

NONLINEAR ACOUSTIC PULSE PROPAGATION
IN RANGE-DEPENDENT UNDERWATER
ENVIRONMENTS

by
Joseph T. Maestas

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Mechanical Engineering).

Golden, Colorado

Date _____

Signed: _____

Joseph T. Maestas

Signed: _____

Dr. Jon Collis
Thesis Advisor

Signed: _____

Dr. John Berger
Thesis Advisor

Golden, Colorado

Date _____

Signed: _____

Dr. Greg Jackson
Professor and Head
Department of Mechanical Engineering

ABSTRACT

The nonlinear progressive wave equation (NPE) is a time-domain formulation of Eulers fluid equations designed to model low angle wave propagation using a wave-following computational domain. The standard formulation consists of four separate mathematical quantities that physically represent refraction, nonlinear steepening, radial spreading, and diffraction. The latter two of these effects are linear whereas the steepening and refraction are nonlinear. This formulation recasts pressure, density, and velocity into a single variable a dimensionless pressure perturbation which allows for greater efficiency in calculations. The wave-following frame of reference permits the simulation of long-range propagation that is useful in modeling the effects of blast waves in the ocean waveguide. Nonlinear effects such as weak shock formation are accurately captured with the NPE. The numerical implementation is a combination of two numerical schemes: a finite-difference Crank-Nicholson algorithm for the linear terms of the NPE and a flux-corrected transport algorithm for the nonlinear terms. In this work, an existing implementation is extended to allow for a penetrable fluid bottom. Range-dependent environments, characterized by sloping bathymetry, are investigated and benchmarked using a rotated coordinate system approach.

TABLE OF CONTENTS

| | |
|---|------|
| ABSTRACT | iii |
| LIST OF FIGURES AND TABLES | vi |
| LIST OF SYMBOLS | viii |
| LIST OF ABBREVIATIONS | ix |
| ACKNOWLEDGMENTS | x |
| CHAPTER 1 INTRODUCTION | 1 |
| CHAPTER 2 BACKGROUND | 6 |
| 2.1 Derivation of the NPE | 6 |
| 2.2 Numerical Schemes | 13 |
| 2.2.1 Flux-Corrected Transport | 14 |
| 2.2.2 Finite-Difference Method | 16 |
| 2.3 Range-dependent Approaches | 19 |
| 2.3.1 Stair-step Method | 20 |
| 2.3.2 Terrain-following Coordinates | 21 |
| 2.3.3 Rotated Coordinates | 23 |
| 2.4 Thesis Introduction | 24 |
| CHAPTER 3 IMPLEMENTATION | 26 |
| 3.1 Boundary Conditions | 26 |
| 3.2 Penetrable Ocean Bottom | 28 |
| 3.2.1 Artificial Absorbing Layer | 29 |
| 3.3 Computer Programs | 31 |

| | | |
|------------------|---------------------------------------|----|
| 3.3.1 | Serial Code | 31 |
| 3.3.2 | Parallel Code | 32 |
| 3.3.3 | Parallel Performance | 34 |
| 3.4 | Solution Benchmarking | 36 |
| CHAPTER 4 | SLOPED OCEAN BATHYMETRY | 41 |
| 4.1 | Approach | 41 |
| 4.2 | Numerical Implementation | 45 |
| 4.3 | Solution Benchmarking | 48 |
| CHAPTER 5 | CONCLUSIONS AND FUTURE WORK | 52 |
| 5.1 | Conclusions | 52 |
| 5.2 | Future Work | 53 |
| REFERENCES CITED | | 55 |
| APPENDIX - | A: COMPUTER PROGRAM LAYOUT | 57 |

LIST OF FIGURES AND TABLES

| | | |
|-------------|---|----|
| Figure 2.1 | Sloped bottoms are modeled as a series of range-independent steps. | 21 |
| Figure 2.2 | The single scattering solution retains transmitted energy and discards reflected energy at the vertical interface. | 21 |
| Figure 2.3 | In the rotated coordinate approach, the grid is rotated to match the ocean bottom and the surface is approximated as stair steps. | 24 |
| Figure 3.1 | Grid used for applying bottom boundary condition. The empty circle represents the non-physical grid point. | 27 |
| Figure 3.2 | Grid used for applying interface conditions. The empty circles represent non-physical grid points. | 29 |
| Figure 3.3 | A schematic representing a typical computational domain. The darker circles represent increased dampening. | 32 |
| Figure 3.4 | An example of the neighbor communication method used in matrix-vector multiplication. | 33 |
| Figure 3.5 | Total computation time for various numbers of processors. | 35 |
| Figure 3.6 | Parallel speedup for various numbers of processors. | 35 |
| Figure 3.7 | Parallel efficiency for various numbers of processors. | 36 |
| Figure 3.8 | Environment used in range-independent benchmark comparison. | 37 |
| Figure 3.9 | Propagation of the pulse within the Pekeris waveguide. | 38 |
| Figure 3.10 | Results of range-independent environment at receiver depth of 20 m. | 38 |
| Figure 3.11 | Results of range-independent environment at receiver depth of 50 m. | 39 |
| Figure 3.12 | Propagation of the pulse within the Pekeris waveguide. Nonlinear effects are included. | 40 |
| Figure 3.13 | Results of range-independent environment at receiver depth of 20 m. Non-linear effects are included in the NPE model. | 40 |

| | | |
|-------------|---|----|
| Figure 3.14 | Results of range-independent environment at receiver depth of 50 m. Non-linear effects are included in the NPE model. | 40 |
| Figure 4.1 | Computational approach for treating the sloped upper boundary. | 46 |
| Figure 4.2 | Progression of the computational domain during upslope propagation. | 47 |
| Figure 4.3 | Progression of the computational domain during downslope propagation. | 47 |
| Figure 4.4 | Environment used in upslope propagation benchmark comparison. | 48 |
| Figure 4.5 | Results of upslope environment at receiver depth of 20 m. | 49 |
| Figure 4.6 | Results of upslope environment at receiver depth of 50 m. | 49 |
| Figure 4.7 | Environment used in downslope propagation benchmark comparison. | 50 |
| Figure 4.8 | Results of downslope environment at receiver depth of 20 m. | 51 |
| Figure 4.9 | Results of downslope environment at receiver depth of 50 m. | 51 |
| Figure 5.1 | An interpolator-extrapolator scheme is used to treat variable slopes. | 54 |
| Table 3.1 | Parallel Performance | 35 |

LIST OF SYMBOLS

| | |
|--|--------------|
| material derivative | D_t |
| partial derivative with respect to time | ∂_t |
| partial derivative with respect to the x dimension | ∂_x |
| partial derivative with respect to the y dimension | ∂_y |
| partial derivative with respect to the z dimension | ∂_z |
| coefficient of nonlinearity | β |
| sound speed | c |
| ambient sound speed | c_0 |
| first perturbation in sound speed | c_1 |
| density | ρ |
| pressure | p |
| velocity components in Euler's equations | v_i |
| dimensionless density perturbation | R |
| dimensionless pressure perturbation | Q |
| horizontal grid spacing | Δx |
| vertical grid spacing | Δz |
| time step | Δt |
| range direction in rotated coordinates | \tilde{x} |
| depth direction in rotated coordinates | \tilde{z} |
| angle of sloped ocean bottom | θ |

LIST OF ABBREVIATIONS

| | |
|---|-----|
| Nonlinear Progressive Wave Equation | NPE |
| Parabolic Equation | PE |
| Flux-Corrected Transport | FCT |
| Range-dependent Acoustic Model | RAM |
| Khokhlov-Zabolotskaya-Kuznetsov model | KZK |
| Finite-Difference | FD |
| Crank-Nicolson | CN |
| meters | m |
| seconds | s |
| kilometers | km |

ACKNOWLEDGMENTS

I would like to thank the Colorado Louis Stokes Alliance for Minority Participation (CO-AMP) for awarding me the Bridge to the Doctorate Fellowship which partially funded this work. This thesis would not have succeeded without the guidance of Dr. Jon Collis.

CHAPTER 1

INTRODUCTION

The nonlinear progressive wave equation (NPE) is a time-domain wave propagation model used to describe an acoustic pulse by means of a moving computational domain that tracks the wave front (i.e. Lagrangian coordinates). This model accounts for refraction and nonlinear steepening which are both nonlinear effects, as well as diffraction and in some cases radial spreading, which are linear effects. It was originally developed to describe long-range underwater acoustic propagation from high intensity sources (such as explosions) due to its efficient moving grid formulation. The NPE is derived in terms of a single variable from Euler's conservation of mass and momentum equations with the assumptions that associated shocks are weak and that propagation is strongest in the preferred direction of propagation (range direction) (McDonald *et al.*, 1994). The assumption that propagation is strongest in range will be referred to as the small angle approximation. This means the NPE is most effective at modeling acoustic pulses at distances sufficiently far away from the source from hundreds of meters to kilometers in range. The distance at which the waveform becomes planar (small angle propagation) depends on the problem.

The NPE is implemented using two types of numerical schemes: finite-difference (FD) methods are used to treat the linear terms, and a flux-corrected transport (FCT) algorithm is used to treat the nonlinear terms. Such an algorithm requires that for each time step, at each grid point, a flux limiter be calculated and used to refine the nonlinear behavior of the model. The numerical scheme implemented in this thesis is a combination of the two numerical methods (FD and FCT) into a Crank-Nicolson approach.

Generally, in ocean acoustics, the environment is defined by depth and range: depth, or the z-axis, starts from zero at the surface and positively increases as depth increases, and range, or the x-axis, is specified as the direction of propagation. An important aspect of modeling ocean acoustics is being able to treat realistic underwater environments. Realistic

environments are made up of non-flat ocean bottom topologies, varying medium properties, and non-rigid bottom sediment interfaces. This thesis will look at adapting the NPE model to account for these environmental conditions in two-dimensional settings. Underwater environments can be characterized as depth-dependent, range-dependent, or spatially three-dimensionally dependent. In depth-dependent environments, the properties of the fluid (density and speed of sound) can vary with depth only. In range-dependent environments, the properties of the fluid will vary with range, and the geometry can change with range as well such as with variably sloping ocean bottoms. The NPE is naturally able to handle depth-dependence, and recent research (McDonald & Piacsek, 2011) has seen adjustments made to the NPE that allow for weak range-dependence, in the form of a variable sound speed profile. This range-dependent technique is only meant to treat small variations in medium properties, so it cannot handle the sloping bottom case (the impedance mismatch between the sediment layer and fluid is too great).

Traditionally, range-dependent environments were treated using either range discretization methods, coordinate transformation methods, or a combination of both. One range discretization method known as the stair-stepping approach splits the domain into a series of range-independent steps and then matches interface conditions between range-independent regions. This procedure is difficult to implement into the NPE model because the moving computational domain cannot naturally treat vertical interfaces. Coordinate transformation approaches consist of at least two main methods, terrain-following coordinates (also known as mapping solutions) and coordinate rotation solutions. Terrain-following coordinates are implemented by shifting the z -coordinate to match the ground height, $h(x)$, which effectively shifts the computational domain up and down with the terrain surface (Leissing, 2009). This approach is well suited for environments with rigid bottoms, but is ineffective for most ocean acoustics problems where penetrable bottoms are present because errors develop at the fluid-bottom interface. The rotated coordinate approach rotates the computational domain so that its axes are aligned with the sloped bottom; as a result the ocean surface

becomes sloped and is approximated as a series of stair steps. This technique is optimal for the underwater NPE model, although certain provisions will need to be made to account for the upper boundary.

Problems in ocean acoustics are solved either in the time-domain or the frequency-domain, dependent on application. Solutions in the time-domain are advantageous for blast waves because explosions are inherently broadband, that is, the source emits a wide band of frequencies. It is possible to use Fourier analysis to separate the source signal into its individual frequency components and run a frequency-domain model for each component, but this is inefficient when so many frequencies are present. Further, nonlinearities cause interaction among frequency components which prevents a frequency domain approach (McDonald & Kuperman, 1987). This frequency interaction problem can be resolved, but the correction is tedious and unrewarding for cases containing many frequencies.

Currently, there are a number of time and frequency-domain methods used for modeling underwater acoustics that include analytic solutions, parabolic wave equation (PE) approaches, Khokhlov-Zabolotskaya-Kuznetsov (KZK) approaches, and direct, first principles modeling amongst others. While each method has its own strengths, the NPE has proven to be the most effective for describing nonlinear acoustic pulse propagation. These methods are discussed briefly here in the following paragraphs.

There are a number of analytic-based computational solutions that have been developed for ocean acoustics which include wavenumber integration methods, normal mode methods, and ray methods. Each of these methods assumes a time-harmonic source and so is based on the Helmholtz equation, which is the frequency domain wave equation. These methods are able to handle range-dependent environments through both the stair-stepping and the coordinate transformation approaches, but they are unable to efficiently capture nonlinear effects (Porter, 1992).

The parabolic wave equation (PE) is a popular method for modeling underwater acoustics in the frequency-domain. The PE is based on an approximation to the elliptic Helmholtz

equation through a differential operator factorization and is solved as an initial value problem in range with boundary conditions in depth. While the NPE has been referred to as a parabolic equation (McDonald & Kuperman, 1987) because it is a one-way wave equation, it differs from classical underwater acoustic PE solutions in that there is not an operator factorization into a product of incoming and outgoing wave energy. Like the analytic solutions, this method can handle range-dependent environments because it allows for variations in the medium at each range step (stair-stepping approach), but also like analytic solutions, the PE method cannot capture nonlinear effects because it is derived from the linear Helmholtz equation (Kusel, 2005).

The Khokhlov-Zabolotskaya-Kuznetsov (KZK) approaches use an equation that is similar to the NPE which is able to capture nonlinear effects, but, like the NPE, it is not able to effectively model range-dependent environments. There have been recent developments, however, that allow for the modeling of weak range-dependence, that is, small variations in medium properties (Jing & Cleveland, 2007), but these methods cannot handle sloping bottoms where variations in medium properties are large.

Direct, first principles methods such as the finite-element (FEM) and finite-volume methods have been used to model underwater acoustic problems with high fidelity because they model the governing equations (Euler's equations) directly. These methods often require large amounts of computing power, so are impractical for use with computational domains with length scales on the order of kilometers. These types of solutions, though severely limited computationally, have proven useful in benchmarking solutions for small-scale canonical scenarios (Collis *et al.*, 2009).

The NPE is well suited for long-range underwater acoustic pulse propagation because it provides a good balance of accuracy and efficiency. This method is more accurate than analytic methods and the PE approach because it can capture the nonlinear effects that often occur with high intensity sources. This method is more efficient than direct modeling because the moving grid allows for only a small portion of the entire domain to be discretized. The

NPE method is also more straightforward than the KZK method, for which modeling ocean bottom sediment layers can be difficult. A critical aspect missing from NPE development is the treatment of range-dependent underwater environments; this will be addressed in this thesis.

The organization of this thesis is as follows: Chapter 2 contains a derivation of the NPE given in Section 4.1, followed by an explanation of the numerical schemes used to simulate propagation. Section 2.2 gives details on the FCT algorithm used on nonlinear terms and the FD scheme used on linear terms. Chapter 3 contains information about the boundary, interface, and initial conditions used for simulation. Chapter 4 discusses the method used for handling sloped ocean bottoms. Finally, Chapter 5 gives conclusions and possible future work.

CHAPTER 2

BACKGROUND

This chapter will give a detailed derivation of the nonlinear progressive wave equation (NPE), following largely from the earlier work of (McDonald *et al.*, 1994). After deriving the equation, the FCT algorithm (McDonald & Ambrosiano, 1984) will be described, and FD methods used will be explained. Brief descriptions of various range-dependent solutions are given and, finally, the main goals of this work and approach to be taken are discussed.

