SANDIA REPORT
SAND2019-10517
Printed September 2019

Sandia National Laboratories

Paraniso 1.0: 3-D Full Waveform Seismic Simulation in General Anisotropic Media

Leiph A. Preston

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico
87185 and Livermore,
California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods/

ABSTRACT

Many geologic materials and minerals are seismically anisotropic, with the most general anisotropic material having up to 21 independent elastic coefficients. This report outlines the development of a 3-D, generally anisotropic, linear elastic full waveform finite-difference solver. First, a mathematical description of the solution equations will be described. The finite-difference implementation of these equations will then be shown. Finally, a comparison of results from this new solver to other solutions will be provided as verification that the new algorithm can accurately replicate these solutions.

ACKNOWLEDGEMENTS

The author also acknowledges the National Nuclear Security Administration, Defense Nuclear Nonproliferation Research and Development (DNN R&D), and the Source Physics Experiment (SPE) working group, a multi-institutional and interdisciplinary group of scientists and engineers.

CONTENTS

LIST OF FIGURES

This page left blank

ACRONYMS AND DEFINITIONS

Abbreviation	Definition
3-D	Three dimensional
PML	Perfectly matched layers
CPML	Convolutional perfectly matched laters
MPML	Multi-axial perfectly matched layers
CFL	Courant- Friedrichs-Lewy
MPI	Message passing interface

1. INTRODUCTION

In a seismically anisotropic material, seismic waves propagate with different wave speeds depending on the direction of propagation. Many common geologic minerals exhibit seismic anisotropy as a result of their crystal structure. Thus, seismic modeling of small-scale individual crystals would often necessitate anisotropic modeling. For larger scales of real (not manufactured) earth materials, the heterogeneous composition and random orientations of different minerals often greatly weakens anisotropy in bulk rock. However, large-scale structural patterns, such as preferential layering, faulting, joints, etc., can induce seismic anisotropy even in cases where the rock itself can be considered basically isotropic. Therefore, even at larger scales anisotropic modeling may be warranted. One particularly well-known bulk anisotropic material that has both preferred mineral orientations and fine-scale layering is shale (e.g., Johnston and Christensen, 1995).

The most general form of anisotropy (least symmetry) is called triclinic and has 21 independent elastic coefficients (e.g., Anderson, 1989). As symmetry increases, the number of independent elastic coefficients decrease. A monoclinic mineral has 13 elastic constants, orthorhombic has 9, various forms of tetragonal and trigonal have 6 or 7 moduli, hexagonal has 5, cubic has 3 independent coefficients, and finally, as the most symmetric form, an isotropic material has only 2 independent elastic moduli (Anderson, 1989).

Preston (2018) described implementation of an orthorhombic anisotropic algorithm, Pararhombi. As mentioned above, an orthorhombic medium consists of 9 independent elastic constants and has three orthogonal symmetry planes (Tsvankin, 1991). The fact that an orthorhombic material has three orthogonal symmetry planes allows us to use the same finite-difference grid approach as we use in our isotropic algorithms. However, this only works if the three orthogonal axes of the orthorhombic material align with the 3-D Cartesian axes that define the computational model domain. The earth is rarely that obliging. Once the orthorhombic axes are rotated off of the computational grid, then from the reference frame of the computation, all 21 elastic moduli are needed for the computation, even though there are still only 9 independent ones.

To handle the non-aligned orthorhombic case above as well as any arbitrary anisotropic media, we developed a 3-D generally anisotropic elastic full waveform simulation algorithm called Paraniso. This report describes the basic mathematical theory behind the algorithm, the implementation of the equations using the finite-difference method, and verification of the algorithm with some simple test cases. Note that due to the similarity of the topics discussed in Preston (2018), some material will be repeated where relevant for completeness.

2. THEORY

In this section, the physical equations and constitutive relations for a general anisotropic media are first described. We will then provide specialized equations for use at the earth's surface using the stress-free boundary condition assumption.

2.1. Physical Equations and Constitutive Relations

The basic physical equations and constitutive relations for linear anisotropic elastic media are well known and are provided in may texts, including Preston (2018). They are restated here for completeness. The equations of motion and time derivative of the stress-strain constitutive equations are (e.g., Aki and Richards, 2002):

$$\frac{\partial v_i(\mathbf{x},t)}{\partial t} = \frac{1}{\rho(\mathbf{x})}\left(\frac{\partial \sigma_{ij}(\mathbf{x},t)}{\partial x_j} + f_i(\mathbf{x},t)\right) \tag{2.1}$$

$$\frac{\partial \sigma_{ij}(\mathbf{x},t)}{\partial t} = C_{ijkl}(\mathbf{x})\frac{\partial v_k(\mathbf{x},t)}{\partial x_l} + \frac{\partial m_{ij}(\mathbf{x},t)}{\partial t} \tag{2.2}$$

where v_i is the i^{th} component of particle velocity, σ_{ij} is a component of the 3x3 symmetric stress tensor, ρ is density, f_i is a body force source term, C_{ijlk} is the elastic modulus tensor, m_{ij} is a symmetric moment tensor source, $\mathbf{x}$ is a point in 3-D space, and t is time. In Equations 2.1 and 2.2 repeated indices imply summation (Einstein summation convention). C_{ijkl} is a 4^{th} rank tensor; thus, it has 81 elements. However, it can be shown through symmetries of the stress and strain tensors and energy considerations that only 21 of these elements can be independent in a general anisotropic media (Aki and Richards, 2002).

Equations 2.1 and 2.2 form a coupled set of first-order partial differential equations called the velocity-stress system. This is the system of equations that we solve in Paraniso. Because of the of symmetries in C_{ijkl} mentioned above, one can write the stress-strain relationship in a more compact form often referred to as Voigt notation (e.g., Musgrave, 1970):

$$\begin{bmatrix}\sigma_{xx}\\ \sigma_{yy}\\ \sigma_{zz}\\ \sigma_{yz}\\ \sigma_{xz}\\ \sigma_{xy}\end{bmatrix} = \begin{bmatrix} c_{xxxx} & c_{xxyy} & c_{xxzz} & c_{xxyz} & c_{xxxz} & c_{xxxy}\\ c_{xxyy} & c_{yyyy} & c_{yyzz} & c_{yyyz} & c_{yyxz} & c_{yyxy}\\ c_{xxzz} & c_{yyzz} & c_{zzzz} & c_{zzyz} & c_{zzxz} & c_{zzxy}\\ c_{xxyz} & c_{yyyz} & c_{zzyz} & c_{yzyz} & c_{yzxz} & c_{yzxy}\\ c_{xxxz} & c_{yyxz} & c_{zzxz} & c_{yzxz} & c_{xzxz} & c_{xzxy}\\ c_{xxxy} & c_{yyxy} & c_{zzxy} & c_{yzxy} & c_{xzxy} & c_{xyxy}\end{bmatrix}\begin{bmatrix}\epsilon_{xx}\\ \epsilon_{yy}\\ \epsilon_{zz}\\ 2\epsilon_{yz}\\ 2\epsilon_{xz}\\ 2\epsilon_{xy}\end{bmatrix} \tag{2.3}$$

where the 6x6 matrix is a compact version of C_{ijlk} for general anisotropy and $\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$ is the strain with u_i being the displacement. Note that there are indeed 21 unique components in the symmetric matrix of Equation 2.3.

