powerful parallel computers such as the Connection Machine must decidedly be the suitable tools. The parallel implementation of the algorithms should also be fairly important to the application of these algorithms. From the above discussion, it is also clear that a good or an optimal implementation on the massively parallel computer critically depends on the machine architecture and the parallel language features.

Chapter 8 Conclusion

8.1 Summary

The purpose of this thesis research is to develop a novel numerical algorithm, based on Tal-Ezer's time-marching method, to simulate wave scattering phenomena in the time-domain. The simulation usually is implemented on the supercomputers, which require very high vectorization or parallelism in the algorithm. The pseudospectral methods are chosen for the spatial derivative approximation and the polynomial expansion or interpolation methods are developed for the time-marching.

In Chapter 2, the theoretical principles of pseudospectral methods are introduced. The pseudospectral (or named as collocation) methods always associate a set of grid points with some specific basis function sets. The choice of the basis function sets is determined by the boundary conditions of the domain. Usually, the Fourier series is chosen for periodic domains and Chebyshev polynomials are chosen for non-periodic domains. Compared to the finite-difference methods, the pseudospectral methods have "infinite-order" or "exponential" convergence. This "exponential" convergence, however, degrades in the presence of internal boundary and also some boundary conditions imposed on the external boundaries. The discussion is focused on the derivative representations of the Fourier and the Chebyshev pseudospectral methods. The latter one is extensively addressed in consecutive chapters.

First in Chapter 3, the method of expanding the solution time dependence by Chebyshev polynomials is presented. Its accuracy, resolution, convergence and stability issue are also discussed. The time-marching scheme is later interpreted as a special case of the general Faber polynomial approximation. It is limited to the case of a skew-symmetric spatial operator with purely real or imaginary eigenvalues. Hence, it is only applicable to the Fourier pseudospectral spatial approximation which implies the periodic boundary situations. Then, a more general Faber polynomial time-marching scheme and its practical implementation, the Newton-form interpolation, are introduced, to deal with those more complicated spatial operators occurring in the wave equation.

By manipulating the Chebyshev collocated derivative operators in space, non-periodic boundary conditions are incorporated into the polynomial time-marching scheme, as presented in Chapter 4. Homogeneous Dirichlet, Neuman and the mixed boundary conditions respectively are included in the polynomial time-marching scheme without changing the polynomial marching procedure. The key is to incorporate the homogeneous boundary conditions into the spatial operator so that the eigenvalue distribution of the combined operator reflects the boundary constraints. Consequently, the Newton-form interpolation, which is determined by the eigenvalue distribution domain and the function form of the symbolic solution, is also affected by the boundary constraints. After incorporating the boundary conditions, the new scheme still maintains the same high resolution and accuracy as discussed in Chapter 3, for the model problems presented. These numerical examples show that the stability limitation over the time step in this new scheme is $O(\frac{1}{N^2})$.

Furthermore, in order to simulate a long period of wave-scattering in a restricted domain without involving severe edge reflection, approximate non-reflecting boundary conditions are introduced in Chapter 5. The approximate absorbing boundary conditions are also incorporated into the polynomial time-marching in a similar way. It is shown that the 1st and 2nd order approximate absorber can be conveniently included in the polynomial time-marching without any significant algorithm changes or more computational cost. One complete numerical simulation example, two-dimensional seismic reflection model, is presented in Chapter 6. The snapshots of the visualization demonstrate scattering phenomena of diffraction, reflection and transmission.

Chapter 7 addresses the parallel programming issues of the numerical simulation on the supercomputers Fujitsu VPX240/10 and TMC CM-5. The architectures of the two supercomputers are introduced and compared. The detailed programming and optimization considerations in Fortran (vectorized Fortran on the VPX240/10 and CM-Fortran on the CM-5) are also discussed. Some preliminary optimization results are presented there. The performance data show that the polynomial time-marching scheme can be highly vectorized or parallelized. Therefore, it's very suitable for implementation on supercomputers for simulating some large-scale complex physical phenomena.