2.1 Derivation of the NPE

The NPE is derived by taking Euler’s fluid dynamical equations and combining them, using the adiabatic equation of state, into an equation of a single unknown dependent variable. This equation, written in terms of one dependent variable, is useful in the area of computational ocean acoustics because it allows for efficient computation when dealing with long-range propagation (greater than 1 km). Further, the NPE is capable of describing several different physical effects of propagation such as diffraction, refraction, and nonlinear steepening. The approach used in the derivation presented here is taken from a previously published paper (McDonald *et al.*, 1994). One assumes that the background medium velocity is zero and that the pulse is at sufficiently far distance from the source (a distance where pressures may be assumed to have “acoustic” values). Throughout this derivation, the Einstein summation convention, $c_i x^i = \sum c_i x^i = c_1 x^1 + c_2 x^2 + c_3 x^3$, and the shorthand spatial derivative notation, $\frac{\partial}{\partial x_i} = \partial_i$, will be employed. The Euler equations of continuity of mass and momentum and the adiabatic equation of state are given by

$$\frac{\partial \rho}{\partial t} + \partial_i(\rho v_i) = 0 \tag{2.1}$$

$$\partial_t(\rho v_i) + \partial_j(\rho v_i v_j + p \delta_{ij}) = 0, \quad i, j = 1, 2, 3 \tag{2.2}$$

$$p \approx p_0 + c^2(\rho - \rho_0) + \frac{1}{2} \left[\frac{\partial c^2}{\partial \rho} \right]_0 (\rho - \rho_0)^2, \quad (2.3)$$

where x_i and t represent the spatial and temporal dimensions ($i = 1, 2, 3$ represent the x , y , and z dimensions, respectively), p and v_i are pressure and velocity values, ρ is the fluid density, c is the speed of sound in the fluid (note that in (2.3) $c^2 = \frac{\partial p}{\partial \rho}$ such as seen in the acoustic wave equation), and δ_{ij} is the Kronecker delta. Expansion of (2.2) gives

$$v_i \frac{\partial \rho}{\partial t} + \rho \frac{\partial v_i}{\partial t} + v_j \rho \partial_j v_i + v_i \partial_j (\rho v_j) + \delta_{ij} \partial_j p = 0 \quad (2.4)$$

Substituting (2.1) into (2.4), using the definition of the Kronecker delta function, and dividing by ρ gives

$$\frac{\partial v_i}{\partial t} + v_j \partial_j v_i + \frac{1}{\rho} \partial_i p = 0 \quad (2.5)$$

Differentiating (2.1) with respect to time gives

$$\frac{\partial^2 \rho}{\partial t^2} = -\partial_i \left(\rho \frac{\partial v_i}{\partial t} + v_i \frac{\partial \rho}{\partial t} \right) \quad (2.6)$$

Substituting Eqns. (2.1) and (2.5) into (2.6) gives the nonlinear wave equation, written in terms of the five unknowns (ρ , p , and v_i , $i = 1, 2, 3$).

$$\frac{\partial^2 \rho}{\partial t^2} = \partial_i^2 p + \partial_i \partial_j (\rho v_i v_j). \quad (2.7)$$

This nonlinear wave equation is then recast in a moving frame that tracks the traveling pulse (a Lagrangian approach). This methodology assumes that the majority of the acoustic energy is propagated in the x -direction. The new variables (x' , y' , z' , t') will be used to represent the moving-frame coordinate system, while the variables (x , y , z , t) will be representative of the stationary system. The new computational grid will move at a constant speed c_0 (representative of the average sound speed in the fluid) in the x -direction. The nonlinear

wave equation is simplified by employing a perturbation-like scaling where the new variables are scaled in order to retain largest terms and discard the small terms. The new variables are given by:

$$\begin{aligned}
x' &:= x - c_0 t \\
y' &:= \epsilon^{1/2} y \\
z' &:= \epsilon^{1/2} z \\
t' &:= \epsilon t,
\end{aligned} \tag{2.8}$$

where the scaling factor $\epsilon \ll 1$ is used to emphasize the predominance of propagation in the x -direction. The variable t is transformed using a factor of ϵ as opposed to $\epsilon^{1/2}$ in order to weaken the overall dependence of the solution on the new variable t' ; it is assumed that the evolution of the pulse (within the moving frame) is small compared to the velocity of the moving frame.

The transformation from Eulerian to Lagrangian coordinates requires the use of a material derivative in time. The material derivative, $D_{t'} = \partial_t + c_0 \partial_x$, gives the time derivative while following a specific particle in the fluid. The differential operators are now written as:

$$\begin{aligned}
\partial_x &:= \partial_{x'} \\
\partial_y &:= \epsilon^{1/2} \partial_{y'} \\
\partial_z &:= \epsilon^{1/2} \partial_{z'} \\
\partial_t &:= \epsilon D_{t'} - c_0 \partial_x = \epsilon D_{t'} - c_0 \partial_{x'}
\end{aligned} \tag{2.9}$$

The ∂_t term is formulated by rearranging the formula given above for $D_{t'}$; this implies that the effects of the advective term, $c_0 \partial_x$, will be carried throughout the derivation. By using the change of variables in (2.8) and differential operators in (2.9), Eqns. (2.1), (2.2), and (2.7) can be expressed in terms of the new, moving coordinate system:

$$(\epsilon D_{t'} - c_0 \partial_{x'}) \rho = -\partial_{x'}(\rho v_1) - \epsilon^{1/2} \partial_{y'}(\rho v_2) - \epsilon^{1/2} \partial_{z'}(\rho v_3), \tag{2.10}$$

$$\begin{aligned}
(\epsilon D_{t'} - c_0 \partial_{x'}) v_i = & - [v_1 \partial_{x'} v_i + \epsilon^{1/2} v_2 \partial_{y'} v_i + \epsilon^{1/2} v_3 \partial_{z'} v_i] \\
& - \frac{1}{\rho} [\delta_{i1} \partial_{x'} p + \delta_{i2} \epsilon^{1/2} \partial_{y'} p + \delta_{i3} \epsilon^{1/2} \partial_{z'} p],
\end{aligned} \tag{2.11}$$

and

$$\begin{aligned}
(\epsilon^2 D_{t'}^2 - 2\epsilon c_0 D_{t'} \partial_{x'} + c_0^2 \partial_{x'}^2) \rho = & (\partial_{x'}^2 + \epsilon(\partial_{y'}^2 + \partial_{z'}^2)) p \\
& + \partial_{x'}^2 (\rho v_1^2) + 2\epsilon^{1/2} \partial_{x'} \partial_{y'} (\rho v_1 v_2) \\
& + 2\epsilon^{1/2} \partial_{x'} \partial_{z'} (\rho v_1 v_3) + \epsilon \{ \partial_{y'}^2 (\rho v_2^2) + \partial_{z'}^2 (\rho v_3^2) \\
& + 2\partial_{y'} \partial_{z'} (\rho v_2 v_3) \}.
\end{aligned} \tag{2.12}$$

The primes are dropped to ease further notation. Perturbations are applied to the five dependent variables ρ , p , and v_i to represent an acoustic disturbance. Naive expansions are used for ρ and p and a square root expansion is used for v_i (a relationship in agreement with linear acoustic theory):

$$\begin{aligned}
\rho & \approx \rho_0 + \epsilon \rho_1 + \epsilon^2 \rho_2 \\
p & \approx p_0 + \epsilon p_1 + \epsilon^2 p_2 \\
v_i & \approx \epsilon v_{1i} + \epsilon^{3/2} v_{2i} + \epsilon^2 v_{3i},
\end{aligned} \tag{2.13}$$

where subscripts $i = 1, 2, 3$ denote the x , y , and z coordinates and a subscript of zero denotes a constant, ambient quantity. For each velocity variable v_i each term in the expansion represents a small perturbation to that velocity in the i^{th} direction. Notice that v_{0i} is omitted due to the assumption that the background velocity is absent; this term will be retained in an investigation in Chapter 3. Each of the subscripted values in (2.13), except the constant, ambient values, is in general a function of x , y , z , and t . Substituting the appropriate asymptotic expansions into the continuity equation (2.10), gives

$$\begin{aligned}
& (\epsilon D_t - c_0 \partial_x) (\rho_0 + \epsilon \rho_1 + \epsilon^2 \rho_2) \\
& = -\partial_x ((\rho_0 + \epsilon \rho_1 + \epsilon^2 \rho_2) (\epsilon v_{11} + \epsilon^{3/2} v_{21} + \epsilon^2 v_{31})) \\
& \quad - \epsilon^{1/2} \partial_y ((\rho_0 + \epsilon \rho_1 + \epsilon^2 \rho_2) (\epsilon v_{12} + \epsilon^{3/2} v_{22} + \epsilon^2 v_{32})) \\
& \quad - \epsilon^{1/2} \partial_z ((\rho_0 + \epsilon \rho_1 + \epsilon^2 \rho_2) (\epsilon v_{13} + \epsilon^{3/2} v_{23} + \epsilon^2 v_{33})).
\end{aligned} \tag{2.14}$$

Looking at leading order (ϵ^0) terms yields

$$-c_0 \partial_x \rho_0 = -\partial_x \rho_0$$

which is trivially satisfied as all derivatives of ρ_0 (a constant) are zero. Equating first order (ϵ) terms gives

$$\partial_x v_{11} = \frac{c_0}{\rho_0} \partial_x \rho_1. \quad (2.15)$$

Eqn. (2.15) is integrated with respect to x to obtain

$$v_{11} = c_0 \frac{\rho_1}{\rho_0}. \quad (2.16)$$

Integration constants that may be functions of y and z are ignored by assuming that the expansions given in (2.13) are consistent with the narrow-angle approximation (propagation is mainly in the x -direction). Integration constants that are constant values are omitted as well since they disappear through differentiation at the end of the derivation. The values v_{ji} , $i = 2, 3$ in (2.12) are neglected because of the same narrow-angle assumption. Now, p is rewritten in terms of ρ by substituting the appropriate asymptotic expansions into Eqn. (2.12) to get

$$\begin{aligned} & (\epsilon^2 D_t^2 - 2\epsilon c_0 D_t \partial_x + c_0^2 \partial_x^2)(\rho_0 + \epsilon \rho_1 + \epsilon^2 \rho_2) \\ & = (\partial_x^2 + \epsilon(\partial_y^2 + \partial_z^2))(p_0 + \epsilon p_1 + \epsilon^2 p_2) \\ & + \partial_x^2((\rho_0 + \epsilon \rho_1 + \epsilon^2 \rho_2)(\epsilon^2 v_{11}^2 + \dots)) \\ & + 2\epsilon^{1/2} \partial_x \partial_y((\rho_0 + \epsilon \rho_1 + \epsilon^2 \rho_2)(\epsilon v_{11} + \dots)(\epsilon v_{12} + \dots)) \\ & + 2\epsilon^{1/2} \partial_x \partial_z((\rho_0 + \epsilon \rho_1 + \epsilon^2 \rho_2)(\epsilon v_{11} + \dots)(\epsilon v_{13} + \dots)) + O(\epsilon^3). \end{aligned} \quad (2.17)$$

In the above equation, it is assumed $D_t^2 \approx 0$ since it is quite small when compared to the $c_0 D_t \partial_x$ term. Retaining this term in the derivation leads to the ‘‘high-angle correction’’ which provides greater accuracy near the source. The leading order (ϵ^0) terms are again trivially

satisfied. Equating the first order (ϵ) terms gives

$$c_0^2 \partial_x^2 \rho_1 = \partial_x^2 p_1. \quad (2.18)$$

Integrating twice with respect to x yields

$$p_1 = c_0^2 \rho_1. \quad (2.19)$$

Now equate second-order (ϵ^2) terms, giving

$$-2c_0 D_t \partial_x \rho_1 + c_0^2 \partial_x^2 \rho_2 = \partial_x^2 p_2 + (\partial_y^2 + \partial_z^2) p_1 + \partial_x^2 \rho_0 v_{11}^2. \quad (2.20)$$

Integrating once with respect to x gives

$$-2c_0 D_t \rho_1 + c_0^2 \partial_x \rho_2 = \partial_x p_2 + \int_{x_f}^x (\partial_y^2 + \partial_z^2) p_1 \, dx + \partial_x \rho_0 v_{11}^2, \quad (2.21)$$

where the limits of integration are assigned according to the problem being considered.

The final step of the derivation involves writing the entire equation in terms of one variable, $R = \frac{\rho_1}{\rho_0}$ (a dimensionless density perturbation). To do this, p_2 and ρ_2 are replaced using the adiabatic equation of state, given in (2.3). The terms $(\rho - \rho_0)^n$ are evaluated under ambient conditions in order to be consistent with $[\frac{\partial c^2}{\partial \rho}]_0$ (recall that a subscript of zero indicates an ambient state). Also, an asymptotic expansion (with only a single, first-order correction) is introduced for the sound speed c and is given by

$$c \approx c_0 + \epsilon c_1(x, y, z), \quad (2.22)$$

where $c_1(x, y, z)$ is a spatial variation from c_0 . Substituting (2.13) and (2.22) into (2.3) yields

$$\begin{aligned} (p_0 + \epsilon p_1 + \epsilon^2 p_2) &= p_0 + (c_0 + \epsilon c_1)^2 (\epsilon \rho_1 + \epsilon^2 \rho_2) \\ &\quad + \frac{1}{2} \left[\frac{\partial c^2}{\partial \rho} \right]_0 (\epsilon \rho_1 + \epsilon^2 \rho_2)^2. \end{aligned} \quad (2.23)$$

Equating second-order (ϵ^2) terms and differentiating once with respect to x yields

$$\partial_x p_2 - c_0^2 \partial_x \rho_2 = 2c_0 c_1 \partial_x \rho_1 + \left[\frac{\partial c^2}{\partial \rho} \right]_0 \rho_1 \partial_x \rho_1. \quad (2.24)$$

Eqns. (2.24), (2.16), and (2.19), in conjunction with the substitution $R = \frac{\rho_1}{\rho_0}$, are used in order to transform (2.21) into the NPE given as

$$D_t R = -\partial_x \left[c_1 R + \frac{\beta c_0}{2} R^2 \right] - \frac{c_0}{2} \int_{x_f}^x (\partial_y^2 + \partial_z^2) R \, dx, \quad (2.25)$$

where

$$\beta = 1 + \left[\frac{\partial c}{\partial \rho} \right]_0 \frac{\rho_0}{c_0}$$

is a dimensionless constant known as the coefficient of nonlinearity, and it has been shown to have a value around 3.5 for the ocean. For the purposes of this paper, only two-dimensional propagation is considered; ∂_y^2 is disregarded so that the unknown R is a function of x , z , and t . The domain of consideration is, in general, defined by a grid (x, z) , with the positive x axis pointing to the right and the positive z axis pointing down. The lower limit of the integral, x_f , is taken to be a point ahead of the pulse where $R = \partial_x R = 0$. For the propagation of a discrete pulse, one has only to begin the integral ahead of the pulse (McDonald & Kuperman, 1987) (a fact that will be utilized in the formulation of FD methods following this section).

Each term in (2.25) represents a different physical process: the process of refraction is described by $-\partial_x(c_1 R)$, nonlinear steepening by $-\partial_x(\frac{\beta c_0}{2} R^2)$, and diffraction by $-\frac{c_0}{2} \int_{x_f}^x \partial_z^2 R \, dx$. Refraction can be defined as the change in a direction of a wave due to small changes in the sound speed of the medium; the refractive term is defined as such because it consists of the dependent variable R multiplied by c_1 , a value which represents a perturbation to the sound speed. The nonlinear steepening term is defined as such because it contains the dependent variable R nonlinearly; thus β is called the ‘‘coefficient of nonlinearity’’. Diffraction can be defined as how a wave will bend after encountering an obstacle; thus the diffraction term can be so named because it describes how wave propagation in one direction is affected by

factors described in another direction.

The different mathematical representations of each physical term naturally leads to different numerical treatments for each term: the refraction and nonlinear steepening terms (hereafter referred to as the nonlinear terms) are treated with FCT, and the results obtained are then used as a correction to the diffraction term (hereafter referred to as the linear term) which is treated by using the trapezoidal rule to approximate the integral, a second-order FD formula to approximate the z derivative, and finally a Crank-Nicolson (CN) update to march forward in time.

2.2 Numerical Schemes

Numerical evaluation of the NPE involves discretizing the solution on a computational domain. Recall that effects due to changes in the y direction are ignored, so the general domain described is $x_0 \leq x \leq x_N$ meters (m) wide and $z_0 \leq z \leq z_M$ m deep, where the endpoints of each interval are finite values. In general, $x_0 = c_0 t$, the distance traveled by the grid (with a flat ocean bottom), and z_0 equals zero, a common designation for the ocean surface. Also, to simplify further calculations the following definitions are made: $W \equiv x_N - x_0$, the width of the computational domain, and $D \equiv z_M - z_0$, the depth or height of the computational domain. Each direction is represented by a uniform grid such that $x_i = i\Delta x$, $i = 0, 1, \dots, N$, and $z_j = j\Delta z$, $j = 0, 1, \dots, M$. Time is also discretized on a uniform grid such that $t_n = n\Delta t$, $n = 0, 1, \dots, \tau$; time begins at $t = 0$ s and simulations are carried out to some finite time value $t = t_\tau$ s. The solution is then given on the domain, $(x_0 \leq x \leq x_N, z_0 \leq z \leq z_M, 0 \leq t \leq t_\tau)$, as $R(x_i, z_j, t_n)$. This discrete grid will be used throughout this section. Initial and boundary conditions are described in the next chapter.

The nonlinear terms in the NPE depend solely on x (due to the x derivative), so FCT is applied to each j th column of constant depth. Thus, for a two-dimensional problem, FCT is applied $M + 1$ times for a single time step. In contrast, the linear terms of the NPE depend on both x (due to the integral) and z (due to the second z derivative), therefore the FD

scheme is applied on all grid points, or $(N + 1)(M + 1)$ times, for every time step. Details of each method are given below.

2.2.1 Flux-Corrected Transport

The FCT method (McDonald & Ambrosiano, 1984) is a hybrid scheme, resulting from a combination of first- and second-order upwind discretization methods. Central to the scheme is the computation of flux terms, so chosen to preserve monotonicity in the numerical solution, a property that will forbid the generation of new extrema (results that would be numerically inaccurate) or the enhancement of old extrema (results that would be physically unrealistic). Fluxes are computed at intermediate grid points $x_{i+\frac{1}{2}}$, placed at the midpoint of x_i and x_{i+1} . Using these intermediate points, the first portion of the algorithm is first-order, as calculations centered at $x_{i+\frac{1}{2}}$ relate to one grid point to the immediate left or right (x_i or x_{i+1}); the results are first-order accurate, but not enough to prevent the rapid advancement of numerical errors. To correct such errors, second-order results centered at $x_{i+\frac{1}{2}}$ that now relate to two grid points to the immediate left or right (x_{i-1} and x_i or x_{i+1} and x_{i+2}) are computed. The first- and second-order results are then combined to give a stable scheme that can be used to accurately deal with any nonlinearities present in the solution to the NPE.