There are mathematical and physical constraints on the permissible values and combinations of values that are allowed for the values of C_{ijkl}. Square roots of certain linear combinations of different C_{ijkl} divided by densities yield wave speeds in different directions, which must be real

and positive. However, for a general anisotropic media these relationships are complicated and, especially the physical constraints, for an arbitrary (physically possible) medium, are poorly understood. The basic mathematical and physical requirement is that the matrix in Equation 2.3 be positive definite. As a consequence all diagonal components of the matrix must be positive. Bennett (1972) recasts the problem in terms of a Q-ellipsoid, a geometric interpretation of Equation 2.3, that can be used to visualize and determine relationships among the elastic moduli.

2.2. Stress-Free Surface Equations

Although in reality the earth's surface is a boundary between rock and air, it is sometimes convenient to assume the air does not exist, i.e, the earth surface is a boundary between rock and vacuum. Given that air has a density and strength orders of magnitude less than rock, this is a very good approximation in many cases. If one assumes vacuum above the earth's surface, then the boundary condition will be stress-free at the surface since a vacuum cannot support any stresses and the normal components of stresses at the interface must be continuous across an interface (Aki and Richards, 2002). The basic conditions imposed by stress-free boundary conditions for the velocity-stress system of equations, assuming a horizontal surface (vertical normal), are:

$$\frac{\partial \sigma_{zz}}{\partial t} = \frac{\partial \sigma_{xz}}{\partial t} = \frac{\partial \sigma_{yz}}{\partial t} = 0 \tag{2.4}$$

Writing these out for Equation 2.2 without a source term:

$$\begin{aligned} 0 = \frac{\partial \sigma_{zz}}{\partial t} = c_{xxzz}\frac{\partial v_x}{\partial x} + c_{yyzz}\frac{\partial v_y}{\partial y} + c_{zzzz}\frac{\partial v_z}{\partial z} + c_{zzxy}\left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x}\right) \\ + c_{zzxz}\left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x}\right) + c_{zzyz}\left(\frac{\partial v_z}{\partial y} + \frac{\partial v_y}{\partial z}\right) \end{aligned} \tag{2.5}$$

$$\begin{aligned} 0 = \frac{\partial \sigma_{xz}}{\partial t} = c_{xxxz}\frac{\partial v_x}{\partial x} + c_{yyxz}\frac{\partial v_y}{\partial y} + c_{zzxz}\frac{\partial v_z}{\partial z} + c_{xzxy}\left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x}\right) \\ + c_{xzxz}\left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x}\right) + c_{yzxz}\left(\frac{\partial v_z}{\partial y} + \frac{\partial v_y}{\partial z}\right) \end{aligned} \tag{2.6}$$

$$\begin{aligned} 0 = \frac{\partial \sigma_{yz}}{\partial t} = c_{xxyz}\frac{\partial v_x}{\partial x} + c_{yyyz}\frac{\partial v_y}{\partial y} + c_{zzyz}\frac{\partial v_z}{\partial z} + c_{yzxy}\left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x}\right) \\ + c_{yzxz}\left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x}\right) + c_{yzyz}\left(\frac{\partial v_z}{\partial y} + \frac{\partial v_y}{\partial z}\right) \end{aligned} \tag{2.7}$$

These equations impose constraints on the velocity gradient terms at the interface. We can solve the above system of equations for the vertical derivative terms of the velocities, given the horizontal derivative terms. The vertical derivative terms are:

$$\begin{aligned}\frac{\partial v_x}{\partial z} = \frac{1}{D}\Bigg[&\left(c_{xzyz}c_{zzyz} - c_{yzyz}c_{zzxz}\right)\left(c_{xxzz}\frac{\partial v_x}{\partial x} + c_{yyzz}\frac{\partial v_y}{\partial y} + c_{zzxy}\frac{\partial v_x}{\partial y} + c_{zzxz}\frac{\partial v_z}{\partial x}\right.\\ &\left. + c_{zzyz}\frac{\partial v_z}{\partial y} + c_{zzxy}\frac{\partial v_y}{\partial x}\right) - \left(c_{xzyz}c_{zzzz} - c_{zzxz}c_{zzyz}\right)\left(c_{xxyz}\frac{\partial v_x}{\partial x} + c_{xyyz}\frac{\partial v_x}{\partial y}\right.\\ &\left. + c_{xyyz}\frac{\partial v_y}{\partial x} + c_{xzyz}\frac{\partial v_z}{\partial x} + c_{yyyz}\frac{\partial v_y}{\partial y} + c_{yzyz}\frac{\partial v_z}{\partial y}\right) + \left(c_{yzyz}c_{zzzz} - c_{zzyz}^2\right)\left(c_{xxxz}\frac{\partial v_x}{\partial x}\right.\\ &\left. + c_{xyxz}\frac{\partial v_x}{\partial y} + c_{xyxz}\frac{\partial v_y}{\partial x} + c_{xzxz}\frac{\partial v_z}{\partial x} + c_{xzyz}\frac{\partial v_z}{\partial y} + c_{yyxz}\frac{\partial v_y}{\partial y}\right)\Bigg]\end{aligned} \tag{2.8}$$

$$\begin{aligned}\frac{\partial v_y}{\partial z} = \frac{1}{D}\Bigg[&-\left(c_{xzxz}c_{zzyz} - c_{xzyz}c_{zzxz}\right)\left(c_{xxzz}\frac{\partial v_x}{\partial x} + c_{yyzz}\frac{\partial v_y}{\partial y} + c_{zzxy}\frac{\partial v_x}{\partial y} + c_{zzxz}\frac{\partial v_z}{\partial x}\right.\\ &\left. + c_{zzyz}\frac{\partial v_z}{\partial y} + c_{zzxy}\frac{\partial v_y}{\partial x}\right) + \left(c_{xzxz}c_{zzzz} - c_{zzxz}^2\right)\left(c_{xxyz}\frac{\partial v_x}{\partial x} + c_{xyyz}\frac{\partial v_x}{\partial y} + c_{xyyz}\frac{\partial v_y}{\partial x}\right.\\ &\left. + c_{xzyz}\frac{\partial v_z}{\partial x} + c_{yyyz}\frac{\partial v_y}{\partial y} + c_{yzyz}\frac{\partial v_z}{\partial y}\right) - \left(c_{xzyz}c_{zzzz} - c_{zzxz}c_{zzyz}\right)\left(c_{xxxz}\frac{\partial v_x}{\partial x}\right.\\ &\left. + c_{xyxz}\frac{\partial v_x}{\partial y} + c_{xyxz}\frac{\partial v_y}{\partial x} + c_{xzxz}\frac{\partial v_z}{\partial x} + c_{xzyz}\frac{\partial v_z}{\partial y} + c_{yyxz}\frac{\partial v_y}{\partial y}\right)\Bigg]\end{aligned} \tag{2.9}$$