8.2 Contributions

The following is a list of the major contributions of this thesis:

- Implementation of the Fourier pseudospectral time-marching technique on the CM-2 supercomputers to simulate some simple-shape scalar wave scatterings.
- Extension of the polynomial time-marching scheme to the non-periodic boundary value problems, including incorporating homogeneous boundary conditions, Dirichlet, Neumann or the mixed, into the Chebyshev collocation operators, thus significantly extending the application range of the polynomial time-marching technique.
- Derivation of the eigenvalue distribution of the combined two-dimensional Chebyshev collocation operator via *Kronecker product.*
- Incorporation of the 2nd order approximate absorbing boundary conditions into the polynomial time-marching scheme, thus making it possible to apply the polynomial time-marching scheme to simulate wave propagation in opened domains.
- Simulation and visualization of a complex seismic wave scattering model by the polynomial time-marching scheme on both of the CM-5 and the Fujitsu VPX240/10 supercomputers, including optimization of the parallel implementation in CM-Fortran (Fortran 90) on the CM-5 and in the vectorized Fortran on the VPX240/10.
- Examination, via numerical examples, for the one-dimensional 2nd order wave equation of the accuracy for different N. The exponential convergence degrades in the presence of internal boundaries or external applied boundary conditions.

8.3 Future Work

Due to the severe accuracy degradation in the presence of medium discontinuities or 2nd order absorbing boundary conditions, further work is needed to study the internal boundary manipulation and also a higher order absorbing approximation should be incorporated. A full comparison with an analytical solution in the two-dimensional variable medium should be studied in the future as well.

In this thesis, the polynomial time-marching scheme for two-dimensional Chebyshev collocation case have been studied. For three-dimensional case or the Chebyshev/Fourier mixed collocation case (for example, in cylinder or spheric coordinates), the whole derivation procedure but the configuration of the collocation operators remains the same. Therefore, in order to efficiently obtain the collocation matrix operator and its eigenvalue distribution in the three-dimensional case, an appropriate form of the three-dimensional Chebyshev collocation operator incorporated with boundary conditions needs to be built up.

Generally, this polynomial time-marching scheme deals with boundary-initial-value problems without any source excitation. However, some short pulse source problems can always be approximated by the same scheme as used in Chapter 6. Further theoretical work is needed to work out the cases with a long-lasting source.

Also, three-dimensional seismic forward modelling based on the complete elastic wave equation will be certainly considered as one of the goals for this algorithm. The major difficulty would be on constructing the three-dimensional operator and deriving its spectral radius.

Appendix A Some Snapshot of Fourier Pseudospectral Time-marching Results

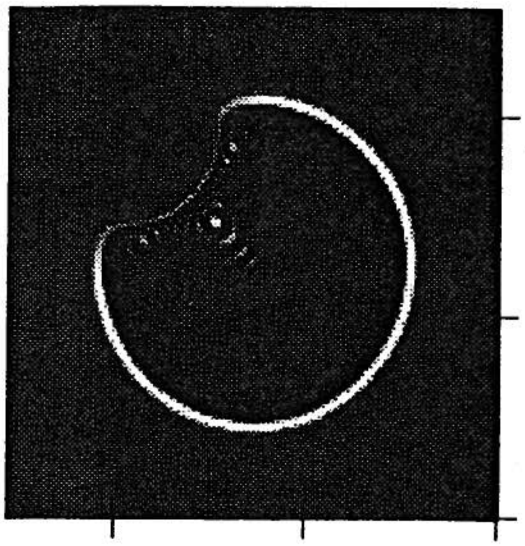


Figure A.1 Snapshot of wave scattering described in page 37—39

Appendix B Previous Study Results on the Performance of the Connection Machine CM-2

The Connection Machine CM-2/200 is a massively parallel, single instruction multiple data (SIMD) machine. A fully configured CM-2 has 64K one-bit processors. All these processing elements are organized into processing nodes each of which includes 32 processors, and the distributed memory, a 64–bit floating-point accelerator and communication interfaces for interprocessor communication. The interprocessor connection network of the CM-2 provides three forms of interprocessor communication: general communication, which is based on hypercube message passing router and physically connects the addresses differing by one bit; NEWS grid communication, which provides the hardware connections for each processor to its four nearest neighbors in a two-dimensional Cartesian grid; and global communication, which is useful for the global cumulative and reduction computations along one axis. These architectures provide more flexible parallel programming techniques in high level languages.