The portion of (2.25) to be evaluated using the FCT algorithm is written as

$$D_t R = -\partial_x \left(c_1 R + \frac{\beta c_0}{2} R^2 \right). \quad (2.26)$$

Equations presented in this section are calculated for all $i = 0, 1, \dots, N$ at fixed j , and are used toward obtaining the solution at time t_{n+1} . One begins the scheme by computing

$$\phi_i^n = c_1 R_i^n + \frac{\beta c_0}{2} (R_i^n)^2. \quad (2.27)$$

Calculations now begin at the intermediate grid points; a characteristic direction variable, $w_{i+\frac{1}{2}}^n$, and the first-order fluxes, $f_{i+\frac{1}{2}}^n$, are computed using the stored values of ϕ_i^n . These

terms are defined as:

$$w_{i+\frac{1}{2}}^n = (\phi_{i+1}^n - \phi_i^n)(R_{i+1}^n - R_i^n) \quad (2.28)$$

$$f_{i+\frac{1}{2}}^n = \frac{\Delta t}{\Delta x} \begin{cases} \phi_i^n & \text{if } w_{i+\frac{1}{2}}^n \geq 0 \\ \phi_{i+1}^n & \text{if } w_{i+\frac{1}{2}}^n < 0 \end{cases} . \quad (2.29)$$

The fluxes $f_{i+\frac{1}{2}}^n$ are corrected using the characteristic speed, $u_i^n = \partial_x(\phi_i^n)$. If $u_i^n < 0$ and $u_{i+1}^n > 0$, then a correction is applied such that

$$f_{i+\frac{1}{2}}^n = \frac{1}{2} \left(f_{i+\frac{3}{2}}^n + f_{i+\frac{1}{2}}^n - \frac{\Delta t}{\Delta x} (R_{i+1}^n - R_i^n)(u_{i+1}^n - u_i^n) \right), \quad (2.30)$$

otherwise the flux remains unchanged. Next, one computes and saves R_i^* (an update that preserves monotonicity) and ϕ_i^* (ensuring the stability of the second-order scheme):

$$R_i^* = R_i^n - f_{i+\frac{1}{2}}^n + f_{i-\frac{1}{2}}^n \quad (2.31)$$

$$\phi_i^* = \phi_i^n - \frac{1}{2} u_i^n (f_{i+\frac{1}{2}}^n - f_{i-\frac{1}{2}}^n). \quad (2.32)$$

The second-order fluxes, $F_{i+\frac{1}{2}}^n$, are then computed as

$$F_{i+\frac{1}{2}}^n = \frac{\Delta t}{2\Delta x} \begin{cases} 3\phi_i^* - \phi_{i-1}^* & \text{if } w_{i+\frac{1}{2}}^n \geq 0 \\ 3\phi_{i+1}^* - \phi_{i+2}^* & \text{if } w_{i+\frac{1}{2}}^n < 0 \end{cases} . \quad (2.33)$$

These fluxes are also corrected using the characteristic speed as in (2.30). Using $f_{i+\frac{1}{2}}^n$ and $F_{i+\frac{1}{2}}^n$, one can compute the flux correction $\delta f_{i+\frac{1}{2}}^n$ as

$$\delta f_{i+\frac{1}{2}}^n = F_{i+\frac{1}{2}}^n - f_{i+\frac{1}{2}}^n, \quad (2.34)$$

which is filtered using a flux limiter, given by

$$\delta f_{i+\frac{1}{2}}^n = \sigma_{i+\frac{1}{2}}^n \max(0, \min[|\delta f_{i+\frac{1}{2}}^n|, \sigma_{i+\frac{1}{2}}^n (R_{i+2}^* - R_{i+1}^*), \sigma_{i+\frac{1}{2}}^n (R_i^* - R_{i-1}^*)]) \quad (2.35)$$

for $\sigma_{i+\frac{1}{2}}^n$, the sign of $\delta f_{i+\frac{1}{2}}^n$ (defined as one if $\delta f_{i+\frac{1}{2}}^n = 0$). It is necessary to apply the limiting function (or so-called filter) to $\delta f_{i+\frac{1}{2}}^n$ because the second-order scheme does not inherently

preserve monotonicity, as does the first-order scheme, otherwise $F_{i+\frac{1}{2}}^n$ could introduce numerical errors at this step. Further, the choice of filter is not unique; different problems may require more appropriate choices of filter. An appropriate filter may be chosen through trial and error (the filter used here is as described by (McDonald & Ambrosiano, 1984)). The scheme is then completed with the calculation

$$R_i^{n+1} = R_i^* - \delta f_{i+\frac{1}{2}}^n + \delta f_{i-\frac{1}{2}}^n. \quad (2.36)$$

In summary, to accurately numerically evaluate the nonlinear terms in the NPE one can follow the steps outlined in this section. The correct order of the steps is as follows: (2.27), (2.28), (2.30) (if the conditions placed on u_i^n and u_{i+1}^n are satisfied), (2.31), (2.33), (2.30) (with $f_{i+\frac{1}{2}}^n$ replaced by $F_{i+\frac{1}{2}}^n$ if the conditions placed on u_i^n and u_{i+1}^n are satisfied), then (2.34), (2.35), and (2.36). Through the use of selective upwinding methods (dependent upon the direction of the characteristic speed of the unknown at intermediate grid points) and appropriate filters applied to the combined fluxes, numerical errors that may result during evaluation will be corrected at each time step, preventing effects that may be physically inaccurate from developing.

2.2.2 Finite-Difference Method

Finite-difference (FD) methods are commonly used to numerically evaluate partial differential equations as they offer good accuracy and are relatively simple to implement. In some cases, such methods may be appropriate for only a few terms of the equation, and must then be combined with other evaluation methods. Such is the case when dealing with the NPE. A numerical scheme, resulting from a combination of FD approximations of derivatives, a trapezoid rule approximation for the integral, and a CN update, is used to evaluate the linear term not treated by the previously described FCT algorithm.

The portion of (2.25) to be evaluated using finite-differences is written as

$$D_t R = \frac{-c_0}{2} \int_{x_f}^x \partial_z^2 R \, dx. \quad (2.37)$$

This portion of the NPE is fundamentally different than the nonlinear portion described in the previous section; here, the unknown is dependent upon two spatial dimensions. This difference affects the discretization used, as discussed at the beginning of the section. Calculations are done at each grid point for all $i = 0, 1, \dots, N$ and $j = 0, 1, \dots, M$, and are used to obtain the solution at the next time step t_{n+1} . Also, recall that $D_t = \partial_t + c_0 \partial_x$ was introduced in the derivation of the NPE, where its dependence on ∂_x has already been accounted for in the NPE formulation; numerical approximations of D_t are then treated the same as those given for ∂_t . The FD formulas that approximate the derivatives in (2.37) are given by:

$$\partial_z^2 R_{i,j}^n \approx \frac{R_{i,j-1}^n - 2R_{i,j}^n + R_{i,j+1}^n}{\Delta z^2} \quad (2.38)$$

$$D_t R_{i,j}^n \approx \frac{R_{i,j}^{n+1} - R_{i,j}^n}{\Delta t}. \quad (2.39)$$

Employing the CN method (averaging the $\partial_z^2 R$ term in time) provides more accuracy and transforms (2.37) into

$$\partial_t R_{i,j}^n = \frac{-c_0}{2} \int_{x_f}^x \frac{1}{2} (\partial_z^2 R_{i,j}^n + \partial_z^2 R_{i,j}^{n+1}) \, dx. \quad (2.40)$$

Applying formulas (2.38) and (2.39) to (2.40) gives

$$\frac{R_{i,j}^{n+1} - R_{i,j}^n}{\Delta t} = \frac{-c_0}{4\Delta z^2} \int_{x_f}^x (R_{i,j-1}^n - 2R_{i,j}^n + R_{i,j+1}^n + R_{i,j-1}^{n+1} - 2R_{i,j}^{n+1} + R_{i,j+1}^{n+1}) \, dx. \quad (2.41)$$

Numerically, the upper limit of the integral is representative of a grid point x_i , and the lower limit is taken to be the right-most grid point $x_f = x_N$. The trapezoidal rule is then applied

to the integral such that

$$\int_{x_N}^{x_i} R_{i,j}^n dx \approx \frac{-\Delta x}{2} \left(R_{i,j}^n + R_{N,j}^n + 2 \sum_{k=i+1}^{N-1} R_{k,j}^n \right). \quad (2.42)$$

The fact that $R_{N,j}^n = 0$ (representative of a quiescent medium ahead of the pulse) is used to eliminate this term in the approximation. Substituting Eqn. (2.42) into Eqn. (2.41) and moving all unknown terms to the left hand side results in an $N \times M$ system of linear equations for $R_{i,j}^{n+1}$, given by

$$\begin{aligned} R_{i,j}^{n+1} &- \frac{c_0 \Delta t \Delta x}{8 \Delta z^2} (R_{i,j-1}^{n+1} - 2R_{i,j}^{n+1} + R_{i,j+1}^{n+1}) = R_{i,j}^n + \frac{c_0 \Delta t \Delta x}{8 \Delta z^2} (R_{i,j-1}^n - 2R_{i,j}^n + R_{i,j+1}^n) \\ &+ \frac{c_0 \Delta t \Delta x}{4 \Delta z^2} \sum_{k=i+1}^{N-1} (R_{k,j-1}^n - 2R_{k,j}^n + R_{k,j+1}^n + R_{k,j-1}^{n+1} - 2R_{k,j}^{n+1} + R_{k,j+1}^{n+1}). \end{aligned} \quad (2.43)$$

This system is now rewritten using matrix-vector notation. Let $\sigma = \frac{c_0 \Delta t \Delta x}{8 \Delta z^2}$, and consider $R_{i,j}^n$ for all $j = 0, 1, \dots, M$ and fixed i to create the $M \times 1$ column vector

$$\vec{R}_i^n = (R_{i,1}^n \ R_{i,2}^n \ \dots \ R_{i,M-1}^n \ R_{i,M}^n)^T.$$

The system (2.43) can then be written as the condensed system of N linear equations

$$(\mathbf{I} - \sigma \mathbf{A}) \vec{R}_i^{n+1} = (\mathbf{I} + \sigma \mathbf{A}) \vec{R}_i^n + 2\sigma \sum_{k=i+1}^{N-1} (\mathbf{A}(\vec{R}_k^n + \vec{R}_k^{n+1})) \quad (2.44)$$

where $\mathbf{A} = \begin{bmatrix} 2 & -1 & \dots & 0 \\ -1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 \\ 0 & \dots & -1 & 2 \end{bmatrix}$ is the $M \times M$ matrix resulting from approximating the z

derivative, and \mathbf{I} is the $M \times M$ identity matrix. The fact that this is a banded, tridiagonal system allows the use of specialized sparse matrix solvers, increasing efficiency. Boundary and interface conditions specific to a given problem are then incorporated in the matrix \mathbf{A} when discretized.

Defining the vector and matrix quantities in this way allows one to solve for a matrix of

unknown values

$$\mathbf{R}^{n+1} = [\vec{R}_1^{n+1} \ \vec{R}_2^{n+1} \ \dots \ \vec{R}_i^{n+1} \ \dots \ \vec{R}_{N-1}^{n+1} \ \vec{R}_N^{n+1}]$$

column-wise from right to left (a result that is representative of beginning the integral ahead of the pulse). That is, one first applies $R_{N,j}^{n+1} = 0$, then solves for $R_{N-1,j}^{n+1}$, and so on; thus all the terms in the summation are known quantities. The efficiency provided by the vectorization and the fact that CN methods are numerically stable (granted the choices of Δx , Δz and Δt are compliant with the Courant-Friedrichs-Lewy condition) give confidence that the scheme will not only provide accurate results when applied to the NPE but will provide them in an acceptable amount of computation time. This numerical formulation is appropriate for only the simplest of cases (perfectly reflecting boundaries). In order to account for more complex environments such as penetrable ocean bottoms, sediment layering, and sloping bathymetry certain adjustments need to be made to the numerical scheme; details of such cases will be provided in the following chapters.

2.3 Range-dependent Approaches

Typical ocean environments have sea floors with varying topographies that can significantly affect wave propagation, especially for shallow water environments. Most sea floors have a common structure which usually begin with a continental shelf in the shallow ocean region that continues to the continental slope – a region of steep decent into the deep ocean region – until reaching the ocean basin – a near constant-slope, deep water plain. It is important to note the typical slope angles found in ocean bathymetry are generally less than 10° (F. B. Jensen & Schmidt, 2005). Although these angles are small, they can greatly affect the wave propagation at long ranges and therefore cannot be disregarded.

There are three main methods used for treating variable bathymetry in computational ocean acoustics. The traditional approach is the stair-step approximation in which the environment is discretized into a series of range-independent regions. More recent approaches have used coordinate transformation methods such as terrain-following coordinates (coordi-

nate mapping) and rotated coordinates which offer greater accuracy.

2.3.1 Stair-step Method

A common method for treating range-dependence with one-way wave models is the stair-step approximation (Kusiel, 2005). In this technique, the range is discretized into a series of range-independent regions – regions where medium properties do not vary with range, but can vary with depth. For the case of sloping bathymetry, each region is split into an upper water section and a seafloor section, and the height of the seafloor section changes with subsequent range-independent regions as seen in Figure 2.1. The solution is marched through each region and interface conditions are enforced for both the vertical interfaces and the horizontal interface. The main conditions that need to be enforced for a penetrable fluid bottom are continuity of pressure and continuity of normal particle displacement (or velocity). Were the bottom elastic media, it would be necessary to add a zero tangential stress condition. These interface conditions are easily implemented for the horizontal case, but they are more difficult to implement at the vertical interfaces because one-way wave models are based on outgoing energy. Therefore, energy is difficult to conserve between each region.

One method for imposing interface conditions between regions is the single scattering approximation (Kusiel, 2005). This approximation uses a two-way wave model at each vertical interface to generate the reflected and transmitted fields. The reflected energy is then discarded and the transmitted energy is used to march the one-way wave model in the following region as seen in Figure 2.2.

The stair-step approximation is often employed with parabolic equation (PE) models because the single scattering technique can be applied with little effort due to the existence of efficient two-way PE models (Collins & Evans, 1992). This is not the case for the NPE model, however, and implementing this approximation may prove difficult. It may be possible to apply Euler’s equations (or perhaps Burger’s equation) directly at the interface in order

to generate the forward and back-scattered energy needed for the single scattering solution, but this has not been explored.

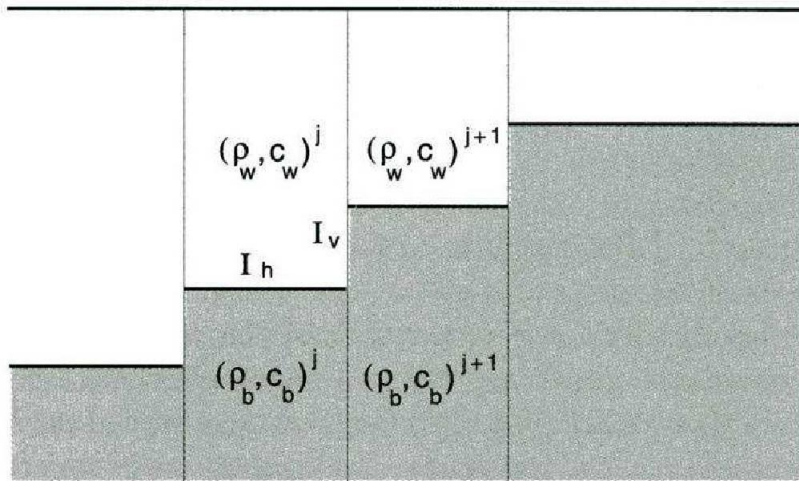


Figure 2.1: Sloped bottoms are modeled as a series of range-independent steps.

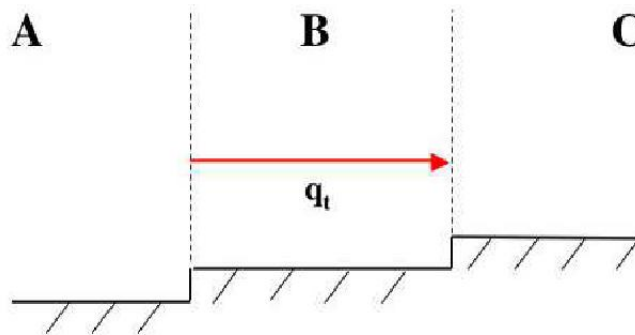


Figure 2.2: The single scattering solution retains transmitted energy and discards reflected energy at the vertical interface.

2.3.2 Terrain-following Coordinates

Other approaches for handling sloping bathymetry include coordinate transformation methods that shift or rotate the coordinate system to take into account the slope. These methods allow for vertical interfaces to be avoided altogether, so energy loss across range-independent regions is no longer an issue. One such method is the terrain-following coor-

dinate approach, also known as coordinate mapping (Outing, 2004). In this approach, the computational domain is shifted vertically so it aligns with the seafloor. This shift is done with the following change of variables:

$$x \rightarrow x, \tag{2.45}$$

$$z \rightarrow z - d(x), \tag{2.46}$$

where $d(x)$ is the depth of the seafloor as a function of range. This new set of variables is then carried through the derivation of the model in consideration and a new governing equation is generated. This new governing equation has many new terms that account for waveform shifting and changes in diffusion due to changes in seafloor depth. These additional terms are often difficult to implement, so certain assumptions are made to simplify the new governing equation. The main assumption is that the first and second derivatives of $d(x)$ are small, that is, the ocean bottom is smooth and its slope is relatively small as well.

This coordinate mapping technique has previously been applied to PE models for both fluid and elastic bottom layers (Outing, 2004), although an energy correction term was implemented to avoid interface errors. This method has also been implemented with the NPE for atmospheric acoustic modeling (Leissing, 2009). The case examined was acoustic pulse propagation over a slight hill with a maximum angle of about 12.4° and the surface approximated as a rigid boundary. Results agreed well with the numerical benchmark solution.

Terrain-following coordinates are often advantageous because they allow for the treatment of variably sloping ocean bottoms since $d(x)$ can be any continuous, smooth function of x . The main drawback to this approach is the increased complexity of the numerical solution. The additional terms in the governing equation make implementation impractical unless simplifying assumptions are made. If the sloped ocean bottom is penetrable (energy can move across the interface), these assumptions lead to errors developing at the sloped interface (Outing, 2004). Therefore, this approach is only appropriate for ocean bottoms with small slope angles and rigid seafloors.