$$\frac{\partial v_z}{\partial z} = \frac{1}{D}\left[\left(c_{xzxz}c_{yzyz} - c_{xzyz}^2\right)\left(c_{xxzz}\frac{\partial v_x}{\partial x} + c_{yyzz}\frac{\partial v_y}{\partial y} + c_{zzxy}\frac{\partial v_x}{\partial y} + c_{zzxz}\frac{\partial v_z}{\partial x} + c_{zzyz}\frac{\partial v_z}{\partial y}\right.\right.$$
$$\left. + c_{zzxy}\frac{\partial v_y}{\partial x}\right) - \left(c_{xzxz}c_{zzyz} - c_{xzyz}c_{zzxz}\right)\left(c_{xxyz}\frac{\partial v_x}{\partial x} + c_{xyyz}\frac{\partial v_x}{\partial y} + c_{xyyz}\frac{\partial v_y}{\partial x}\right.$$
$$\left. + c_{xzyz}\frac{\partial v_z}{\partial x} + c_{yyyz}\frac{\partial v_y}{\partial y} + c_{yzyz}\frac{\partial v_z}{\partial y}\right) + \left(c_{xzyz}c_{zzyz} - c_{yzyz}c_{zzxz}\right)\left(c_{xxxz}\frac{\partial v_x}{\partial x}\right.$$
$$\left.\left. + c_{xyxz}\frac{\partial v_x}{\partial y} + c_{xyxz}\frac{\partial v_y}{\partial x} + c_{xzxz}\frac{\partial v_z}{\partial x} + c_{xzyz}\frac{\partial v_z}{\partial y} + c_{yyxz}\frac{\partial v_y}{\partial y}\right)\right] \tag{2.10}$$

with

$$D = -c_{xzxz}c_{yzyz}c_{zzzz} + c_{xzxz}c_{zzyz}^2 + c_{xzyz}^2c_{zzzz} - 2c_{xzyz}c_{zzyz}c_{zzxz} + c_{yzyz}c_{zzxz}^2 \tag{2.11}$$

Equation 2.11 imposes another constraint: it cannot equal zero. These equations do simplify to the much easier to read (and derive) equations for an isotropic elastic free surface in that special case.

3. IMPLEMENTATION

The equations given in Section 2 are given for continuous space and time variables. To implement the solution of those equations on a computer, we will discretize those equations using a finite-difference approach.

3.1. Finite-Difference Grid

The equations of motion (Equation 2.1) are identical for all forms of anisotropy including isotropic media. However, the structure of Equation 2.3 (and hence Equation 2.2) does change. For isotropic and the coordinate-aligned orthorhombic media described in Preston (2018), the structure of Equation 2.3 allows one to use a standard staggered grid where velocity and stress nodes are offset 1/2 grid node from each other spatially. This is due to the fact that only certain spatial derivatives of velocities are needed for each stress term. For a general anisotropic media, however, all of the spatial derivatives for all of velocities are needed for each stress term. Thus, a standard staggered grid cannot be used. We must use a standard non-staggered grid where all stresses, velocities, and medium parameters are collocated spatially (Figure 3.1). However, given that the velocity equations (Equation 2.1) only depend on stresses and that the stress updating equations (Equations 2.2) only depend on velocities (and assumed time-invariant medium parameters), a staggered grid in time can still be used for generally anisotropic media. In the staggered time updating scheme, all stress components are updated on the integer time step raster, while all velocities terms are updated on the half-integer time raster.

3.2. Finite-Difference Equations

Equations 2.1 and 2.2 are discretized according to the grid outlined in Section 3.1 and solved using 4th and 2nd order accurate finite-difference operators in space and time, respectively. Standard Taylor Series finite-difference coefficients are used by default, but others may be substituted if desired. Paraniso utilizes an explicit, leap-frog approach for time stepping. Since

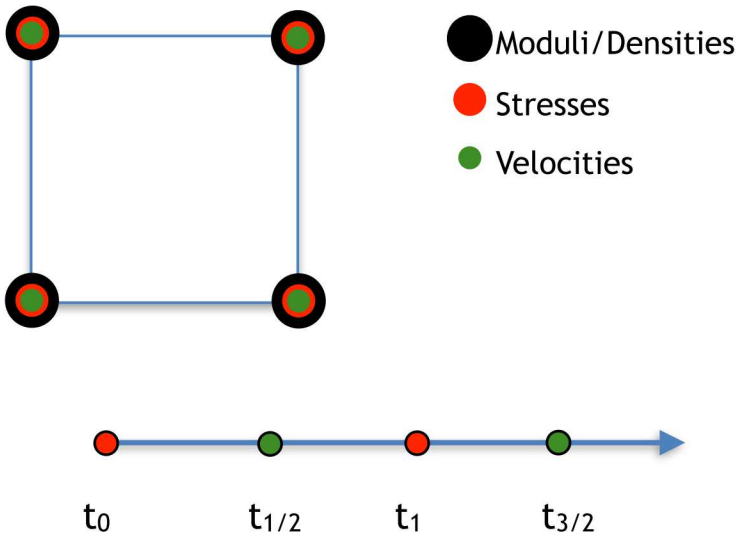

Figure 3.1: One face of a unit cell (top) and time axis (bottom) for the finite-difference scheme.

we are utilizing a non-staggered spatial grid, all three velocity updating equations will have the same form. Similarly, all stress updating equations will have the same form. Thus, we show only one updating equation for a generic velocity and one for a generic stress.

In the following equations, $p_x = \frac{h_t}{h_x} D_{inner}$, $q_x = \frac{h_t}{h_x} D_{outer}$, h_t is the time step, h_x is the grid node spacing in the x-dimension, and D_{inner} and D_{outer} are the finite difference coefficients, which are 2/3 and -1/12 for Taylor Series coefficients. Similarly, p_y, q_y, p_z, and q_z are defined analogously. Indices i, j, k, and l are integers specifying the grid node locations for the x, y, z and time dimensions, respectively.

Source terms, f_x, f_y, f_z, m_{xx}, etc., are trilinearly extrapolated to the 8 respective dependent variable nodes surrounding the input source location and linearly interpolated in time, if needed. Point measurements of the dependent variables (i.e., receivers) are performed with trilinear or tricubic interpolation of the surrounding respective dependent variable nodes, depending on user input. All receiver data are linearly interpolated in time, if needed, to the integer time raster. Besides receivers that measure velocity, one can also choose to measure pressure, acceleration, displacement, or rotation rate. Pressure is defined as $p = -\frac{1}{3}\left(\sigma_{xx} + \sigma_{yy} + \sigma_{zz}\right)$. Rotation rate is the curl of the velocity field.