Some early stage work is based on Fourier pseudospectral and implemented on the Connection Machine CM-2 with 8K processors. The following is the outline of this work:

Performance Results for The Parallel Implementation of Fourier Pseudospectral Time-Marching on CM-2

The following table shows some performance data for computing a 128*128 two-dimensional problem in one time step (the Chebyshev polynomials were evaluated up to 28th order): From the performance data in the above table, it is clear that the

		The FFT complex working array layout (10000:send, :send)	The FFT complex working array layout (:serial,:send)[1]	Layout is the same as the first one, but attached to 8K processors
Total one step time-marching	CM time (sec)	0.8115	1.6923	0.4591
	CM Elapsed time (sec)	0.8326	1.7169	0.4877
Pseudo-spectral evaluation (computing all the required derivatives)	CM time (sec)	0.7702	1.6511	0.4376
	CM Elapsed time (sec)	0.7913	1.6757	0.4662
Chebyshev Polynomial Expansion	CM time (sec)	0.0398	0.0397	0.0206
	CM Elapsed time (sec)	0.0398	0.0397	0.0206

Table B.1 The 128*128 one time step performance comparison: attached to 4K processor except otherwise described. The CM-2 used here has 256 Kbits/proc. distributed memory and one 64–bits floating-point processing unit per 32 processors. The clock frequency of this CM-2 is 7 MHz.

pseudospectral evaluation uses the dominant part the execution time (about 95% of the

[1] the wave speed array, wavenumber array are also set in the same way.

total time), and the serial operation for the Chebyshev polynomial expansion is also quite efficient as the CM time is equal to the CM Elapsed time. Furthermore, setting the first space axis of a two-dimensional problem as SERIAL will greatly decrease the parallel execution efficiency. Increasing the number of the parallel processors to 8K can increase the performance by a factor about 1.7 (less than linear)

As previously mentioned, the kernel parallel operation of this algorithm is the evaluation of various orders of the spatial derivative operator by the pseudospectral methods, either Fourier (for periodic interval) or Chebyshev (for boundary value problems) collocation. Hence, some conventional pseudospectral parallel implementation techniques, as analyzed by R.B. Pelz in [36], will also be applied in this implementation:

1. To minimize the communication cost in the FFT, it is desirable that the x direction (the first space axis of the array) be mapped as locally as possible. The x axis weight is set to be greater than the other axes;
2. To match the FFT butterfly communication pattern, which always operates between data elements at a distance of 2^k, the FFT complex working array axes should be laid out in SEND hypercube ordering;
3. To avoid wasteful communication cost, set the frequency domain axis ordering in bit-reversed order;
4. To make the wavenumber multiplication more efficient, precompute all the one-dimensional wavenumber vectors and combine them as a three-dimensional bit-reversed ordered wavenumber array with the same layout as the working array. Thus, the interprocessor communication during the multiplication can be minimized.

In addition to the above, because the new algorithm requires the polynomials of the spatial derivative operator for time-marching, it is also necessary to extend the previous parallel implementation techniques for the conventional pseudospectral methods:

1. To make better use of the Chebyshev and Bessel recurrence relations, the Chebyshev polynomial coefficients and the Bessel function values are calculated on the front-end

beforehand. The computation of the Bessel function values is based on the NETLIB special function routines (written by W. J. Cody of Argonne National Laboratory);

2. To evaluate the Chebyshev polynomial of the spatial derivatives, the Chebyshev polynomial coefficients and the Bessel functions will be addressed individually for multiplication with the various order derivative arrays. Therefore, storing these coefficients and function values as frond-end arrays instead of parallel CM arrays can dramatically reduce the single CM array element movement, which is one of the most inefficient operations in the SIMD machines;
3. Because of the above consideration, the order index dimensions of the derivative arrays are also desirably set as SERIAL, so that some serial DO-LOOPs can be used for the polynomial evaluation. (the DO-LOOP is usually more efficient in the SERIAL dimension than FORALL[43]);
4. As the new time-marching technique is not as simple as the finite difference one, which may use the nearest neighbor NEWS grid communication, it is also expected to set the time dimension as SERIAL so as to use DO-LOOP for the more complicated time-marching operations;
5. If the first space axis of the FFT working array (complex) is set SERIAL, all the other arrays should also be laid out in the same way so as to minimize the communication cost of the operations between them and the working array. Because the time axis of the solution array has been set SERIAL (mentioned above), the two-dimensional problems will only have one parallel dimension in the solution array. This layout usually is also inefficient because the spreadout of the only parallel dimension may be less than the CM parallel processing requirement[43] (For an 8K CM-2, a 1K parallel element dimension is the minimum requirement for the *Slicewise* mode. Otherwise some processors have to be idled). Therefore, all the space axes have to be laid out as SEND ordering, but the first one should be given the highest weight.