2.3.3 Rotated Coordinates

Another coordinate transformation technique used for handling sloping ocean bottoms is the rotated coordinate approach (Outing, 2004). In this approach, the coordinate system is rotated such that the computational domain is parallel to the sloping ocean bottom. The ocean surface becomes a sloping line that is approximated as a series of stair steps as seen in Figure 2.3 below. The pressure release boundary condition $R = 0$ is applied to the horizontal areas of the steps. As the lengths of the vertical areas of the steps become small, $|R|$ becomes small in these areas by continuity (Outing, 2004). This means, for small slope angles, conditions at the vertical interfaces of the steps need not be explicitly imposed.

In general, the rotated coordinate approach is achieved by introducing a set of cylindrical coordinates,

$$\begin{Bmatrix} \tilde{x} \\ \tilde{z} \end{Bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{Bmatrix} x \\ z \end{Bmatrix}, \quad (2.47)$$

where θ is the angle of the sloped bottom, and \tilde{x} and \tilde{z} are tangent and normal components of the ocean bottom, respectively. This transformation is then carried through the derivation of the model in consideration and a new governing equation is generated. This new governing equation does not have any new additional terms, rather, there are $\cos \theta$ and $\sin \theta$ values multiplying the terms of the original governing equation. It should be noted that this approach is only able to treat constant sloped bottoms, and additional methods are needed to treat variable slopes.

The rotated coordinates approach has been successfully applied to PE models for both constant and variably sloping ocean bottoms (Outing, 2004). Variable slopes were treated with an interpolator-extrapolator scheme; this scheme marches the solution for one slope slightly past the endpoint in order to generate the initial condition for the solution of the next slope. Results agreed well with benchmark numerical models. The rotated coordinates

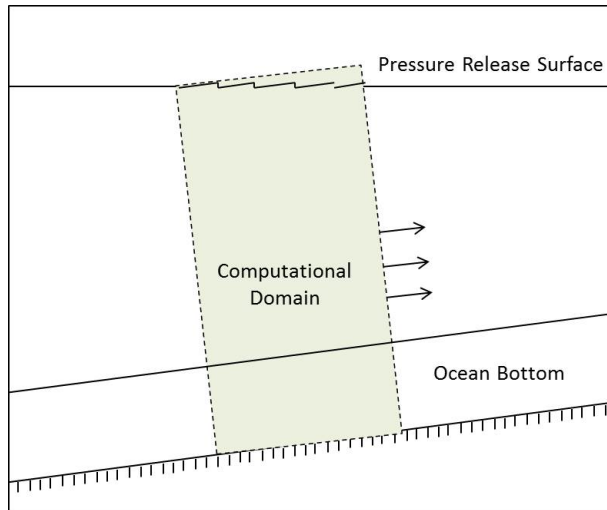


Figure 2.3: In the rotated coordinate approach, the grid is rotated to match the ocean bottom and the surface is approximated as stair steps.

approach has not yet been applied to the NPE model, and this is the major thrust of this thesis.

This method has been shown to provide greater accuracy than the terrain-following coordinates method for underwater environments, specifically when penetrable ocean bottoms are present (Outing *et al.*, 2006). The main drawbacks to this method are that it can be difficult to treat variable sloped ocean bottoms and the seafloor must be approximated as a series of sloped lines, that is, the bottom cannot have curvature.

2.4 Thesis Introduction

The main goal of this thesis is to enhance the current CSM NPE code implemented by Larissa Taylor (Taylor, 2011) to be able to treat more realistic ocean environments. The following chapter will introduce interface and boundary conditions necessary for modeling penetrable ocean bottoms, including multiple sediment layers. An artificial absorbing layer will also be employed to prevent reflections from the computational grid now that the ocean-bottom interface is no longer rigid. Results from this enhanced model will be compared to those of a Fourier-transformed parabolic equation solution to benchmark the model for the

linear case. The NPE model will then be adapted to treat constant sloped ocean bottoms using the rotated coordinates approach for which solutions will be compared to a Fourier-transformed PE model. All benchmark comparison are done for a linear case because the PE model is one of the few long-range models able to treat range-dependence. However, the nonlinear aspect of the NPE has previously been shown to be accurate (McDonald & Kuperman, 1987). Finally, future work will be discussed including possible routes of research for treating variable slopes and elastic sediment layers.

CHAPTER 3

IMPLEMENTATION

In order to simulate blast wave propagation in realistic ocean environments, a number of enhancements have been made to the standard NPE model. This chapter provides details on numerical procedures used to account for acoustic penetration into ocean sediment layering as well as the boundary conditions and initial conditions required for simulations. Brief descriptions of programming implementations and performance consideration are provided in this chapter. Results of this model are compared to those of the parabolic equation solution for a range-independent case to benchmark the improved model.

3.1 Boundary Conditions

The standard NPE is written in terms of a dimensionless density perturbation, R , which is not conducive for applying boundary conditions. It is possible to reformulate the NPE in terms of a dimensionless pressure variable $Q \equiv p'/\rho_0 c_0^2$ (Ambrosiano *et al.*, 1990):

$$D_t Q = -\partial_x \left[c_1 Q + \frac{\beta c_0}{2} Q^2 \right] - \frac{c_0}{2} \int_{x_f}^x (\partial_y^2 + \partial_z^2) Q \, dx. \quad (3.1)$$

This form of the equation is no different than the original formulation. The reason for this is because $Q = R + \mathcal{O}(\epsilon^2)$, and substitution of this term into (2.25) leads to an error of size $\mathcal{O}(\epsilon^3)$ (terms of this size were disregarded in the derivation). Use of this new dimensionless pressure variable, Q , allows for the treatment of boundary and interface conditions. The boundary conditions enforced during simulation are as follows. The ocean surface is considered a pressure-release boundary, $Q = 0$ at $z = 0$, and the medium ahead of the computational domain is considered quiescent, $Q = 0$ at $x = x_N$. These conditions presume that the acoustic pulse is propagating in a calm ocean devoid of any surface waves.

In basic ocean environment models, the seafloor is often considered a rigid boundary to simplify calculations. In such a case, the Neumann boundary condition $\frac{\partial Q}{\partial z} = 0$ is applied

at the bottom boundary. To implement this boundary condition with the finite-difference scheme presented in Section 2.2.2, the bottom boundary is placed just below the last row of the computational domain, between row M and artificial row $M + 1$. The grid points on row $M + 1$ represent non-physical values since they lay outside of the domain. Now, the Neumann boundary condition is applied to grid points on either side of the bottom boundary as shown in Figure 3.1. The discrete finite-difference boundary condition is given by

$$\frac{Q_{M+1} - Q_M}{\Delta z} = 0, \quad (3.2)$$

where Q_{M+1} represents a non-physical value. This equation is solved, trivially, giving $Q_{M+1} = Q_M$.

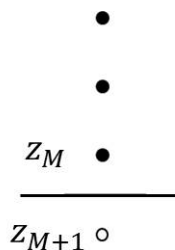


Figure 3.1: Grid used for applying bottom boundary condition. The empty circle represents the non-physical grid point.

This equality is applied to equation (2.43) with $j = M$ which in turn leads to a modified \mathbf{A} matrix. The M^{th} row of \mathbf{A} becomes

$$\mathbf{A}_M = \begin{bmatrix} 0 & \cdots & 1 & -1 \end{bmatrix}, \quad (3.3)$$

and the rest of the finite-difference scheme outlined in Section 2.2.2 remains the same.

3.2 Penetrable Ocean Bottom

Although the rigid seafloor model is computational efficient, it does not adequately describe realistic ocean environments where sound is able to travel into and out of ocean bottom sediment layers. In this research effort, sediment layers in the ocean bottom are treated as fluids with densities and sound speeds that differ from those of the water column. This leads to the approximation that shear wave propagation in the seafloor is negligible (nonelastic material): this is true for unconsolidated soils which are often found in the seabed, but this is not necessarily accurate for dense bottom media.

Two interface conditions that need to be satisfied at the fluid-fluid interface at the ocean bottom are continuity of pressure and continuity of normal particle speed. Mathematically, these conditions are given by

$$Q_u = Q_l, \tag{3.4a}$$

$$\frac{1}{\rho_u} \frac{\partial Q_u}{\partial z} = \frac{1}{\rho_l} \frac{\partial Q_l}{\partial z}, \tag{3.4b}$$

where the subscripts u and l denote the upper and lower fluids. To implement these conditions with the finite-difference scheme, the interface is placed between rows j and $j + 1$ of the the computational domain. The interface conditions are applied to grid points on either side of the interface as shown in Figure 3.2. This results in discrete formulae

$$\frac{Q_{u_j} + Q_{u_{j+1}}}{2} = \frac{Q_{l_j} + Q_{l_{j+1}}}{2} \tag{3.5a}$$

$$\text{and} \quad \frac{1}{\rho_u} \frac{Q_{u_{j+1}} - Q_{u_j}}{\Delta z} = \frac{1}{\rho_l} \frac{Q_{l_{j+1}} - Q_{l_j}}{\Delta z}, \tag{3.5b}$$

where $Q_{u_{j+1}}$ and Q_{l_j} are termed non-physical values. The formulas in (3.5) correspond to a set of two equations for two unknowns, the non-physical values. Solving for the unknowns gives

$$Q_{u_{j+1}} = \frac{\rho_l - \rho_u}{\rho_l + \rho_u} Q_{u_j} + \frac{2\rho_u}{\rho_l + \rho_u} Q_{l_{j+1}} \tag{3.6a}$$

$$\text{and} \quad Q_{l_j} = \frac{2\rho_l}{\rho_l + \rho_u} Q_{u_j} + \frac{\rho_u - \rho_l}{\rho_l + \rho_u} Q_{l_{j+1}}. \tag{3.6b}$$

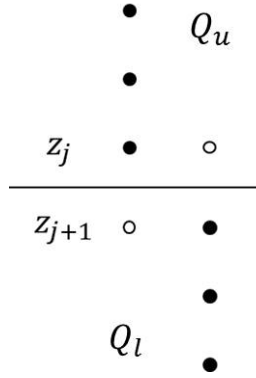


Figure 3.2: Grid used for applying interface conditions. The empty circles represent non-physical grid points.

Applying these terms to Eqn. (2.43) for rows j and $j + 1$ leads to a modified \mathbf{A} matrix. The rows associated with the fluid-fluid interface become

$$\mathbf{A}_j = \left[0 \quad \dots \quad 1 \quad - \left(2 - \frac{\rho_l - \rho_u}{\rho_l + \rho_u} \right) \quad \frac{2\rho_u}{\rho_l + \rho_u} \quad \dots \quad 0 \right] \quad (3.7)$$

and

$$\mathbf{A}_{j+1} = \left[0 \quad \dots \quad \frac{2\rho_l}{\rho_l + \rho_u} \quad - \left(2 - \frac{\rho_u - \rho_l}{\rho_l + \rho_u} \right) \quad 1 \quad \dots \quad 0 \right]. \quad (3.8)$$

Again, the finite-difference scheme is executed in a similar manner as described in Section 2.2.2, except now there are multiple σ values corresponding to the various fluid layers.

3.2.1 Artificial Absorbing Layer

The penetrable ocean bottom allows for the possibility of energy to propagate to the base of the computational domain. Therefore, an artificial absorbing layer is needed to damp out any energy near the bottom of the computational grid to prevent any unrealistic reflections. Most frequency-domain solutions use a complex wavespeed to attenuate energy in the seafloor

– an effective model for dispersive sediments. The complex wave speed can be adjusted at the bottom of the domain to damp out all energy. The NPE is not complex-valued and is set in the time-domain, so this method is not trivial and therefore is not considered here (Castor *et al.*, 2004). The absorbing layer used is numerically implemented by adding a $-R/\tau$ term to Eqn. (2.25) (or, equivalently, adding a $-Q/\tau$ term to Eqn. (3.1)). The numerical damping coefficient τ varies as a function of depth and is defined solely in the artificial absorbing region, $z_{damp} \leq z \leq z_M$, where z_{damp} is the depth at which the artificial absorbing layer begins. After the damping term is added to the NPE, the finite-difference scheme given in Eqn. (2.44) now becomes

$$(\mathbf{I} - \sigma \mathbf{A}) \vec{R}_i^{n+1} = \left(\mathbf{I} + \sigma \mathbf{A} - \frac{\Delta t}{\tau(z)} \mathbf{I} \right) \vec{R}_i^n + 2\sigma \sum_{k=i+1}^{N-1} (\mathbf{A}(\vec{R}_k^n + \vec{R}_k^{n+1})). \quad (3.9)$$

From Eqn. (3.9) it is inferred that the $\Delta t/\tau$ term should disappear near the beginning of the artificial absorbing layer, $z = z_{damp}$ and it should go to unity near the bottom of the computational domain at $z = z_M$. This ensures no damping at the beginning of the absorbing layer and complete damping at the bottom of the computational grid. The numerical damping coefficient $\tau(z)$ can be any continuous, smooth function that satisfies $\Delta t/\tau = 0$ at $z = z_{damp}$ and $\Delta t/\tau = 1$ at $z = z_M$. The function used throughout this work is of the form

$$\tau(z) = \frac{C}{(z - z_{damp} + 1)^\alpha}, \quad (3.10)$$

where C is an arbitrary large number and the exponent α is a value that satisfies

$$(z_M - z_{damp} + 1)^\alpha = \frac{C}{\Delta t}. \quad (3.11)$$

Simulations of multiple environments have shown that this function provides adequate damping near the bottom of the domain while maintaining a smooth transition from the non-damped region to the artificial absorbing layer.

3.3 Computer Programs

The updated NPE model has been implemented with two main computer programs. The first is a standard serial code written in MATLAB which takes advantage of sparse matrix solvers and the vast library of mathematical functions provided by MATLAB. The second is a parallel code written, as part of this work, in Fortran 90 that makes use of the Message Passing Interface (MPI) for parallelizing computations when used on multi-core machines. Both programs are examined, followed by a discussion of the computational efficiency gained by the parallel code.

3.3.1 Serial Code

Initially, a serial program was written in MATLAB that was capable of performing NPE simulations. It makes use of a built-in numerical algorithm for solving the system of equations given by (2.44). This code was executed for a sample simulation in order to determine the serial run time, T_1 , which is used to determine parallel performance, discussed in Section 3.3.3. The following paragraphs give a brief outline of how the serial code works so that the parallel code may be easily understood. Refer to Appendix A for detailed explanation of both the serial and parallel programs.

The main program first calls the input data reading procedure to read in information pertaining to the initial condition and data saving/printing. Next, the program initializes the overpressure field matrix, Q , as well as the grid parameter arrays x and z . Figure 3.3 provides an illustration of how the computational domain is organized. Then a subroutine generates the starting field (initial condition) given user inputs, after which the program enters into time stepping. For every time step the following operations are performed. Every column of the Q matrix is passed to the FCT algorithm to calculate a corresponding column of the intermediate field, Q_{int} . The wavefront boundary condition is enforced by setting the bottom row of Q and Q_{int} to zero. Next, the code loops through the rows, starting from the last row and ending at the first row, summing up the Q and Q_{int} values from the current row to

the last row for the current and previous time steps. The matrix multiplications specified in Eqn. (2.44) are executed, and the final system of equations are solved using a least squares approach. The upper boundary condition is enforced by setting first column of Q to zero. Finally, the appropriate data is saved and the program continues to the next time step until the total simulation range has been covered.

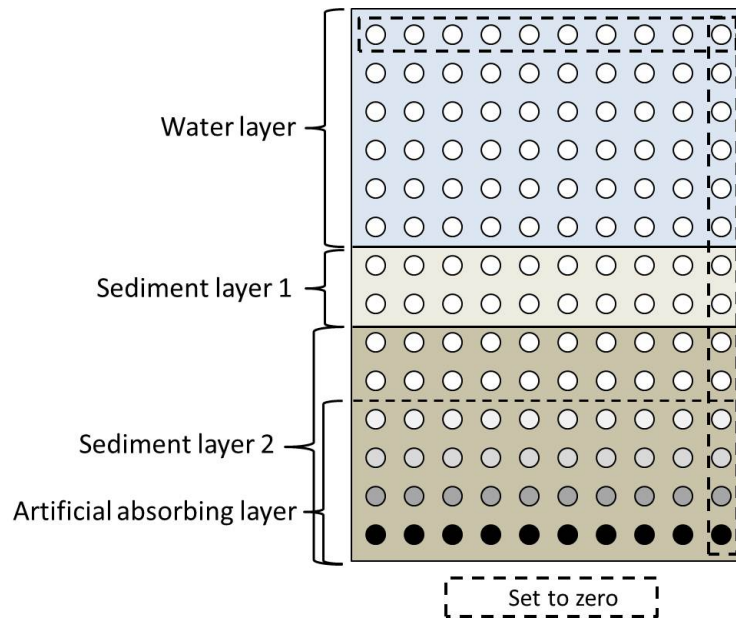


Figure 3.3: A schematic representing a typical computational domain. The darker circles represent increased dampening.

3.3.2 Parallel Code

The parallel version of the code is very similar to the serial version. In the main program, the MPI environment is initialized and the columns of the Q matrix are divided among the different processors. Every processor then reads in the data and initializes its set of columns, and then they follow the same procedures as in the serial code up until solving the system. Since Fortran does not have a built-in algorithm for solving a system of equations, a tridiagonal system solver was written. The solver used was the Tridiagonal Matrix Algorithm (TDMA), also known as the Thomas algorithm, because it is computationally efficient at

handling tridiagonal matrix systems (Conte, 1972). This algorithm is explained in more detail in Appendix A. The TDMA requires that certain arrays be built prior to time stepping, so a subroutine is used to create these required arrays on each core. Within the time step loop, each core performs the FCT algorithm in parallel since it uses columns of the Q and not rows. The remainder of the program is essentially the same except that MPI send and receive commands are required for both the matrix-vector multiplications and the TDMA solver.