3.2.1. Velocity

$$
\begin{aligned}
v_a\left(x_i, y_j, z_k, t_{l+1/2}\right) = v_a\left(x_i, y_j, z_k, t_{l-1/2}\right) + \frac{1}{\rho\left(x_i, y_j, z_k\right)} \times \\
\Bigg[p_x \left[\sigma_{ax}\left(x_{i+1}, y_j, z_k, t_l\right) - \sigma_{ax}\left(x_{i-1}, y_j, z_k, t_l\right)\right] \\
+ q_x \left[\sigma_{ax}\left(x_{i+2}, y_j, z_k, t_l\right) - \sigma_{ax}\left(x_{i-2}, y_j, z_k, t_l\right)\right] \\
+ p_y \left[\sigma_{ay}\left(x_i, y_{j+1}, z_k, t_l\right) - \sigma_{ay}\left(x_i, y_{j-1}, z_k, t_l\right)\right] \\
+ q_y \left[\sigma_{ay}\left(x_i, y_{j+2}, z_k, t_l\right) - \sigma_{ay}\left(x_i, y_{j-2}, z_k, t_l\right)\right] \\
+ p_z \left[\sigma_{az}\left(x_i, y_j, z_{k+1}, t_l\right) - \sigma_{az}\left(x_i, y_j, z_{k-1}, t_l\right)\right] \\
+ q_z \left[\sigma_{az}\left(x_i, y_j, z_{k+2}, t_l\right) - \sigma_{az}\left(x_i, y_j, z_{k-2}, t_l\right)\right] \\
+ h_t f_a\left(x_i, y_j, z_k, t_{l+1/2}\right)\Bigg]
\end{aligned}
\tag{3.1}
$$

Replace all occurrences of subscript *a* with x, y, or z in Equation 3.1 to obtain the equation for a specific velocity term.

3.2.2. Stress

$$\begin{aligned}\sigma_{ab}\left(x_i, y_j, z_k, t_{l+1}\right) = \sigma_{ab}\left(x_i, y_j, z_k, t_l\right) &+ C_{xxab}\left(x_i, y_j, z_k\right) \times D_x\left[v_x\right]\left(x_i, y_j, z_k, t_{l+1/2}\right) \\ &+ C_{yyab}\left(x_i, y_j, z_k\right) \times D_y\left[v_y\right]\left(x_i, y_j, z_k, t_{l+1/2}\right) \\ &+ C_{zzab}\left(x_i, y_j, z_k\right) \times D_z\left[v_z\right]\left(x_i, y_j, z_k, t_{l+1/2}\right) \\ &+ C_{xyab}\left(x_i, y_j, z_k\right) \times D_x\left[v_y\right]\left(x_i, y_j, z_k, t_{l+1/2}\right) \\ &+ C_{xyab}\left(x_i, y_j, z_k\right) \times D_y\left[v_x\right]\left(x_i, y_j, z_k, t_{l+1/2}\right) \\ &+ C_{xzab}\left(x_i, y_j, z_k\right) \times D_x\left[v_z\right]\left(x_i, y_j, z_k, t_{l+1/2}\right) \\ &+ C_{xzab}\left(x_i, y_j, z_k\right) \times D_z\left[v_x\right]\left(x_i, y_j, z_k, t_{l+1/2}\right) \\ &+ C_{yzab}\left(x_i, y_j, z_k\right) \times D_y\left[v_z\right]\left(x_i, y_j, z_k, t_{l+1/2}\right) \\ &+ C_{yzab}\left(x_i, y_j, z_k\right) \times D_z\left[v_y\right]\left(x_i, y_j, z_k, t_{l+1/2}\right) \\ &+ m_{ab}\left(x_i, y_j, z_k, t_{l+1}\right)\end{aligned} \tag{3.2}$$

where

$$\begin{aligned}D_x\left[v_n\right]\left(x_i, y_j, z_k, t_{l+1/2}\right) = \Bigg[&p_x\left[v_n\left(x_{i+1}, y_j, z_k, t_{l+1/2}\right) - v_n\left(x_{i-1}, y_j, z_k, t_{l+1/2}\right)\right] \\ &+ q_x\left[v_n\left(x_{i+2}, y_j, z_k, t_{l+1/2}\right) - v_n\left(x_{i-2}, y_j, z_k, t_{l+1/2}\right)\right]\Bigg]\end{aligned} \tag{3.3}$$

$$\begin{aligned}D_y\left[v_n\right]\left(x_i, y_j, z_k, t_{l+1/2}\right) = \Bigg[&p_y\left[v_n\left(x_i, y_{j+1}, z_k, t_{l+1/2}\right) - v_n\left(x_i, y_{j-1}, z_k, t_{l+1/2}\right)\right] \\ &+ q_y\left[v_n\left(x_i, y_{j+2}, z_k, t_{l+1/2}\right) - v_n\left(x_i, y_{j-2}, z_k, t_{l+1/2}\right)\right]\Bigg]\end{aligned} \tag{3.4}$$

$$D_z\left[v_n\right]\left(x_i, y_j, z_k, t_{l+1/2}\right) = \left[p_z\left[v_n\left(x_i, y_j, z_{k+1}, t_{l+1/2}\right) - v_n\left(x_i, y_j, z_{k-1}, t_{l+1/2}\right)\right] + q_z\left[v_n\left(x_i, y_j, z_{k+2}, t_{l+1/2}\right) - v_n\left(x_i, y_j, z_{k-2}, t_{l+1/2}\right)\right]\right] \tag{3.5}$$

Substitute subscripts a and b throughout Equation 3.2 and the subscript n in Equations 3.4-3.6 with x, y, z to obtain the updating formula for a specific stress term.

3.3. Accuracy and Stability

By using a non-staggered grid in space and a staggered grid in time, all space and time finite-difference equations will be centered, which has favorable numerical characteristics. However, since the spatial extent of a finite-difference operator is larger in the non-staggered scheme compared to a staggered scheme like that used in isotropic or coordinate-aligned orthorhombic media, finer spatial grid spacing will be required to achieve the same level of numerical accuracy. Using standard Neumann analysis one can theoretically predict the numerical wave speed as a function of wavenumber relative to the medium's physical wave speed (see Haney and Aldridge, 2008 for a detailed description of the method). Figure 3.2 demonstrates the difference in accuracy between a standard staggered grid and a non-staggered grid. The sampling parameter is the reciprocal of the number of grid nodes per wavelength so the smaller the sampling parameter is where the curve significantly departs from the ideal of unity, the lower the accuracy