Appendix C Derivation of (6.1.5)

Step 1: represent the 2nd order equation (6.1.1) into Chebyshev collocation matrix form (no boundary conditions).

$$[v] = \frac{1}{\rho}\{r_x^2[D_x](\mu)[D_x][v] + r_z^2[v][D_z](\mu)[D_z]\} \tag{C.1}$$

Here, r_x, r_z are defined in (6.1.7). The []s are matrices and ()s are arrays. $[D_x], [D_z]$ are just normal Chebyshev collocated 1st order derivative matrix, defined in (4.1.3).

Step 2: incorporate the top edge free surface (Neumann) boundary condition and the 2nd order terms of all the absorbing boundary conditions into (C.1).

$$[v] = \frac{1}{\rho}\{r_x^2[D_0](\mu)[D_1][v] + r_z^2[v][D_z](\mu)[D_2]\}, \tag{C.2}$$

where

$$D_0 = \begin{bmatrix} d^x_{0,1} & \cdots & d^x_{0,N} \\ \vdots & \cdots & \vdots \\ d^x_{N-1,1} & \cdots & d^x_{N-1,N} \\ 0 & \cdots & 0 \end{bmatrix}_{(N+1)\times N}, \quad D_1 = \begin{bmatrix} d^x_{1,0} & \cdots & d^x_{1,N} \\ \vdots & \cdots & \vdots \\ d^x_{N,0} & \cdots & d^x_{N,N} \end{bmatrix}_{N\times(N+1)} \tag{C.3}$$

$$D_2 = \begin{bmatrix} 0 & d^z_{0,1} & \cdots & d^z_{0,M-1} & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & d^z_{M,1} & \cdots & d^z_{M,M-1} & 0 \end{bmatrix}_{(M+1)^2}$$

The matrix $[D_0]$ includes the 2nd order term of the absorbing boundary condition for the bottom edge. The matrix $[D_1]$ reflects the top edge homogeneous Neumann condition. The matrix $[D_2]$ contains the 2nd order terms of the absorbing boundary conditions for the left and right edges.

Step 3: build the 1st order terms for the absorbing boundary conditions.

$$\text{op2}\{v_t\} = (c)\{r_x[D_4][v_t] + r_z[v_t][D_5]\}, \tag{C.4}$$

where

$$D_4 = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \cdots & \vdots \\ 0 & \cdots & 0 \\ d^x_{N,0} & \cdots & d^x_{N,N} \end{bmatrix}_{(N+1)^2}, \quad D_5 = \begin{bmatrix} -d^z_{0,0} & 0 & \cdots & 0 & +d^z_{0,M} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ -d^z_{M,0} & 0 & \cdots & 0 & +d^z_{M,M} \end{bmatrix}_{(M+1)^2} \tag{C.5}$$

The matrix $[D_4], [D_5]$ contain the 1st order term of the absorbing boundary conditions respectively for the bottom edge, the left and right edges.

Step 4: extract those corner points which have been counted twice by their adjacent edges, include the 0.5 factor for the 1st order term in the absorbing approximation.

$$\text{op1}\{v\} = \frac{1}{\rho}\{r_x^2[D_0](\mu)[D_1][v][A_2] + r_z^2[A_1][v][D_z](\mu)[D_2]\}\ , \tag{C.6}$$
$$\text{op2}\{v_t\} = (c)\{r_x[D_4][v_t][B_1] + r_z[v_t][D_5]\}$$

$$A_1 = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & 0 & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0.5 \end{bmatrix}_{(N+1)^2} \tag{C.7}$$

$$A_2 = \begin{bmatrix} 0.5 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & 0 & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0.5 \end{bmatrix}_{(M+1)^2}, \quad B_1 = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & I_{M-1} & \vdots \\ 0 & \cdots & 0 \end{bmatrix}_{(M+1)^2}$$

References

[1] Milton Abramowitz and Irene A. Stegun, editors. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, pages 877–878. U.S. National Bureau of Standards, 1972.

[2] Keiiti Aki and Paul G. Richards. *Quantitative Seismology: theory and methods.* W. H. Freeman And Company, 1980.

[3] R.M. Alford, K.R. Kelly, and D. M. Boore. Accuracy of finite-difference modeling of the acoustic wave equation. *Geophysics*, 39(6), 1974.

[4] Stephen Barnett. *Matrices, Methods and Applications*. CLARENDON PRESS, OXFORD, 1990.

[5] Joy P. Boyd. *Chebyshev & Fourier Spectral Methods*. Springer-Verlag, 1989.

[6] L.M. Brekhovskikh and Yu.P. Lysanov. *Fundamentals of Ocean Acoustics*. Springer-Verlag, 1982.