In the parallel program, both the tridiagonal matrix and the vectors being multiplied are subdivided among different processors. For a machine with a total of P processing cores, a matrix with M rows is divided such that processor 0 has rows 1 through M/P , processor 1 has rows $M/P + 1$ through $2M/P$ and so on. The last processor, processor $P - 1$, has rows $(P - 1)M/P + 1$ through M . Each processor needs values from neighboring processors in order to complete their work with the matrix-vector multiplication and the TDMA solver. A “neighbor” communication method was used where array values nearest to the neighbor processor are sent, as seen in Figure 3.4. This neighbor communication is much more efficient than the “alltoall” type communication needed when multiplying a full matrix.

$$\begin{array}{c}
 \left[\begin{array}{ccc}
 b_1 & c_1 & \\
 a_2 & b_2 & c_2 \\
 & a_3 & b_3 & c_3 \\
 \hline
 & & a_4 & b_4 & c_4 \\
 & & & a_5 & b_5 & c_5 \\
 & & & & a_6 & b_6 & c_6 \\
 \hline
 & & & & & a_7 & b_7 & c_7 \\
 & & & & & & a_8 & b_8 & c_8 \\
 & & & & & & & a_9 & b_9
 \end{array} \right]
 \begin{array}{c}
 \left[\begin{array}{c}
 x_1 \\
 x_2 \\
 x_3 \\
 x_4 \\
 x_5 \\
 x_6 \\
 x_7 \\
 x_8 \\
 x_9
 \end{array} \right]
 =
 \begin{array}{c}
 \left[\begin{array}{c}
 d_1 \\
 d_2 \\
 d_3 \\
 d_4 \\
 d_5 \\
 d_6 \\
 d_7 \\
 d_8 \\
 d_9
 \end{array} \right]
 \end{array}
 \end{array}
 \left. \begin{array}{l}
 \vphantom{\left[\begin{array}{c}
 x_1 \\
 x_2 \\
 x_3 \\
 x_4 \\
 x_5 \\
 x_6 \\
 x_7 \\
 x_8 \\
 x_9
 \end{array} \right]} \vphantom{=} \vphantom{\left[\begin{array}{c}
 d_1 \\
 d_2 \\
 d_3 \\
 d_4 \\
 d_5 \\
 d_6 \\
 d_7 \\
 d_8 \\
 d_9
 \end{array} \right]} \vphantom{\left. \begin{array}{l}
 \\
 \\
 \\
 \\
 \\
 \\
 \\
 \\
 \\
 \end{array} \right\}}
 \begin{array}{l}
 \left. \begin{array}{l}
 \\
 \\
 \\
 \\
 \\
 \\
 \\
 \\
 \\
 \end{array} \right\} p0 \\
 \left. \begin{array}{l}
 \\
 \\
 \\
 \\
 \\
 \\
 \\
 \\
 \\
 \end{array} \right\} p1 \\
 \left. \begin{array}{l}
 \\
 \\
 \\
 \\
 \\
 \\
 \\
 \\
 \\
 \end{array} \right\} p2
 \end{array}
 \right.
 \end{array}$$

Example Communication

$p0$: send $x(3)$ to $p1$ and receive $x(4)$ from $p1$

$p1$: send $x(4)$ to $p0$ and send $x(6)$ to $p2$. Receive $x(3)$ from $p1$ and $x(7)$ from $p2$

$p2$: send $x(7)$ to $p1$ and receive $x(6)$ from $p1$

Figure 3.4: An example of the neighbor communication method used in matrix-vector multiplication.

3.3.3 Parallel Performance

In order to test the performance of the parallel code, it was ran on the Colorado School of Mines MIO machine using a number of different processing cores. The simulation was for the standard Pekeris waveguide problem with nonlinear effects included. The total simulation range was 1 km, the depth of the water was 600 m, the depth of the single sediment layer was an additional 400 m, and the horizontal extent of the computational domain was 1000 m. The spatial discretization was 1 m in each of the x and z directions leading to a 1000 x 1000 computational grid.

The table and figures below give parallel performance results for up to a total of thirty-two processors. Running with only one core (serial program) takes nearly two hours to complete. This is roughly double the amount of time that it takes for the serial MATLAB version of the code. However, as the number of cores increases the computation time decreases for up to twelve cores. The computation time jumps up dramatically with fourteen cores; this may be due to having to use an additional node on the MIO cluster (each node contains twelve processors). The addition of a new node leads to greater communication complexity and thus more overhead and less efficiency.

The speedup appears to be linear and nearly perfect for the first twelve cores, but it drops off quickly at fourteen cores. Figure 3.7 shows that with four cores the efficiency actually goes above 1.0. Theoretically this is not possible, but it may just be a computational anomaly since the efficiency is only a fraction greater than 1.0.

Table 3.1: Parallel Performance

| Number of cores | Computation Time | Speedup | Efficiency | Serial Fraction |
|-----------------|------------------|-------------|-------------|-----------------|
| 1 | 6845.422135 | 1 | 1 | 1 |
| 2 | 3820.973501 | 1.791538762 | 0.895769381 | 0.116358765 |
| 4 | 1515.384973 | 4.517282576 | 1.129320644 | -0.038170631 |
| 8 | 933.2012749 | 7.335418756 | 0.916927345 | 0.012942707 |
| 12 | 853.3969679 | 8.021380896 | 0.668448408 | 0.04509107 |
| 14 | 1628.241685 | 4.204180618 | 0.300298616 | 0.179232206 |
| 16 | 4612.890402 | 1.483976756 | 0.092748547 | 0.652122669 |
| 20 | 1639.627568 | 4.174985996 | 0.2087493 | 0.199496591 |
| 24 | 1644.380351 | 4.162918956 | 0.173454957 | 0.207181978 |
| 32 | 2355.0132 | 2.906744699 | 0.090835772 | 0.322867057 |

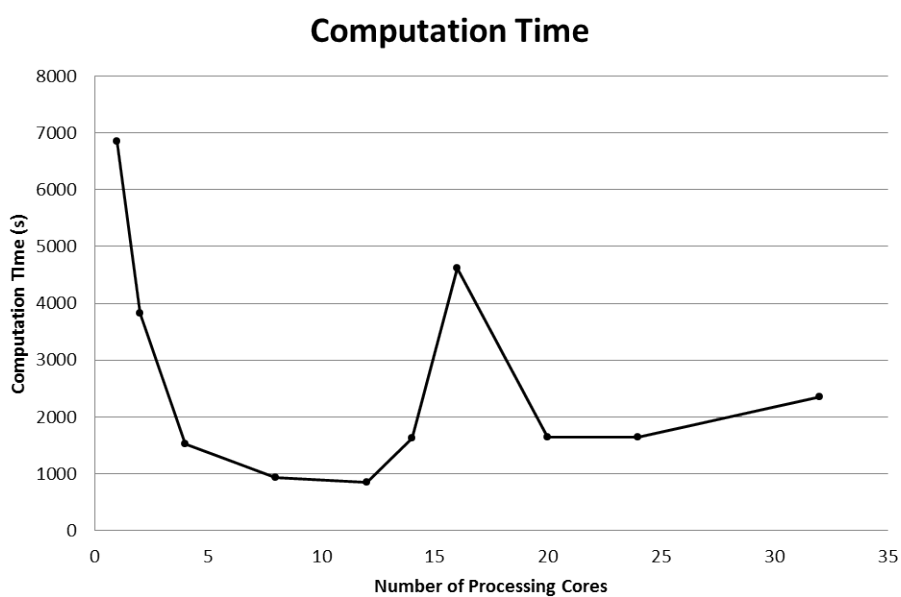


Figure 3.5: Total computation time for various numbers of processors.

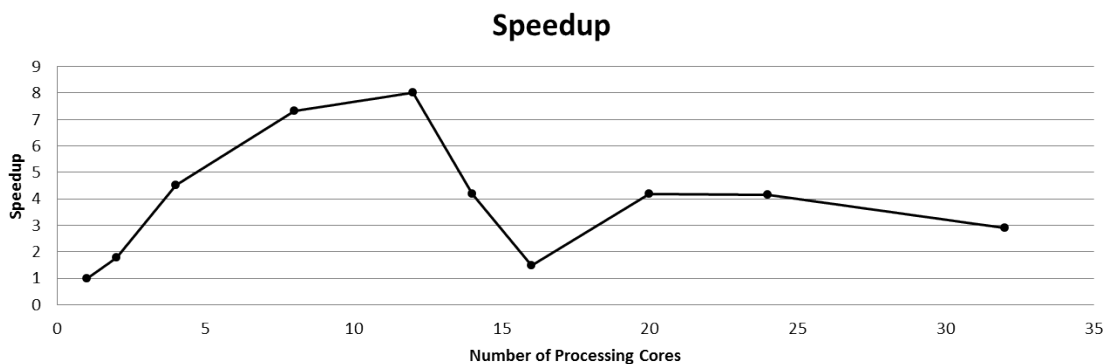


Figure 3.6: Parallel speedup for various numbers of processors.

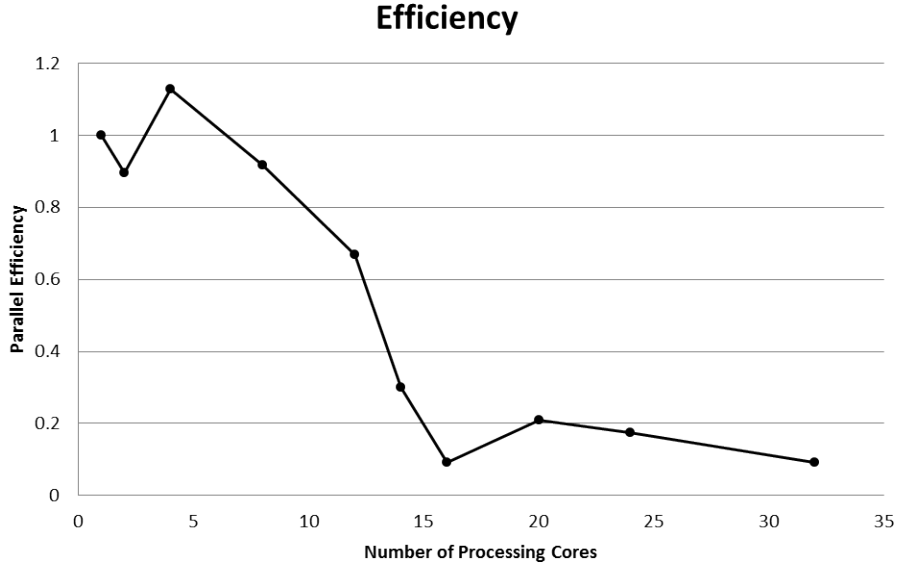


Figure 3.7: Parallel efficiency for various numbers of processors.

Results show that the parallel NPE program is fairly efficient when using a small number of cores. When using more than twelve processing cores, the program starts to lag considerably due to the large amount of communication. The main reason for this is because the computation to communication ratio is low for the matrix-vector multiplication algorithm and the TDMA solver. Even so, dividing up the pressure field matrix Q decreases the memory use, so much larger grids can be used which are often necessary for realistic simulations.

3.4 Solution Benchmarking

The updated NPE model was benchmarked against a parabolic equation solution for a linear, range-independent case. The parabolic equation model used was the Range-dependent Acoustic Model (RAM) developed at the Naval Research Laboratory by Michael Collins (Collins, n.d.). It is a linear, frequency-domain solution known for producing accurate results for both range-independent and dependent environments. Because RAM is a frequency-domain model, a Fourier transform is required to transfer to the time-domain, using a discrete

Fourier synthesis.

The environment used in the comparison is a Pekeris waveguide with a water column depth of 100 m and an additional 100 m of sediment depth as seen in Figure 3.8 below. The water column has a density of 1000 kg/m^3 and a sound speed of 1500 m/s whereas the sediment has a density of 1500 kg/m^3 and a sound speed of 1650 m/s . The source is located at $z_s = 50 \text{ m}$ and emits a pulse described by

$$Q(t) = \begin{cases} \sin(\omega_c t) - \frac{1}{2} \sin(2\omega_c t) & \text{for } 0 < t < 1/f_c \\ 0 & \text{else,} \end{cases} \quad (3.12)$$

where the center frequency f_c has been chosen to be 100 Hz . This problem was analyzed with the serial code using a computational domain of height 200 m and width 90 m . The x-direction grid spacing was 0.225 m , the z-direction grid spacing was 1 m , and the time step was 0.0005 s . Only the linear case was considered and as such the coefficient of nonlinearity β was set to zero.

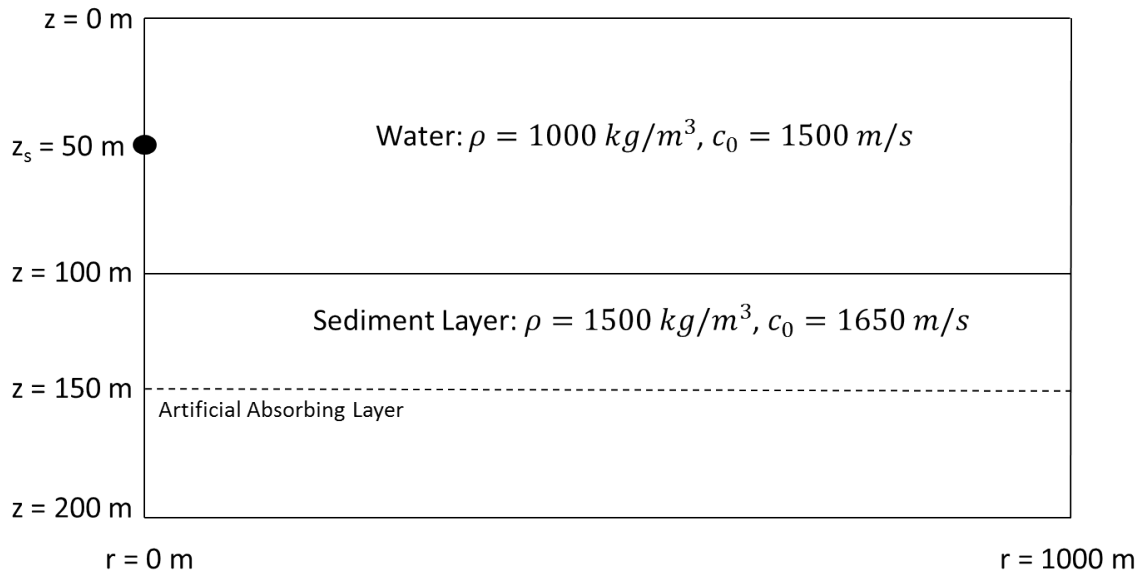


Figure 3.8: Environment used in range-independent benchmark comparison.

The Fourier-transformed PE solution was used to generate the initial waveform for the NPE code, and then the simulation marched the computational domain out to a final range of 1000 m. Figure 3.9 shows the pulse at various ranges in the waveguide. The pressure data was saved at four different receiver locations: 500 m and 1000 m in range (both at 20 m and 50 m in depth). The time series data at these receiver locations were then compared to those of the Fourier-transformed RAM solution. These comparisons are detailed in the plots, Figure 3.10 and Figure 3.11.

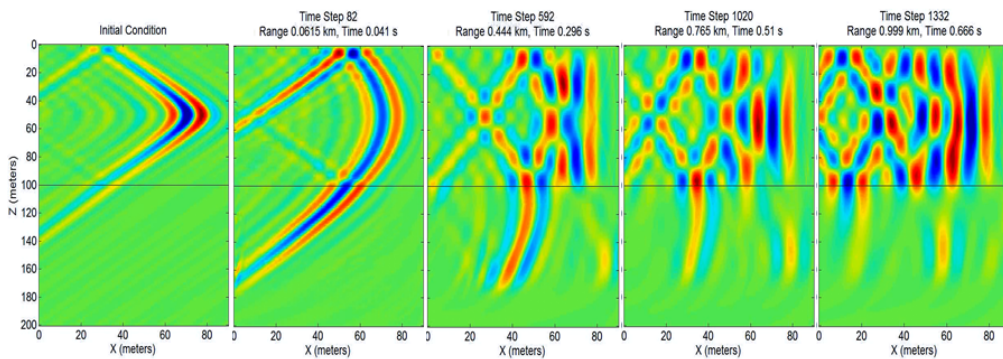


Figure 3.9: Propagation of the pulse within the Pekeris waveguide.

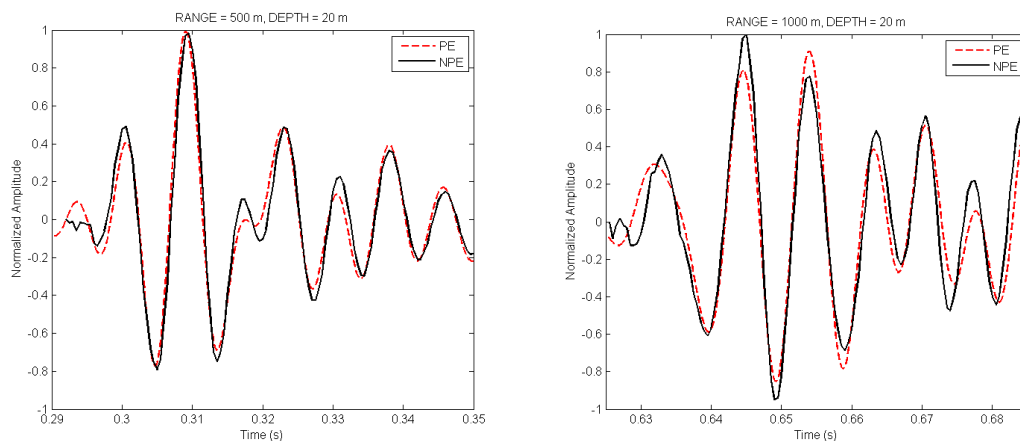


Figure 3.10: Results of range-independent environment at receiver depth of 20 m.