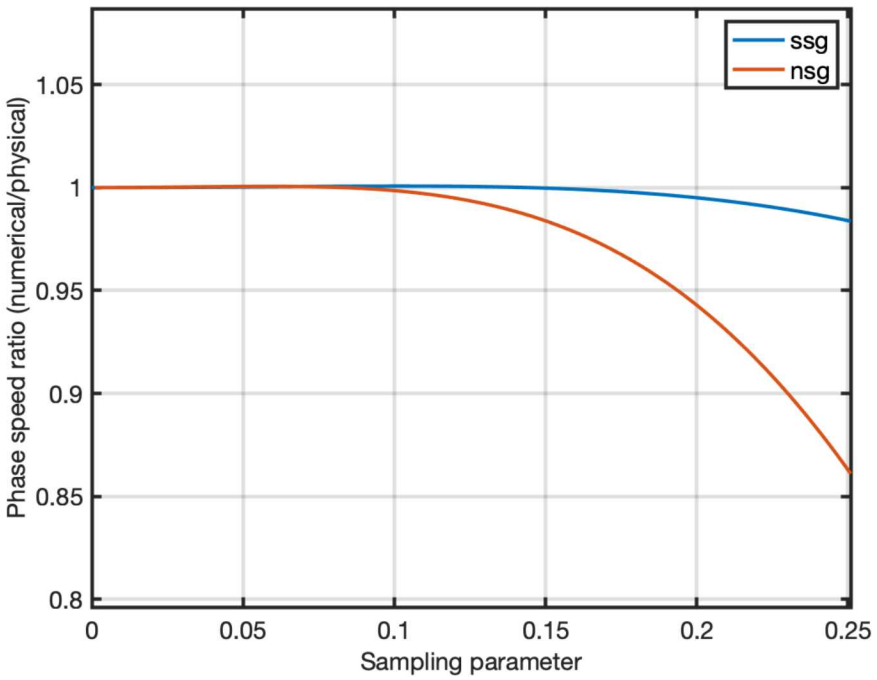

Figure 3.2: Sampling parameter (reciprocal of number of grid nodes per wavelength) vs. the numerical phase speed ratio. Ideal is unity. ssg: standard staggered grid. nsg: non-staggered grid.

of the method at a given number of grid nodes per wavelength. Alternatively, for a given level of accuracy more grid nodes are needed per wavelength (smaller grid spacing). Based on numerical tests about 40% higher resolution is needed to achieve the same level of accuracy, so, for example, if you could use 1 m node spacing in a standard staggered algorithm, then to obtain the same accuracy with Paraniso, one would need a grid spacing of about 0.7 m.

Based on numerical dispersion studies for isotropic elastic media, one should design the grid node spacing to be

$$h_x \leq \frac{V_{min}}{a f_{max}} \tag{3.6}$$

where V_{min} is the minimum wave speed in the model, f_{max} is the maximum frequency desired, and a is a constant that determines the number of grid nodes per wavelength needed for accuracy (e.g., Haney and Aldridge, 2008). For flat or low relief topography, $a = 8.8$ is sufficient if the time step is chosen to be 0.6 of the Courant-Friedrichs-Lewy (CFL) limit (see below). Otherwise, $a = 14$ is optimal.

The CFL stability limit describes a maximum allowable time step for explicit time stepping algorithms. Above the CFL limit the algorithm will go unstable. The CFL limit for Paraniso is

$$h_t \leq \frac{h_x}{b V_{max}} \tag{3.7}$$

where V_{max} is the maximum velocity present in the model and b is a constant that depends on the specifics of the algorithm. For Paraniso $b = 1.299$. This actually gives a 56% larger time step than would be allowed with a standard staggered grid. Thus, part of the higher cost associated with a non-staggered grid is offset by the larger time step. As mentioned above, the actual h_t used in Paraniso should be 0.6 of the CFL limit for maximum accuracy versus runtime.

It should be noted that runtime of Paraniso goes as the fourth power of $1/h_x$, like virtually all 3-D time-domain algorithms. Thus, halving the grid node spacing will increase the computational expense by a factor of 16. Note that halving the grid node spacing is equivalent to doubling the maximum frequency, halving the minimum wave speed, or some combination of the above from Equation 3.6. Simply changing the volume of the model, assuming nothing else changes, will change the runtime by the ratio in volumes. Altering the maximum simulation time or h_t, each by themselves, affects the runtime linearly. Memory requirements are roughly related to the model volume, not counting receivers.

3.4. Absorbing Boundary Conditions

Often, one desires only to run a simulation of a portion of the earth, instead of the whole earth. When one truncates the earth model at artificial boundaries defined by the numerical 3-D domain of the simulation space, it produces artificial reflections at these boundaries that are unphysical. In order that the solution not be polluted by these artificial reflections, an absorbing boundary is constructed along the flanks of the model space to mitigate these undesirable artifacts. Paraniso uses a Multiaxial Perfectly Matched Layer (MPML) (Meza-Fajardo and Papageorgiou, 2008) to efficiently absorb the numerical domain boundary reflections. MPMLs are similar to classical Perfectly Matched Layers (PMLs) (Berenger, 1994) and Convolutional

Perfectly Matched Layers (CPMLs) (Komatitsch and Martin, 2007). The primary difference is that an MPML absorbs energy for waves propagating parallel to domain boundaries instead of perpendicular only. This, of course, does ruin the "perfect" portion of PMLs. A true PML, in a continuum, produces no reflections upon entry into the PML zones. Conversely, for an MPML in a continuum there will be some reflection from the onset of the zone. In a discrete model, however, even true PMLs do produce some reflections upon entry into the zone. Thus, a true PML would seem to be a better choice except for the fact that PMLs are known to go unstable in certain conditions for anisotropic media (Komatitsch and Martin, 2007). MPMLs, on the other hand, can be used stably in anisotropic models, knowing that the price is larger reflections at the onset of the zone.

An MPML is actually a superset of a CPML, which is itself a superset of a PML. Thus, an MPML can support CPML and PML runs, with their advantages, if the model allows it. An MPML contains a so-called cross factor. This cross factor is a fraction that gives the amount of absorption in the parallel directions relative to the perpendicular direction. A cross factor of zero produces a pure CPML. The smaller the cross factor, the closer to a true PML and, thus, smaller reflections at the onset of the zone. Typically, 0.01 to 0.05 are sufficient in many anisotropy models, if needed. It should be noted that many anisotropic models will run stably even with the cross factor set to zero. Thus, for new models one should attempt to run the simulation first with a cross factor of zero and resort to larger cross factors only if warranted. For realistic models with 3-D variability, there is not presently the capability to determine beforehand which models will need non-zero cross factors.

Since an MPML is a superset of CPMLs and PMLs, an MPML also requires the parameters necessary to describe these zones. A PML requires the width of the PML zone (typically 10 nodes) along each of the flanks of the model (except for the Zmin flank if a free surface condition is used) and the theoretical reflection coefficient at the flank boundary (typically 1e-3 to 1e-5). A CPML additionally requires a corner frequency (typically about π times the spectral peak frequency of the simulation) and a stretch factor (which is best to always set to 1 for optimal performance).

3.5. Massively Parallel Implementation

Paraniso uses the Message Passing Interface (MPI) to divide the problem into smaller portions that multiple cores and/or machines can work on simultaneously. The full 3-D model domain is subdivided into user-specified subdomains that allow parallel computation. Only nodes directly on the edges of each of these subdomains need to share information with their direct neighbors. This allows high parallel efficiency in optimal circumstances. Unfortunately, the optimal circumstance is dependent on a variety of hardware- and software-related parameters. For example, using too many processes for a given model can hurt parallel efficiency due to high MPI overhead relative to computation time. Whereas too few processes can overload memory and cause excessive runtimes for large models. For most systems, a rule of thumb is to subdivide the z-dimension the most, followed by y, and have the x-dimension subdivide least. This allows maximum cache coherency for the computations.