[7] L. Cagniard. *Reflection and Refraction of Progressive Seismic Waves*. McGraw-Hill, 1962.

[8] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods in Fluid Dynamics*. Springer-Verlag, 1988.

[9] C. Canuto and A. Quarteroni. Error estimation for spectral and pseudospectral approximations of hyperbolic equations. *SIAM J. Numer. Anal.*, 19:629–642, 1982.

[10] S.W. Ellacott. Computation of Faber series with application to numerical polynomial approximation in the complex plane. *Mathematics of Computation*, 40(162):575–587, 1983.

[11] B. Engquist and A. Majda. Absorbing boundary conditions for the numerical simulation of waves. *Math. Comput.*, 31:629–651, 1977.

[12] John T. Etgen. Accurate wave equation modeling. *Private Communication*, 1991.

[13] B. Fischer and L. Reichel. Newton interpolation in Fejér and Chebyshev points. *Math. Comp.*, 53:265–278, 1989.

[14] Bengt Fornberg. The pseudospectral method: comparisons with finite differences for the elastic wave equation. *Geophysics*, 52(4):483–501, 1987.

[15] Bengt Fornberg. The pseudospectral method: accurate representation of interfaces in elastic wave calculations. *Geophysics*, 53(5):625–637, 1988.

[16] FUJITSU LIMITED. *FUJITSU FORTRAN VP PROGRAMMING GUIDE*, 1991.

[17] FUJITSU LIMITED. *FUJITSU UXP/M FORTRAN77 EX/VP USER'S GUIDE*, 1992.

[18] G. E. Q. Goode. *Numerical Simulation of Viscoelastic Waves*. PhD thesis, University of Calgary, AB., 1993.

[19] David Gottlieb and Richard S. Hirsh. Parallel pseudospectral domain decomposition techniques. *Journal of Scientific Computing*, 4(4):309–325, 1989.

[20] David Gottlieb, M. Yousuff Hussaini, and Steven A. Orzarg. *Introduction: Theory and Applications of Spectral Methods*, pages 1–54. SIAM, Philadelphia, 1984.

[21] David Gottlieb and L. Lustman. The spectrum of the Chebyshev collocation operator for the heat equation. *SIAM J. Numer. Anal.*, 20:909–921, 1983.

[22] R.L. Higdon. Absorbing boundary conditions for difference approximations to the multi-dimensional wave equation. *Math. Comput.*, 47:437–459, 1986.

[23] HPC Center, Calgary, Canada. *HPCC User's Guide*, 1993.

[24] M.Y. Hussaini, D.A. Kopriva, and A.T. Patera. Spectral Collocation Methods. *Applied Numerical Mathematics*, 5:177–208, 1989.

[25] F. Kalantzis, N. Dai, E. R. Kanasewich, S. Phadke, and A. Vafidis. 2–d and 3–d seismic reflection modeling and imaging using vector and parallel supercomputer. In *Proceedings of Supercomputing Symposium'93*, June 1993.

[26] H. Kreiss and J. Oliger. Methods for the approximate solution of time dependent problems. Global Atmospheric Research Programme (GARP) Publications Series No.10, January 1973.

[27] E.L. Lindman. Free space boundary conditions for the time dependent wave equation. *J. Comput. Phys.*, 18:66–78, 1975.

[28] Yong Luo and Matthew J. Yedlin. Polynomial time-marching for non-reflection boundary problems. *Journal of Scientific Computing*. (prepared in Sept. 1994).

[29] Yong Luo and Matthew J. Yedlin. Simulating Some Complex Wave Scattering Problems on CM-2 Connection Machine by Pseudospectral Time-Marching. In *Proceedings of The 3rd International Conference on Applications of Supercomputers in Engineering*, Southampton, U.K., September 1993. Computational Mechanics Publications.

[30] Yong Luo and Matthew J. Yedlin. Solving wave equations by pseudospectral time-marching on cm-2 connection machine. In *Proceedings of The 7th Annual High-Performance Computing Conference of Canada*, Calgary, AB, June 1993.

[31] Yong Luo and Matthew J. Yedlin. Polynomial time-marching for non-periodic boundary value problems. *Journal of Scientific Computing*, 1994. (in press).

[32] A. I. Markushevich. *Theory of Functions of a Complex Variable*. Chelsea, New York, 1977.

[33] V.P. Maslov. *Operational Methods*. MIR Publishers, Moscow, 1976.