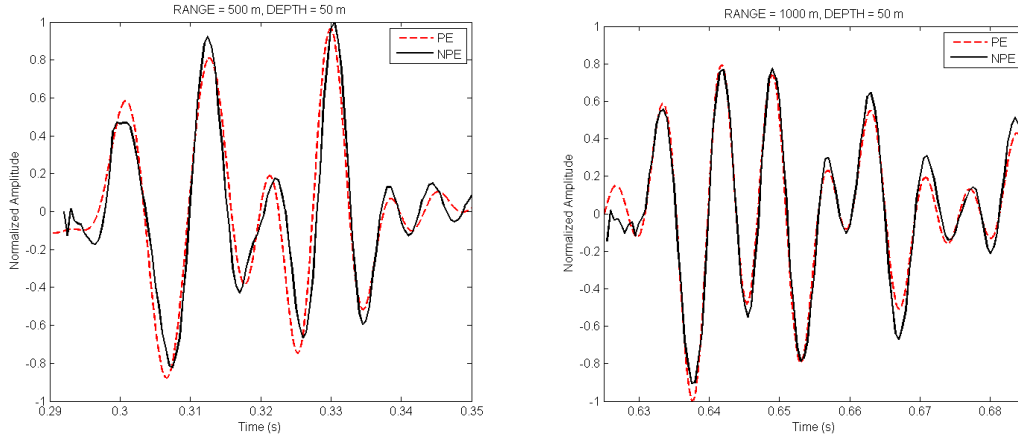


Figure 3.11: Results of range-independent environment at receiver depth of 50 m.

The plots show that the PE and NPE solutions agree very well. This indicates that the penetrable bottom NPE model is correct and can be compared to the Fourier-transformed PE solution for the linear case. As a matter of comparison, the NPE model was also run with nonlinear effects included by setting $\beta = 3.5$. The other simulation parameters remained the same except the computation window was widened to 120 m to allow for speedup of the wavefront due to nonlinear steepening. Figure 3.12 gives snapshots of the pulse as it moves down the waveguide. The pulse is seen to evolve in a manner considerably different from the linear case. The nonlinear results are compared to the Fourier-transformed PE solution in Figure 3.13 and Figure 3.14. These plots indicate that including nonlinear effects has a drastic effect on the overall shape of the waveform. The pulse tends to steepen creating weak shock fronts, and the front of the pulse has an earlier arrival than in the linear model, signifying a propagation velocity slightly greater than the sound speed of water. The nonlinear effects are directly proportional to the amplitude of the wave, so one would expect different results if the initial waveform was scaled at all.

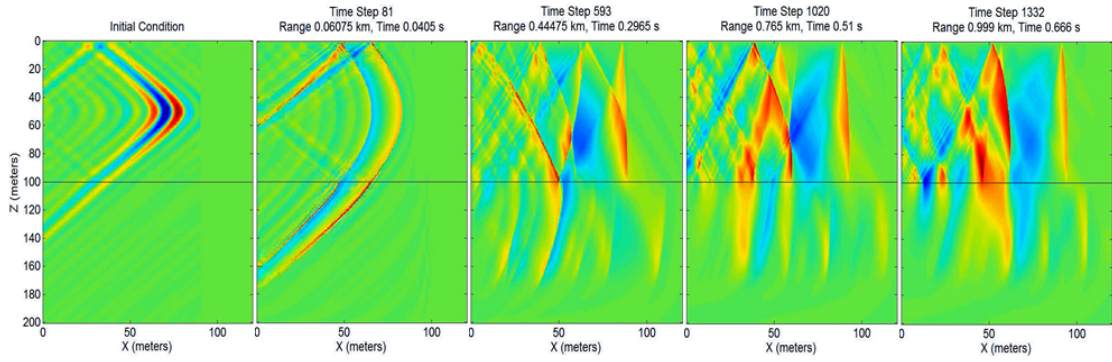


Figure 3.12: Propagation of the pulse within the Pekeris waveguide. Nonlinear effects are included.

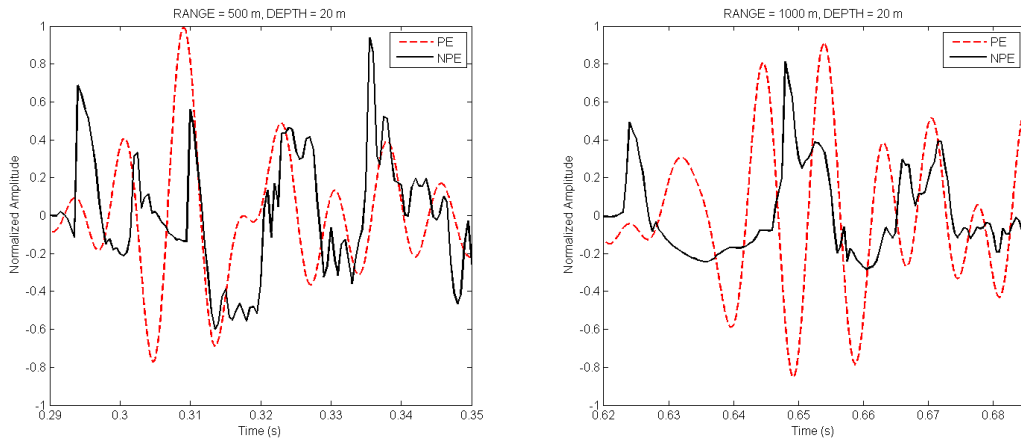


Figure 3.13: Results of range-independent environment at receiver depth of 20 m. Nonlinear effects are included in the NPE model.

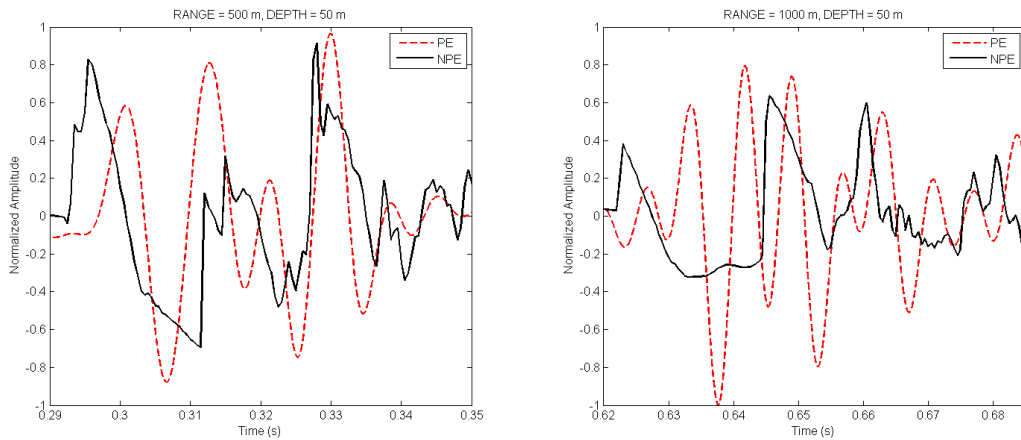


Figure 3.14: Results of range-independent environment at receiver depth of 50 m. Nonlinear effects are included in the NPE model.

CHAPTER 4
SLOPED OCEAN BATHYMETRY

Most ocean environments are not made up of entirely flat seafloors. In fact, while many ocean bottoms appear to be flat, they often are sloped to a small degree, and in shallow oceans, even a small slope can have a significant effect on acoustic pulse propagation over long ranges. A critical aspect missing from the NPE model has been treatment of non-flat ocean bathymetries. These are addressed in this chapter by rotating the coordinate system in the standard NPE model to treat range-dependent environments characterized by constant sloped ocean bottoms. Results of this rotated solution are then compared to those of Fourier synthesized parabolic equation solutions for select range-dependent environments.

4.1 Approach

A rotated coordinate approach was used to treat range-dependent environments characterized by constant sloped ocean bottoms. The topography of the seafloor is assumed to be a constant sloped surface with a slope angle θ positive for upslope propagation and negative for downslope propagation. The standard domain defined in the xz -plane is rotated by θ so that the x -axis runs parallel with the sloped ocean bottom. This is achieved with the coordinate transformation given by

$$\begin{Bmatrix} \tilde{x} \\ \tilde{z} \end{Bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{Bmatrix} x \\ z \end{Bmatrix}, \quad (4.1)$$

where \tilde{x} and \tilde{z} are the tangential and normal coordinates relative to the ocean bottom, respectively. This coordinate transformation is applied to the derivation of the standard NPE in order to derive the new rotated solution. In this approach, the shorter heuristic derivation used by McDonald and Kuperman in (McDonald & Kuperman, 1987) is used rather than the formal derivation for the sake of brevity.

For an inviscid fluid, the Euler equations of continuity of mass and momentum combine to give

$$\frac{\partial^2 \rho}{\partial t^2} = \nabla^2 p + \partial_i \partial_j (\rho v_i v_j), \quad (4.2)$$

where $i = 1, 2, 3$ and the repeated subscripts imply the Einstein summation convention. In the non-rotated (x, z) coordinate system, the Laplacian operator is defined as

$$\nabla^2 f = \partial_x^2 f + \partial_y^2 f + \partial_z^2 f. \quad (4.3)$$

To find the Laplacian in rotated coordinates, $\tilde{\nabla}^2$, the chain rule is applied to the spatial derivatives:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial \tilde{x}} \frac{\partial \tilde{x}}{\partial x} + \frac{\partial f}{\partial \tilde{z}} \frac{\partial \tilde{z}}{\partial x} = \frac{\partial f}{\partial \tilde{x}} \cos \theta - \frac{\partial f}{\partial \tilde{z}} \sin \theta \quad (4.4a)$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial \tilde{y}} \quad (4.4b)$$

$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial \tilde{x}} \frac{\partial \tilde{x}}{\partial z} + \frac{\partial f}{\partial \tilde{z}} \frac{\partial \tilde{z}}{\partial z} = \frac{\partial f}{\partial \tilde{x}} \sin \theta + \frac{\partial f}{\partial \tilde{z}} \cos \theta. \quad (4.4c)$$

Second derivatives are taken and added up to give the rotated Laplacian

$$\begin{aligned} \tilde{\nabla}^2 f &= \frac{\partial^2 f}{\partial \tilde{x}^2} \cos^2 \theta - 2 \frac{\partial^2 f}{\partial \tilde{x} \partial \tilde{z}} \cos \theta \sin \theta + \frac{\partial^2 f}{\partial \tilde{z}^2} \sin^2 \theta + \frac{\partial^2 f}{\partial \tilde{y}^2} + \\ &\frac{\partial^2 f}{\partial \tilde{x}^2} \sin^2 \theta + 2 \frac{\partial^2 f}{\partial \tilde{x} \partial \tilde{z}} \cos \theta \sin \theta + \frac{\partial^2 f}{\partial \tilde{z}^2} \cos^2 \theta, \end{aligned} \quad (4.5)$$

which reduces to

$$\tilde{\nabla}^2 f = \partial_{\tilde{x}}^2 f + \partial_{\tilde{y}}^2 f + \partial_{\tilde{z}}^2 f. \quad (4.6)$$

Therefore, no change in operation has occurred with the coordinate transformation. Now, take $\rho = \rho_0 + \rho'$ and $p = p_0 + p'$, where p_0 and ρ_0 are assumed constant background values, and p' and ρ' are small perturbations. Substituting for p' a second-order expansion in ρ' using the adiabatic equation of state assumed from linear acoustic theory (F. B. Jensen &

Schmidt, 2005) yields

$$p' = c^2 \rho' + \frac{1}{2} \rho'^2 \frac{\partial^2 p}{\partial \rho^2} + O(\rho'^3). \quad (4.7)$$

For v_i , taking a first-order result for plane waves propagating at a small angle φ to the \tilde{x} direction gives

$$v_i = \frac{c\rho'}{\rho_0} \delta_{i,\tilde{x}} + O(\rho'^2, \rho'\varphi), \quad (4.8)$$

where $\delta_{i,\tilde{x}}$ is the Kronecker delta and $i = 1, 2, 3$ represents \tilde{x}, \tilde{y} , and \tilde{z} . Using the spatial derivatives defined in (4.4) and the same reduction used to go from (4.5) to (4.6), the second term of (4.2) now reduces to

$$\partial_i \partial_j (\rho v_i v_j) = \tilde{\nabla}^2 \left(c^2 \frac{\rho'^2}{\rho_0^2} \right) = \tilde{\nabla}^2 \left(c^2 \frac{\rho'^2}{\rho_0} \right). \quad (4.9)$$

Combining Eqns. (4.2), (4.7), and (4.9) and using the rotated Laplacian gives

$$\frac{\partial^2 \rho'}{\partial t^2} = \tilde{\nabla}^2 c^2 \left(\rho' + \frac{\beta \rho'^2}{\rho_0} \right) + O(\rho'^3, \rho'^2 \varphi^2), \quad (4.10)$$

where

$$\beta = 1 + \frac{\rho}{c} \frac{\partial c}{\partial \rho} \quad (4.11)$$

is the coefficient of nonlinearity and $c^2 = \partial p / \partial \rho$. Defining a dimensionless density perturbation $R = \rho' / \rho_0$ changes (4.10) to

$$\frac{\partial^2 R}{\partial t^2} = \tilde{\nabla}^2 c^2 (R + \beta R^2) + O(R^3, R^2 \varphi^2). \quad (4.12)$$

Now, recast (4.12) in a frame moving in the \tilde{x} direction with a constant speed c_0 . The

material derivative in the moving frame is given by

$$D_t = \frac{\partial}{\partial t} + c_0 \frac{\partial}{\partial \tilde{x}}. \quad (4.13)$$

With the substitution $c = c_0 + c_1$, where c_1 is a small change to the sound speed, (4.12) becomes

$$\left(D_t - c_0 \frac{\partial}{\partial \tilde{x}} \right)^2 R = \left(\frac{\partial^2}{\partial \tilde{x}^2} + \frac{\partial^2}{\partial \tilde{y}^2} + \frac{\partial^2}{\partial \tilde{z}^2} \right) (c_0 + c_1)^2 (R + \beta R^2). \quad (4.14)$$

Each of the parentheses in (4.14) contain a dominant term and one or two smaller terms. In the first parenthesis, the D_t is considered small since the evolution of the pulse within the moving window is much smaller than the speed of the window. In the Laplacian terms, the \tilde{x} -derivative is considered dominant since propagation in the range direction is strongest. For the sound speed, c_1 is considered much smaller compared to c_0 . And finally, the R^2 term is considered to be small since the nonlinearity is weak. Expanding (4.14) and keeping only the lowest order in small terms yields

$$-2c_0 \frac{\partial}{\partial \tilde{x}} D_t R + c_0^2 \frac{\partial^2 R}{\partial \tilde{x}^2} = \frac{\partial^2}{\partial \tilde{x}^2} [c_0^2 (R + \beta R^2) + 2c_0 c_1 R] + \left(\frac{\partial^2}{\partial \tilde{y}^2} + \frac{\partial^2}{\partial \tilde{z}^2} \right) c_0^2 R. \quad (4.15)$$

Rearranging, dividing by $-2c_0$, and integrating with respect to \tilde{x} gives the final form of the governing equation,

$$D_t R = -\partial_{\tilde{x}} \left[c_1 R + \frac{\beta c_0}{2} R^2 \right] - \frac{c_0}{2} \int_{\tilde{x}_f}^{\tilde{x}} (\partial_{\tilde{y}}^2 + \partial_{\tilde{z}}^2) R \, d\tilde{x}, \quad (4.16)$$

which is exactly the same form as the non-rotated NPE. In hindsight, this result could have been identified using knowledge of acoustics and the underlying assumptions of the NPE. For example, a radial pulse emitted within a fluid medium free of boundaries does not tend to any one direction. Hence, the propagation direction can be chosen along any ray originating from

the source. In the case of the NPE, the narrow angle assumption was made in Eqn. (4.8) which assumes that energy propagates at a small angle to the \tilde{x} direction. So, the NPE is valid as long as the pulse is far enough away from the source and the waveguide is sufficiently narrow, thus allowing the majority of energy to travel in the propagation direction.

The governing equation may remain the same for the medium, but the initial condition and the boundary and interface conditions must be transformed to the rotated coordinates. If the initial condition is specified as a plane wave, then it needs to be rotated, but no rotation is needed for the spherical wave case as it is axisymmetric. The coordinate rotation results in a flat ocean-bottom interface which allows for explicit application of the interface conditions. The ocean surface becomes a sloped line, but the Dirichlet boundary condition is still easily imposed there.

4.2 Numerical Implementation

Numerically, the NPE is carried out in the manner described in the two previous chapters, but, now, special consideration is taken when applying the upper boundary condition. The sloping ocean surface is approximated as a series of stair steps, and the pressure release boundary condition $Q = 0$ is applied to the horizontal areas of the steps. As the lengths of the vertical areas of the steps become small, $|Q|$ becomes small in these areas by continuity (Outing, 2004). This means, for small slope angles, conditions at the vertical interfaces of the steps need not be explicitly imposed. However, the NPE model uses a moving computational domain that follows the ocean bottom, so the sloped upper boundary will shift vertically as the domain traverses along the waveguide. The amount the boundary moves vertically is given by $\Delta H = c_0 n \Delta t \tan \theta$, where n is the time step. Figure 4.1 shows how the upper boundary is imposed numerically. For every time step, the equation of the sloping line is calculated and then the routine loops through every grid point and determines if the centroid of the grid point lies above or below the line. If the centroid lies above the line, then the pressure value of that grid point is set equal to zero. There are slight differences

in the numerical scheme for upslope and downslope propagation which are outlined in the paragraphs below.