Besides running Paraniso in parallel, the input and output files can be split into more manageable sizes. Older netcdf versions had a hard limit of 2 GB maximum for a single variable in a model file. Thus, to build bigger models, one needed to subdivide the netcdf files into files that each

met the maximum netcdf requirements. Modern versions of netcdf do allow for larger variables, but it can still at times be more convenient and manageable to break up very large models into smaller portions.

4. VERIFICATION

Paraniso is a software product and, thus, needs to be tested against accepted solutions to ensure Paraniso is able to perform its function accurately. In this section, Paraniso will be compared with accepted solutions from an isotropic elastic algorithm and from the orthorhombic anisotropy algorithm Pararhombi.

4.1. Isotropic Elastic Comparison

As mentioned earlier, an isotropic medium is a special case of a general anisotropic medium. Thus, Paraniso should be able to replicate an isotropic elastic solution. The accepted code for isotropic elastic media is Parelasti, which is a Sandia-developed 3-D finite-difference code used for years in the Geophysics Department.

The medium is an isotropic elastic whole space with isotropic Vp = 2500 m/s, Vs = 1500 m/s and density = 2000 kg/m^3. A vertical force source is placed in the center of a 100 m by 100 m by 100 m domain. Vx and Vz receivers were placed from -40 m to +40 m in the x-direction at 5 m increments, all at +10 m in the z direction, from the source. Figures 4.1 and 4.2 show the

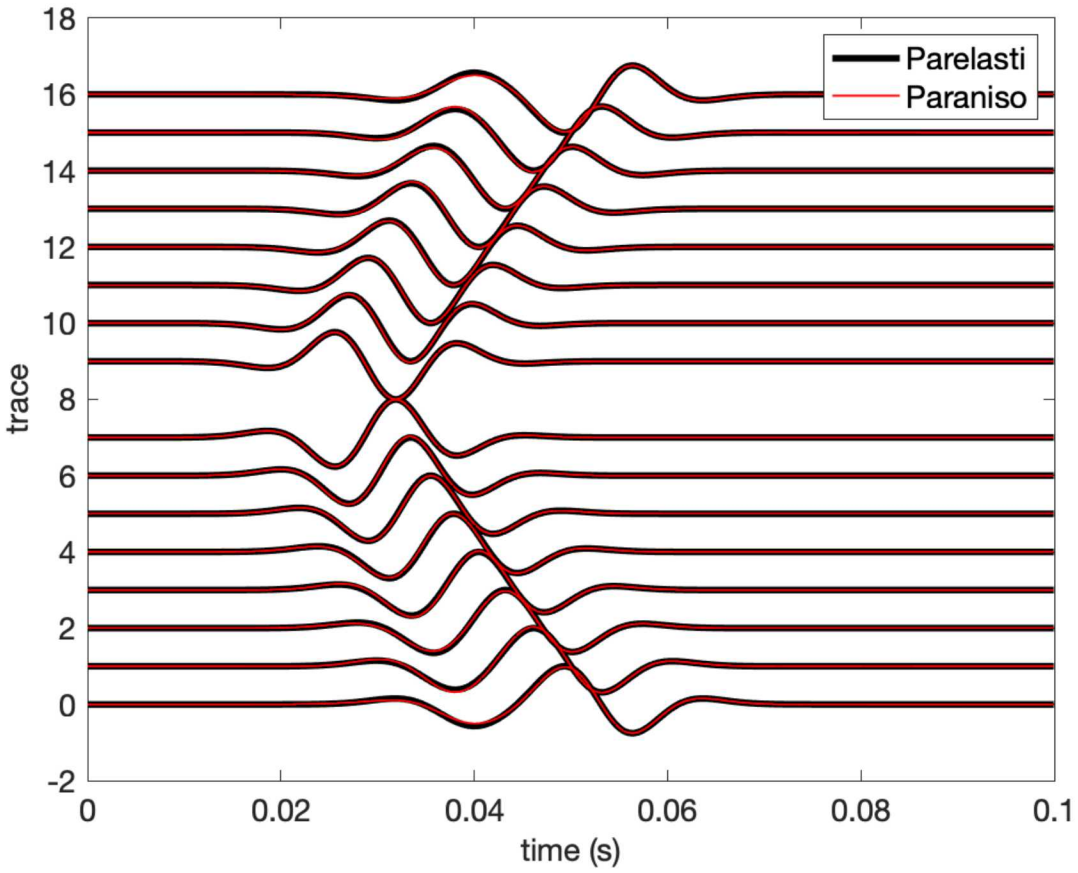

Figure 4.1: Comparison of Vx traces for an isotropic model between Parelasti (black) and Paraniso (red dashed). Trace normalized by Parelasti amplitudes.