[34] Oliver A. McBryan. The Connection Machine: PDE solution on 65,536 processors. *Parallel Computing*, 9(1):1–24, December 1988.

[35] Alireza H Mohammadian, Vijaya Shankar, and William F. Hall. Computation of electromagnetic scattering and radiation using a time-domain finite-volume discretization procedure. *Computer Physics Communications*, 68:175–196, 1991.

[36] Richard B. Pelz. Pseudospectral methods on massively parallel computers. *Computer Methods in Applied Mechanics & Engineering*, 80:493–503, 1990.

[37] Richard B. Pelz. Fourier spectral method on ensemble architectures. *Computer Methods in Applied Mechanics & Engineering*, 89:529–542, 1991.

[38] R.A. Renaut. Absorbing boundary conditions, difference operator, and stability. *Journal of Computational Physics*, 102:236–251, 1992.

[39] Hillel Tal-Ezer. Spectral methods in time for hyperbolic equations. *SIAM J. Numer. Anal.*, 23(1), February 1986.

[40] Hillel Tal-Ezer. Polynomial approximation of functions of matrices and applications. *Journal of Scientific Computing*, 4(1):25–60, 1989.

[41] Hillel Tal-Ezer. High degree polynomial interpolation in Newton form. *SIAM J. Sci. Stat. Comput.*, 12(3):648–667, 1991.

[42] Thinking Machines Corp., Cambridge, AB. *Connection Machine CM-200 Series Technical Summary*, 1991.

[43] Thinking Machines Corp., Cambridge, MA. *Connection Machine Fortran Programming Guide (V.1.13), (V 2.1, Feb. 1994)*, July 1991.

[44] Thinking Machines Corp., Cambridge, MA. *Getting Started in CM Fortran*, 1991.

[45] Thinking Machines Corp., Cambridge, MA. *CM Fortran User's Guide for the CM-5 (v 1.1.3), (v. 2.1, Feb. 1994)*, January 1992.

[46] Thinking Machines Corp., Cambridge, MA. *CM5 Technical Summary*, November 1992.

[47] Thinking Machines Corp., Cambridge, MA. *CMSSL for CM Fortran: CM-5 Edition (v 3.1)*, June 1993.

[48] Thinking Machines Corp., Cambridge, MA. *Using the CMAX Converter (v 1.0)*, July 1993.

[49] Thinking Machines Corp., Cambridge, MA. *CM-5 CM Fortran Performance Guide (v. 2.1)*, 1994.

[50] P. A. Tirkas, C. A. Balanis, and R.A. Renaut. Higher order absorbing boundary conditionss for the finite-difference time-domain method. *IEEE Trans. on Ant. & Prop.*, 40(10):1215–1222, October 1992.

[51] Lloyd. N. Trefethen. Numerical computation of the Schwarz-Christoffel transformation. *SIAM J. Sci. Stat. Comput.*, 1(1):82–102, 1980.

[52] Lloyd N. Trefethen. *SCPACK User's Guide (NETLIB Document)*, 1983.

[53] L.N. Trefethen and L. Halpern. Well-posedness of one-way wave equations and absorbing boundary conditions. *Math. Comput.*, 47:421–435, 1986.

[54] L.N. Trefethen and M.R. Trummer. An instability phenomenon in spectral methods. *SIAM J. Numer. Anal.*, 24(5):1008–1023, 1987.

[55] A Vafidis, F. Abramovici, and E.R. Kanasewich. Elastic wave propagation using fully vectorized high order finite-difference algorithms. *Geophysics*, 57(2), 1992.

[56] R. Vichnevetsky and J.B. Bowles. *Fourier Analysis of Numerical Approximation of Hyperbolic Equations.* SIAM, 1982.

[57] Jean Virieux. Sh-wave propagation in heterogeneous media: Velocity-stress finite-difference method. *Geophysics*, 49(11), 1984.

[58] J.L. Walsh. *Interpolation and approximation by rational functions in the complex domain.* Providence, Rhode Island, 1956.

[59] J.A.C. Weideman and L.N. Trefethen. The eigenvalues of second-order spectral differentiation matrices. *SIAM J. Numer. Anal.*, 25:1279–1298, 1988.

[60] Qin Zhang. *Acoustic Pulse Diffraction by Curved And Planar Structures With Edges.* PhD thesis, The University of British Columbia, Dept. of Electrical Engineering, 1990.

[61] Shengkai Zhao. *Chebyshev Spectral Methods for Potential Field Computation.* PhD thesis, The University of British Columbia, August 1993.