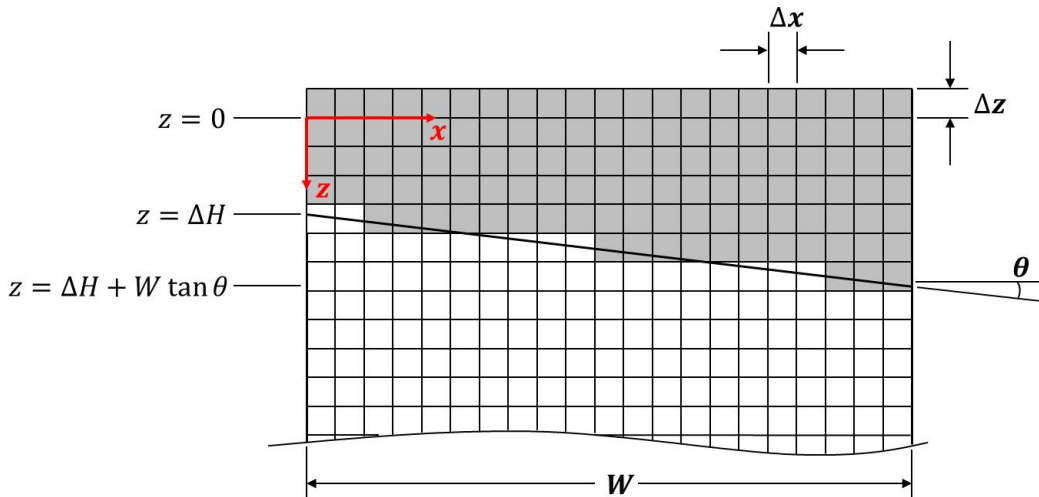


Figure 4.1: Computational approach for treating the sloped upper boundary.

Upslope Propagation In the upslope propagation case, the computational domain is initially fully “submerged” in the ocean. As the domain moves down range, more and more of the domain is displaced above the ocean surface as seen in Figure 4.2. This corresponds to the sloped upper boundary line moving down the grid. The equation of the line is given by

$$z(x) = \Delta H + x \tan \theta, \quad (4.17)$$

where x and z represent the coordinate system of the computational domain and not the global coordinates. One notices that as the upper boundary line is shifted down, energy in the domain is truncated and lost. This truncation error must be taken into account when prescribing the vertical grid spacing.

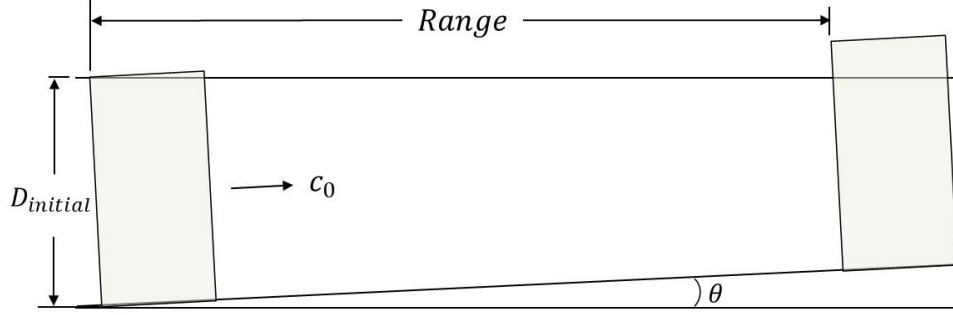


Figure 4.2: Progression of the computational domain during upslope propagation.

Downslope Propagation In the case of downslope propagation, the computational domain initially starts with the top of the grid set a certain height above the sea surface, and as the domain moves down range, it becomes fully submerged as seen in Figure 4.3. The initial height offset is given by $H_0 = r \times \tan \theta + W \sin \theta$, where r is the range. The upper boundary line moves up the grid as the simulation ensues, and it is represented by

$$z(x) = \frac{H_0}{\cos \theta} - \Delta H - x \tan \theta. \quad (4.18)$$

This downslope propagation case is free of the energy truncation error seen in upslope propagation, but the upward shift of the sloped boundary line creates an area on the grid devoid of energy. Added time is spent for the wave to propagate into this void thus producing an error. A smaller vertical grid spacing resolves this issue.

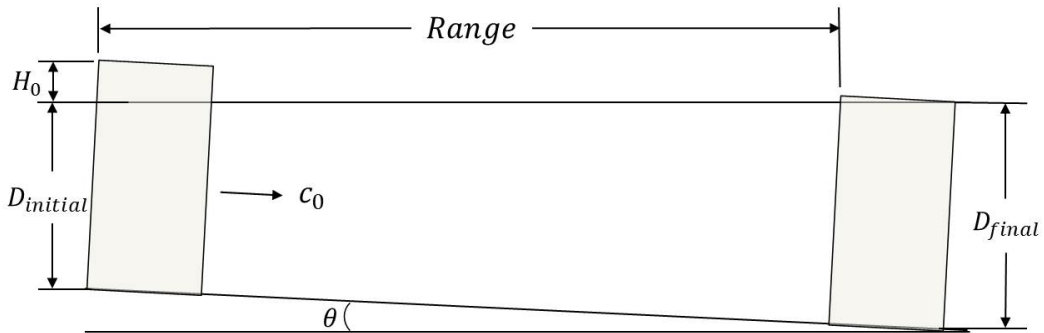


Figure 4.3: Progression of the computational domain during downslope propagation.

4.3 Solution Benchmarking

The rotated NPE model was benchmarked against a parabolic equation solution for two linear, range-dependent cases. The parabolic equation model used is the Range-dependent Acoustic Model (RAM) (a fluid bottom model) in conjunction with discrete Fourier synthesis. The first environment for comparison is an upslope waveguide with slope angle of 2° . Initially, the environment has a water column depth of 100 m and an additional 100 m of sediment depth which decreases linearly down range, as seen in Figure 4.4 below. The water column has a density of 1000 kg/m^3 and a sound speed of 1500 m/s , and the sediment has a density of 1500 kg/m^3 and a sound speed of 1650 m/s . The source is located at $z_s = 50 \text{ m}$ and emits a pulse described by Eqn. (3.12) with a center frequency of 100 Hz . This problem was analyzed with the serial code using a computational domain of height 200 m and width 90 m . The x-direction grid spacing was 0.25 m , the z-direction grid spacing was 1 m , and the time step was 0.0005 s . Only the linear case was considered and as such the coefficient of nonlinearity β was set to zero.

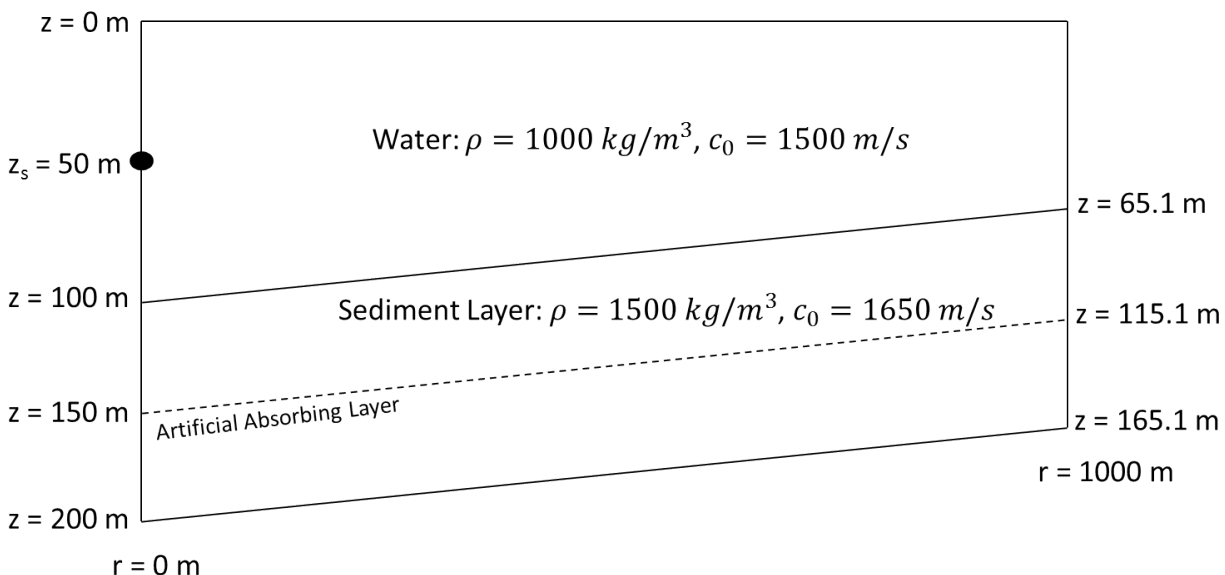


Figure 4.4: Environment used in upslope propagation benchmark comparison.

As was done for the range-independent problem, the starting field used by the NPE was built with the PE solution. The initial waveform generated by the PE solution is essentially a plane wave source, so it was first rotated to match the sloped coordinates. The wave was then marched out in range to a final distance of 1000 m while saving data at ranges of 500 m, and 1000 m, each for depths of 20 m and 50 m. These pressure histories were then compared to the PE solution and the results are shown in Figure 4.5 and Figure 4.6. The results show that the PE and NPE solutions have good agreement with only a small amount of error. The error seems to be mostly in amplitude which may be due in part to the truncation that occurs with the upslope propagation scheme.

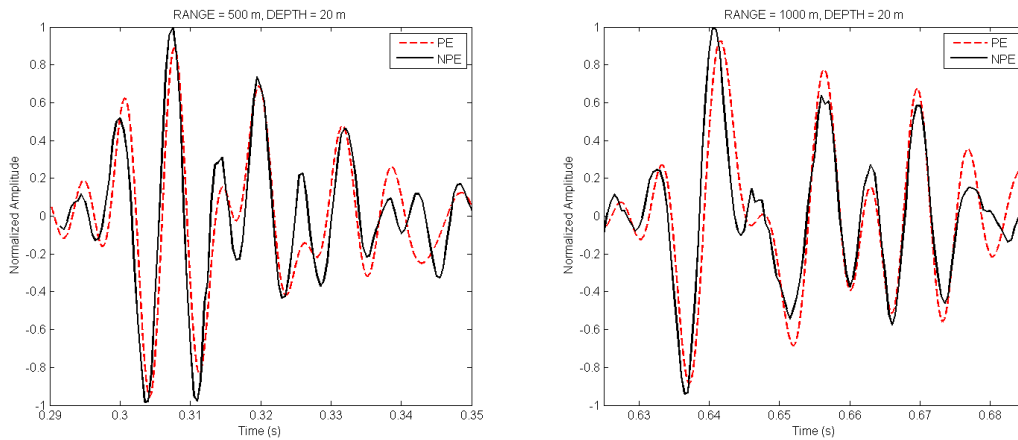


Figure 4.5: Results of upslope environment at receiver depth of 20 m.

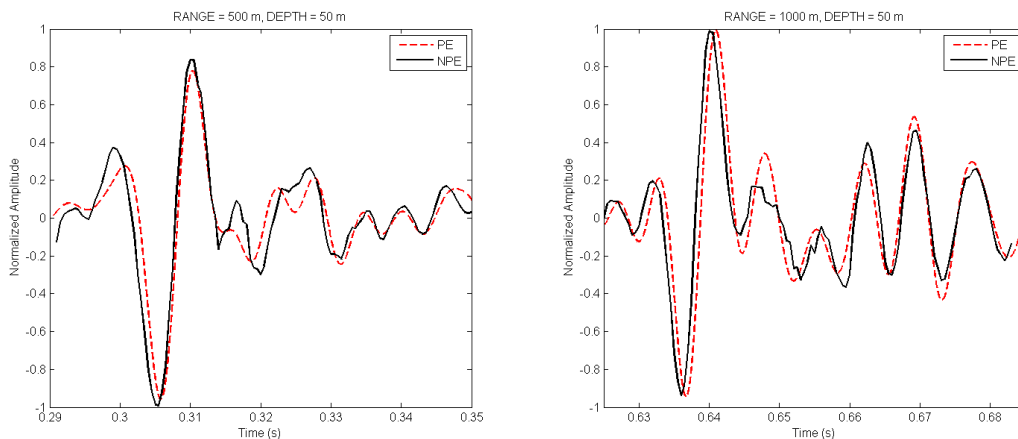


Figure 4.6: Results of upslope environment at receiver depth of 50 m.

The second environment used for comparison is a downslope waveguide with slope angle of -2° . Initially, the environment has a water column depth of 100 m and an additional 100 m of sediment depth which increases linearly down range, as shown in Figure 4.7 below. The geoacoustic parameters (density and sound speed) and the initial condition remain the same as the upslope problem. The computational parameters of the serial code remained the same as well. The NPE-PE comparisons are given below in Figure 4.8 and Figure 4.9. The results show that the PE and NPE solutions agree, but there seems to exist an error associated with the phase of the NPE solution. This error may be due to the method in which the sloped upper boundary is shifted upward during propagation. However, one would expect an error of this type to build up over distance, but results at the 1000 m range appear to be more accurate than those of the 500 m range indicating a different source of error.

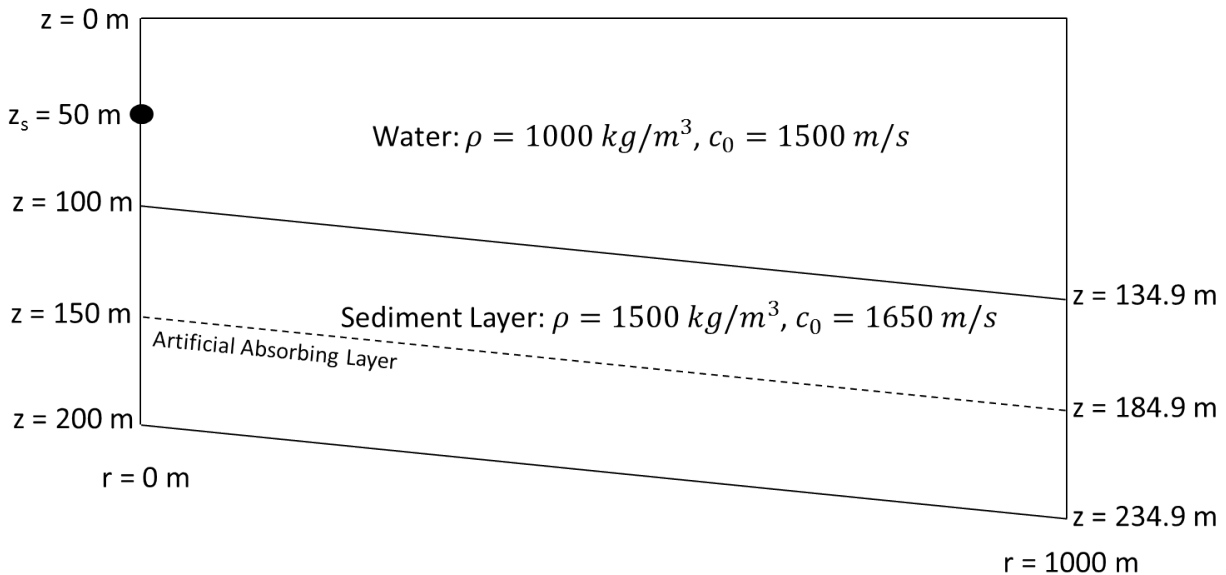


Figure 4.7: Environment used in downslope propagation benchmark comparison.

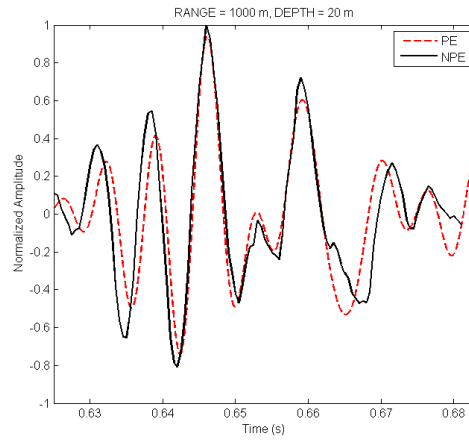
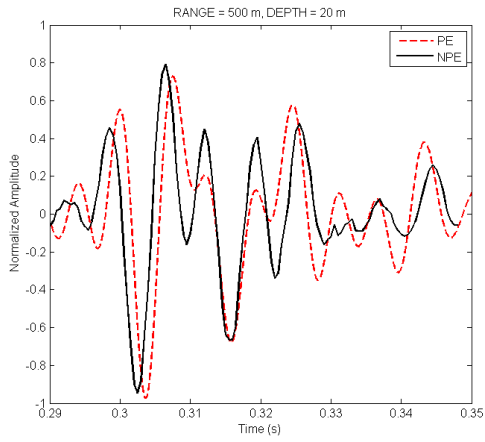


Figure 4.8: Results of downslope environment at receiver depth of 20 m.

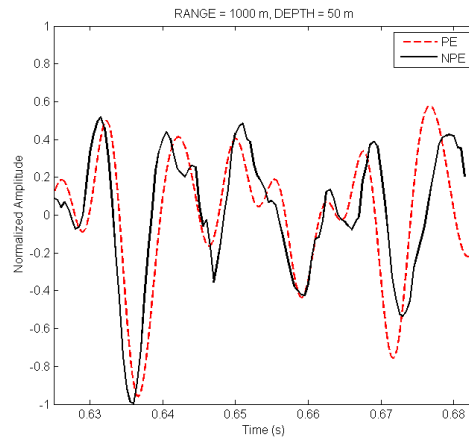
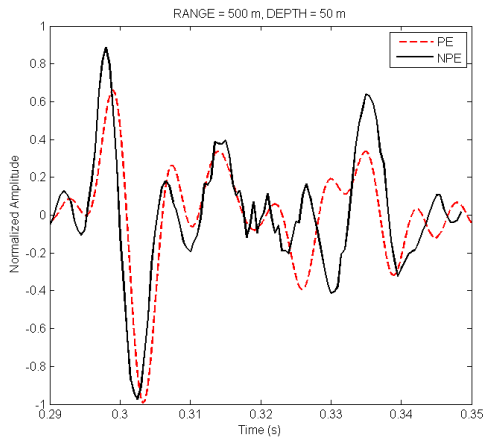


Figure 4.9: Results of downslope environment at receiver depth of 50 m.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

The nonlinear progressive wave equation has been adapted to treat more realistic ocean environments including penetrable ocean bottoms and range-dependence. The updated model was benchmarked against the Fourier-transformed PE solution for the linear case and was shown to be accurate.