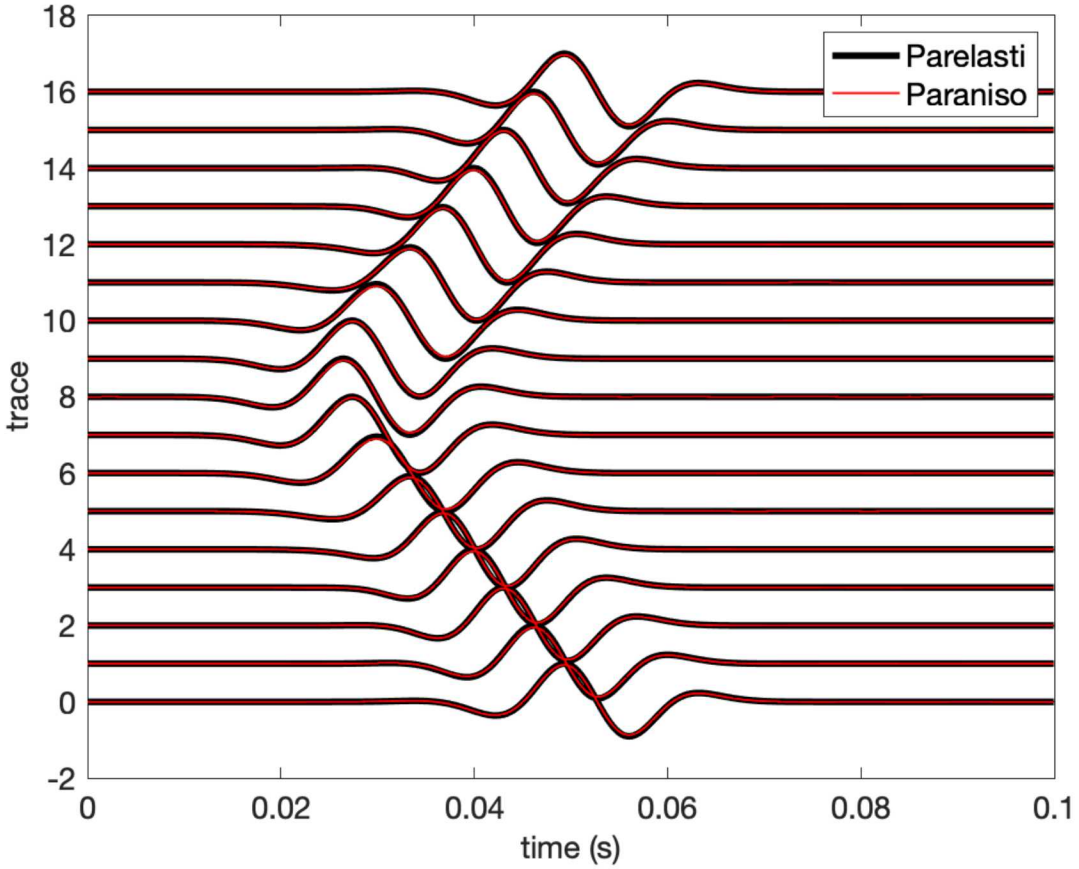

Figure 4.2: Comparison of Vz faces for an isotropic model between Parelasti (black) and Paraniso (red dashed). Trace normalized by Parelasti amplitudes.

excellent agreement between the two algorithms in this test. This indicates that Paraniso can properly reduce to isotropic media.

4.2. Orthorhombic Elastic Comparison

An orthorhombic media is also a special case of a general anisotropic one. As mentioned earlier, Pararhombi is an axis-aligned orthorhombic anisotropy elastic simulation algorithm. Its implementation is described in detail in Preston (2018). We use Pararhombi simulation results as a benchmark to compare with the new algorithm results.

The orthorhombic model is an elastic whole space with the following properties: V_{Px} = 3320 m/s, V_{Py} = 3472 m/s, V_{Pz} = 2697 m/s, V_{Pxy} = 3264 m/s, V_{Pxz} = 2845 m/s, V_{Pyz} = 3001 m/s, V_{Sxy} = 1635 m/s, V_{Sxz} = 1400 m/s, V_{Syz} = 1565 m/s, density = 2500 kg/m^3. V_{Px} is the P-wave speed for propagation along the x-axis, V_{Pxy} is the P-wave speed for propagation in the xy plane at 45° to each axis, V_{Sxy} is the shear wave speed for propagation along the x-axis, polarized in the y-direction (or vice versa); other variables are analogously named. An x-directed force source is located in the center of a 1000 m by 1000 m by 1000 m grid. Vx receivers are placed every 10 m on three axial lines from the source: one line begins at +10 m in x from the source and extends to +300 m in x, with y and z aligned with the source; a second line is identical to the first but the

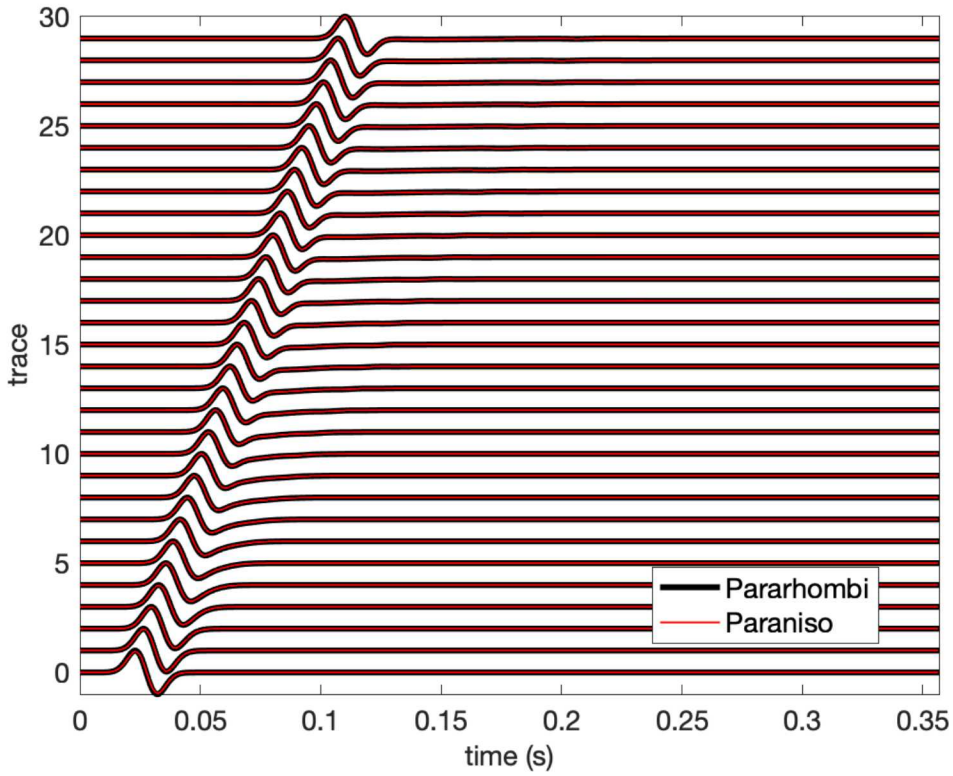

Figure 4.3: Comparison between Pararhombi and Paraniso for Vx receivers along the x-direction from the source. Trace normalized by Pararhombi amplitudes.

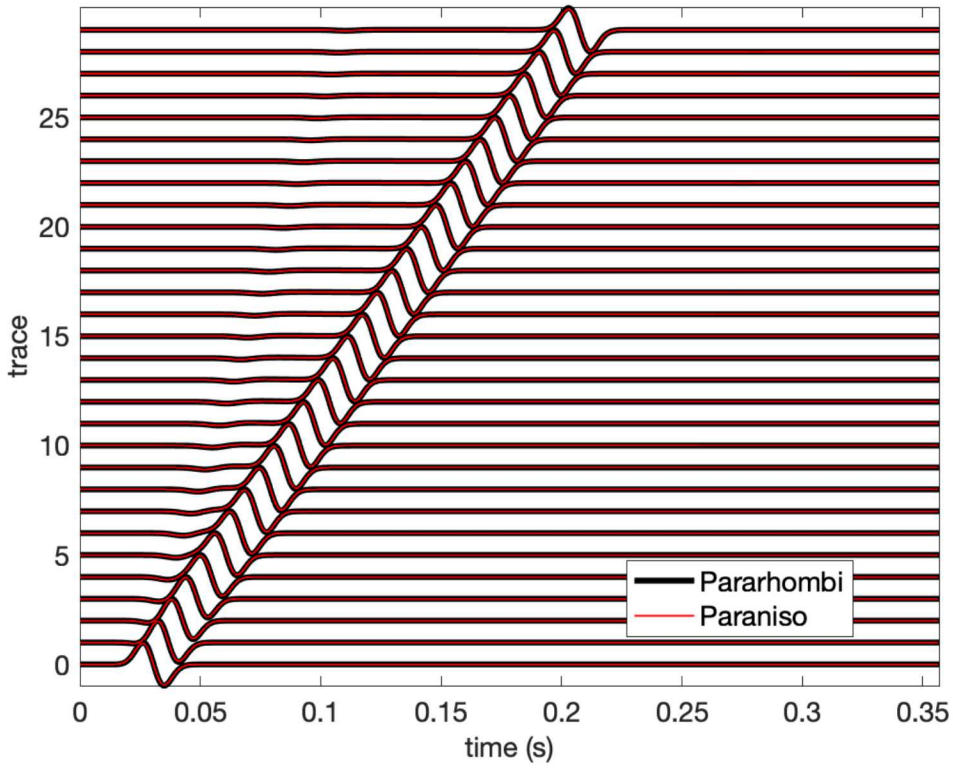

Figure 4.4: Comparison between Pararhombi and Paraniso for Vx receivers along the y-direction from the source. Trace normalized by Pararhombi amplitudes.