5.1 Conclusions

The NPE has been enhanced to allow for the treatment of realistic ocean environments. It is now capable of treating multiple sediment layers which were approximated as fluids. The penetrable ocean bottom permitted energy to propagate to the bottom of the computational domain, so an artificial absorbing layer was implemented to damp out any energy that could cause unrealistic reflections off of the bottom of the grid. Comparing the NPE to the Fourier-transformed parabolic equation solution for a linear, range-independent case has shown that the penetrable ocean bottom NPE model is accurate. Allowing for the presence of nonlinear effects gave insight into how such effects altered the pulse propagation. Even the weak nonlinearity was able to change the solution significantly.

The updated NPE model was then adapted to treat range-dependent environments characterized by constant sloped ocean bottoms – a major aspect missing prior to this work. This was achieved by using a coordinate rotation to align the computational domain with the sloped ocean bottom. This led to a sloped top boundary that was approximated as a series of steps, easily satisfying the pressure release condition there. Care was needed to numerically implement the upper boundary condition, but the scheme used was shown to be efficient and introduce only a small error. Comparisons with the Fourier-transformed PE solution showed that the new rotated NPE model is accurate for shallow water environments with small slope angles in both upslope and downslope propagation.

This enhancements made have greatly increased the usefulness of the NPE model. It is still not as efficient as PE solutions, but it is able to capture nonlinear effects that have been shown to have an effect on pulse propagation in certain instances.

5.2 Future Work

The rotated NPE model was shown to be accurate in modeling constant sloped ocean bottoms with small slope angles. Further work is needed to better characterize the accuracy of this new model for a wide range of slope angles. The characterization of slope angle effects would include a study of the errors associated with the stair-step approximation and the vertical shift of the upper boundary.

There are a number of enhancements that can be made to the NPE model. A major enhancement would be to allow for elastic sediment layers in the seafloor. Currently, the model treats ocean sediment layers as fluids, but this is often unrealistic since many sediments can support shear waves. The current model is also not able to allow for a loss mechanism in the sediment layers. Waves propagating in sediments have a tendency to lose energy as it is dispersed to the surrounding media, so a physics-based loss mechanism is needed to be applied to the NPE. In order to increase accuracy near the source, a high-angle correction can be implemented into the model. This is a first-order correction to the narrow angle assumption made in the derivation of the NPE. This facilitates a more accurate treatment of wave energy traveling in the z direction, perpendicular to the propagation direction.

The main enhancement to be made is to adapt the model to handle variably sloped ocean bottoms. There are multiple methods that could be used to extend the current constant slope model to treat multiple slopes. One method for achieving this would be to use an interpolation-extrapolation scheme such as was done by Outing *et al.* (2006) with the PE method. This method, illustrated in Figure 5.1, marches the solution past the end of one slope to generate the initial waveform used for the following slope. In the case of the PE, only a line of values are interpolated and extrapolated to form the starting field. For the NPE,

an entire plane of values would need to be generated. Another method is to use a stationary form of the NPE to treat the area where the two slopes intersect by use of a skewed grid. An initial NPE solution would march into the intersection area to create the initial condition for the stationary NPE. The stationary NPE model would then propagate the wave to the next slope and generate an initial condition for the subsequent slope simulation.

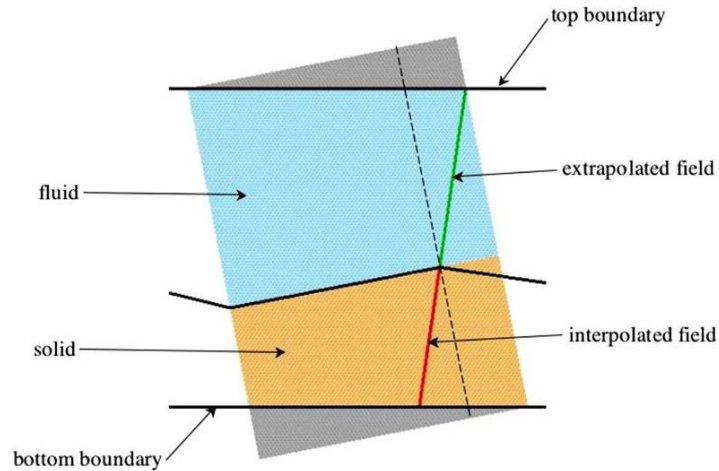


Figure 5.1: An interpolator-extrapolator scheme is used to treat variable slopes.

After a number of enhancements are made, there exists a wide range of interesting problems that can be studied with the NPE. The problem of modeling thin sediment layers with low shear waves speeds is of great interest in the ocean acoustics community and could be studied with the NPE. Another problem is to couple solutions of a first principles model to an NPE model and finally to a PE model to simulate blast wave propagation over entire ocean basins. A final study could look at using the NPE model reproduce results from tank test experiments to provide a definitive benchmark comparison.

REFERENCES CITED

- Ambrosiano, J. J., Plante, D. R., McDonald, B. E., & Kuperman, W. A. 1990. Nonlinear propagation in an ocean acoustic waveguide. *J. Acoust. Soc. Am.*, **87**(4), 1473–1481.
- Castor, K., Gerstoft, P., Roux, P., Kuperman, W. A., & McDonald, B. E. 2004. Long-range propagation of finite-amplitude acoustic waves in an ocean waveguide. *J. Acoust. Soc. Am.*, **116**(4), 2004–2010.
- Collins, M. D., & Evans, R. B. 1992. A two-way parabolic equation for acoustic backscattering in the ocean. *J. Acoust. Soc. Am.*, **91**(3), 1357–1368.
- Collins, Michael D. *User's Guide for RAM Versions 1.0 and 1.0p*. Naval Research Laboratory, Washington, DC 20375.
- Collis, J. M., Siegmann, W. L., Zampolli, M., & Collins, M. D. 2009. Extension of the rotated elastic parabolic equation to beach and island propagation. *IEEE Journal of Oceanic Engineering*, **34**(4), 617–623.
- Conte, Samuel D. 1972. *Elementary Numerical Analysis*. New York: McGraw-Hill.
- F. B. Jensen, W. A. Kuperman, M. B. Porter, & Schmidt, H. 2005. *Computational Ocean Acoustics*. American Institute of Physics.
- Jing, Y., & Cleveland, R. O. 2007. Modeling the propagation of nonlinear three-dimensional acoustic beams in inhomogeneous media. *J. Acoust. Soc. Am.*, **122**(3), 1352–1364.
- Kusel, Elizabeth Thorp. 2005. *New Parabolic Equation Solutions for High Frequency and Elastic Media Problems*. Ph.D. thesis, Rensselaer Polytechnic Institute.
- Leissing, Thomas. 2009 (November). *Propagation d'ondes non linéaires en milieu complexe Application la propagation en environnement urbain*. Ph.D. thesis, UNIVERSITE PARIS-EST.
- McDonald, B. E., & Ambrosiano, J. 1984. High-order upwind flux correction methods for hyperbolic conservation laws. *Journal of Computational Physics*, **56**, 448–460.
- McDonald, B. E., & Kuperman, W. A. 1987. Time domain formulation for pulse propagation including nonlinear behavior at a caustic. *J. Acoust. Soc. Am.*, **81**(5), 1406–1417.
- McDonald, B. E., & Piacsek, A. A. 2011. Nonlinear progressive wave equation for stratified atmospheres. *J. Acoust. Soc. Am.*, **130**(5), 2648–2653.

- McDonald, B. E., Caine, P., & West, M. 1994. A tutorial on the nonlinear progressive wave equation (NPE) – part 1. *Applied Acoustics*, **43**, 159–167.
- Outing, Donald A. 2004. *Parabolic equation methods for range dependent layered elastic media*. Ph.D. thesis, Rensselaer Polytechnic Institute.
- Outing, Donald A., Siegmann, William L., Collins, Michael D., & Westwood, Evan K. 2006. Generalization of the rotated parabolic equation to variable slopes. *J. Acoust. Soc. Am.*, **120**(6), 3534–3538.
- Porter, M. B. 1992 (May). *The KRAKEN Normal Mode Program*. Nrl/mr/5120–92-6920 edn. Naval Research Laboratory, Washington, DC 20375-5000.
- Taylor, Larissa. 2011. *Application of the nonlinear progressive wave equation to simulate nonlinear acoustic propagation near an interface*. M.S. thesis, Colorado School of Mines.

APPENDIX - A: COMPUTER PROGRAM LAYOUT

Main Program The main program first calls the input data reading procedure, `read_data`, which reads in the necessary values from the file `input.in`. The `read_data` subroutine reads in the following information:

| | |
|---------------------------|--|
| <code>num_layers</code> | number of sediment layers |
| <code>dx, dz, k</code> | x and z-direction grid spacing and time step |
| <code>beta</code> | coefficient of nonlinearity |
| <code>c_0, c_b</code> | average sound speed and array of sediment layer sound speeds |
| <code>rho_0, rho_b</code> | density of water and array of sediment layer densities |
| <code>x_range</code> | total range of simulation |
| <code>D_initial, W</code> | height and width of the computational grid |
| <code>D_b</code> | array of starting depths for each sediment layer |
| <code>D_damp</code> | starting depth of the artificial damping (absorbing) layer |

The procedure also reads in information pertaining to the initial condition and data saving/printing. Next, the main program initializes the overpressure field matrix, Q , as well as the grid parameter arrays x and z based on the values of `dx` and `dz`. Next, the `initial_Q` subroutine generates the starting field (initial condition) given user inputs. The program then calls the NPE solver, `NPE_water`, in which the bulk of the computation is completed. Finally, the main program deallocates memory and ends.

NPE Solver The NPE solver procedure first calls the `matrix_setup` subroutine to generate the sparse matrix arrays. The arrays generated include

| | |
|------------------------------------|--|
| <code>A_DL, A_D, A_DU</code> | lower, main, and upper diagonal of the A matrix |
| <code>LHS_DL, LHS_D, LHS_DU</code> | lower, main, and upper diagonal of the $I - \sigma A$ matrix |
| <code>RHS_DL, RHS_D, RHS_DU</code> | lower, main, and upper diagonal of the $I + \sigma A$ matrix |
| <code>SIG</code> | array of sigma values |

Next, the solver enters the time stepping procedure.

Time Step Loop

- Step 1: FCT algorithm is used to calculate the intermediate field, `Qhalf`. This is done by sending every column of the `Q` matrix to the `FCT` subroutine which outputs a column of `Qhalf`.
- Step 2: Enforce the wavefront boundary condition (bottom row of `Q` and `Qhalf` is set to zero).
- Step 3: Loop through the rows, starting from the last row and ending at the first row. Sum up the `Q` values from the current row to the last row for the current and previous time steps.
- Step 4: Perform the matrix multiplications specified in equation (2.44).
- Step 5: Solve the system of equations.
- Step 6: Enforce upper boundary condition by setting first column of `Q` to zero.
- Step 7: Save the appropriate data if specified by the user input and continue to the next time step.

The `NPE_water` procedure finishes upon completion of the time step loop. The parallel version of the code is very similar to the serial version except for some minor differences. In the main program, the MPI environment is initialized using the `MPI_INIT` call and the columns of the `Q` matrix are divided among the different processors. This is done with the ensuing statements

```
chunksize = (M + num_cores - 1)/num_cores
ALLOCATE(Q(N, my_rank*chunksize+1 : MIN(M, (my_rank + 1)*chunksize)))
```

Every processor then reads in the data using the `read_data` subroutine and then each processor initializes its set of columns using the `initial_Q` subroutine. Every processor calls the `NPE_water` procedure and then when that is finished the MPI environment is closed, the memory is deallocated, and the program is complete.

Within the `NPE_water` procedure, all matrices are again split up by columns. The sparse matrix diagonals are again created using the `matrix_setup` subroutine except now each processor contains a section of each array. Since Fortran does not have a built-in algorithm for solving a system of equations, a tridiagonal system solver was written. The solver used was the Tridiagonal Matrix Algorithm (TDMA), also known as the Thomas algorithm, because it is computationally efficient at handling tridiagonal matrix systems. This algorithm is explained in more detail below. The TDMA requires that certain arrays be built prior to time stepping, so the `FE_array_gen` subroutine is used to create these required arrays on each core. Within the time step loop, each core performs the `FCT` subroutine in parallel since it uses columns of the `Q` and not rows. The remainder of the program is essentially the same except that MPI send and receive commands are required for both the matrix-vector multiplications performed in Step 4 and the TDMA solver used in Step 5. Details of these procedures are given in the sub-sections below.

Tridiagonal Matrix Algorithm Solver In general, a tridiagonal system of equations is defined by

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i, \tag{A.1}$$

where $a_1 = 0$ and $c_n = 0$. In matrix form, this system is written as

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \cdot & \\ & & \cdot & \cdot & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \cdot \\ \cdot \\ d_n \end{bmatrix} \tag{A.2}$$

The TDMA solver requires two main steps: the forward elimination phase and the backward substitution phase. During the forward elimination phase, three new arrays are created,

m_{FE} , b_{FE} , and d_{FE} . Theoretically, no new arrays are needed since the old arrays can be overwritten with new values, but for the NPE code it is necessary to generate the new arrays (only once) so they do not have to be generated repeatedly during the time stepping. This algorithm requires only a small amount of computations which leads to a complexity of $\mathcal{O}(n)$ which is much more efficient than Gaussian elimination which has a complexity of about $\mathcal{O}(n^3)$. Details of the algorithm are given below.

Forward elimination phase:

$$(m_{FE})_1 = 0.0, \quad (b_{FE})_1 = b_1, \quad (d_{FE})_1 = d_1$$

for k = 2 step until n do

$$(m_{FE})_k = \frac{a_k}{(b_{FE})_{k-1}} \tag{A.3}$$

$$(b_{FE})_k = (b_{FE})_k - (m_{FE})_k c_{k-1} \tag{A.4}$$

$$(d_{FE})_k = d_k - (m_{FE})_k (d_{FE})_{k-1} \tag{A.5}$$

end loop (k)

Backward substitution phase:

$$x_n = \frac{(d_{FE})_n}{(b_{FE})_n} \tag{A.6}$$

for k = n-1 stepdown until 1 do

$$x_k = \frac{(d_{FE})_k - c_k x_{k+1}}{(b_{FE})_k} \tag{A.7}$$

end loop (k)

Note that this algorithm is only applicable to matrices that are diagonally dominant, which is to say $|b_i| > |a_i| + |c_i|$, $i = 1, \dots, n$ which is true in the case of the NPE program.

Parallel Communication Parallel communication between processors is needed for matrix-vector multiplications and the TDMA solver. The matrix-vector multiplications that occur twice within the `NPE_water` procedure are performed using the `tridiag_vec_mult` subroutine. A tridiagonal matrix-vector multiplication is given by

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \cdot & \\ & & \cdot & \cdot & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{bmatrix} \quad (\text{A.8})$$

In the parallel program, both the tridiagonal matrix and the vector being multiplied are subdivided among different processors. Processor 0 has rows 1 through M/P , processor 1 has rows $M/P + 1$ through $2M/P$ and so on. The last processor, processor $P - 1$, has rows $(P - 1)M/P + 1$ through M . Each processor needs values from neighboring processors in order to complete their work with the matrix-vector multiplication. Using `MPI_SEND` and `MPI_RECV`, the following communication is performed:

1. Processor 0 sends its last value of the vector array to processor 1 and receives the first value of the vector array from processor 1.
2. Processor n sends its first value of the vector array to processor $n - 1$ and also sends its last value of the vector array to processor $n + 1$. Processor n then receives the last value of the vector array from processor $n - 1$ and the first value of the vector array from processor $n + 1$.
3. Processor $P - 1$ sends its first value of the vector array to processor $P - 2$ and receives the last value of the vector array from processor $P - 2$.

Figure 3.4 gives an example of how this communication works with $M = 9$ and $P = 3$. In the example, the equations

$$\begin{aligned}
d_3 &= a_3x_2 + b_3x_3 + c_3x_4 \\
d_4 &= a_4x_3 + b_4x_4 + c_4x_5 \\
d_6 &= a_6x_5 + b_6x_6 + c_6x_7 \\
d_7 &= a_7x_6 + b_7x_7 + c_7x_8
\end{aligned}$$

require communication between the different processors. Processor 0 does not have the value x_4 , processor 1 does not have the values x_3 and x_7 , and processor 2 does not have the value x_6 . This neighbor communication is much more efficient than the “alltoall” type communication needed when multiplying a full matrix.

A similar method of communication is needed for the TDMA solver. The first processor performs the forward elimination phase and sends its last value of the d_{FE} array to the next processor. The second processor receives the value and goes on to generate its own d_{FE} array. When complete, it sends its last value of the d_{FE} array to the following processor. This continues until the last processor has completed the forward elimination phase at which point it starts the backward substitution phase. As soon as the last processor has completed the backward substitution phase it sends its first value in the solution array to previous processor. That processor receives the value and completes its own backward substitution and then sends the first value in its solution array to previous processor. This continues on until the first processor has completed the backward substitution phase.

During any point in this operation only a single processor is actually performing work. All other processors are idle and waiting to receive the necessary array values. This means that the TDMA solver is still a serial procedure and, when operating with more than one core, is less efficient than operating with a single core because of all the communication costs.