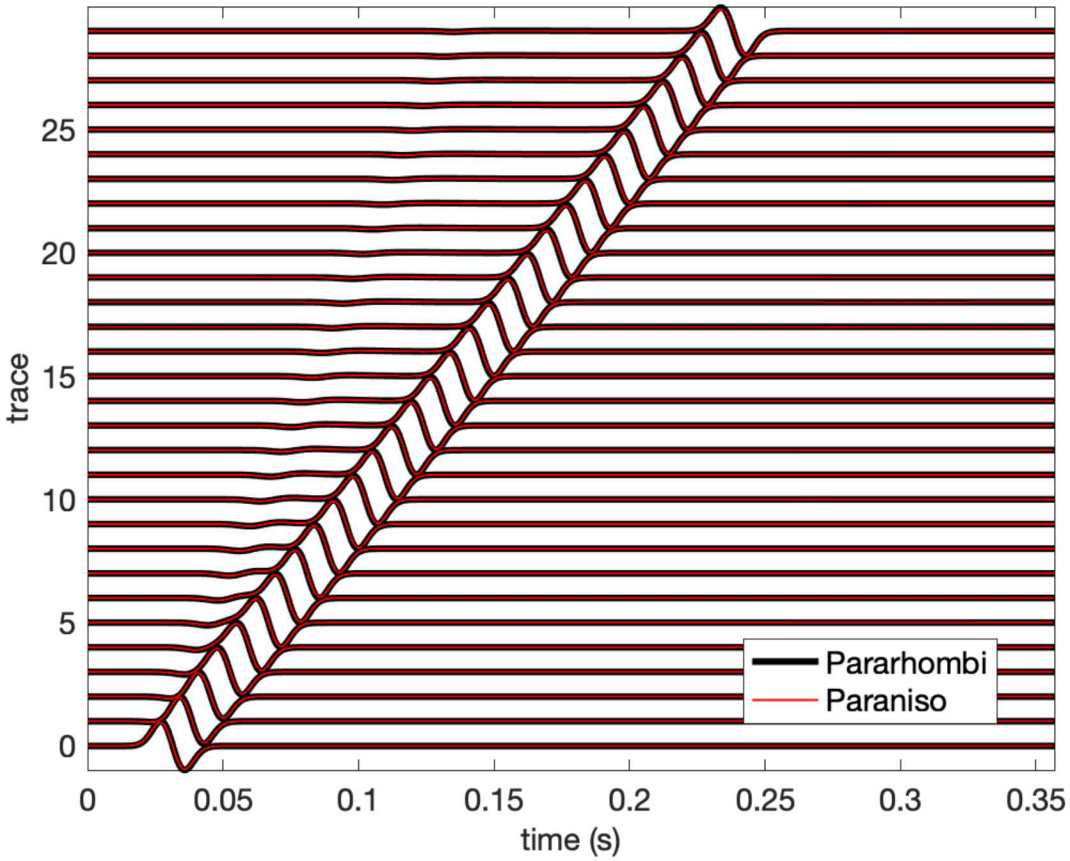

Figure 4.5: Comparison between Pararhombi and Paraniso for Vx receivers along the z-direction from the source. Trace normalized by Pararhombi amplitudes.

line is in the y-direction; the third is the same but in the z-direction. Figures 4.3-4.5 show the results of the comparison of Pararhombi and Paraniso. The comparison is excellent. Note the effects of anisotropy with obvious direction dependence of the arrival times for the main arrivals between Figures 4.4 and 4.5. The largest arrivals in both of these figures are quasi-shear waves, whereas the quasi-P-wave is the largest arrival in Figure 4.3 due to source radiation pattern effects.

5. SUMMARY

We have developed a new massively parallel 3-D finite-difference algorithm that solves the velocity-stress system of partial differential equations in a general elastic anisotropic medium. The equation of motion and constitutive relations were provided in Section 2 along with some discussion of the mathematical and physical constraints imposed on the relationship among the elastic moduli. A treatment of the stress-free surface boundary condition wrapped up Section 2. Section 3 focused on the implementation of the physical equations in a finite-difference scheme. Due to the nature of these equations in a generally anisotropic medium, a non-staggered finite-difference grid is used for the space domain, but a staggered time grid is used to update the velocities and stresses. We then explored the accuracy and stability issues associated with the non-staggered grid and finished the section describing the absorbing boundary conditions and massively parallel design. Finally, in Section 4 we validate the new algorithm by comparing results with known solutions from an isotropic solver and an orthorhombic case. The appendix of this report describes usage of Paraniso.

As is usual, Paraniso is not a static code base. It will continue to be updated and improved with new features. One of the main issues with any anisotropic algorithm is knowing what the elastic moduli are. Although anisotropic properties are known for many pure minerals, these are rarely available for large-scale models over large extents of the earth. Thus, accurate modeling on larger scales can be challenging. This prompts the need for better anisotropic 3-D characterization of the earth to reap the most benefit from these new tools.

REFERENCES

1. Aki, K., and P.G. Richards, *Quantitative Seismology*, Second Edition, University Science Books, Sausalito, CA, 2002.
2. Bennett, H.F., A Simple Seismic Model for Determining Principle Anisotropic Direction, *J. Geophys. Res.*, **77** (17), 3078-3080, 1972.
3. Berenger, J-P., A Perfectly Matched Layer for the Absorption of Electromagnetic Waves, *J. Comp. Phys.*, **114**, 185-200, 1994.
4. Haney, M.M., and D.F. Aldridge, *Numerical Dispersion for the Conventional-Staggered-Grid Finite-Difference Elastic Wave Propagation Algorithm*, SAND2008-4991, Sandia National Laboratories, Albuquerque, NM, July 2008.
5. Johnston, J.E., and N.I. Christensen, Seismic Anisotropy of Shales, *J. Geophys. Res.*, Solid Earth, **100**, B4, 5991-6003, doi:10.1029/95JB00031, 1995.
6. Komatitsch, D., and R. Martin, An Unsplit Convolutional Perfectly Matched Layer Improved at Grazing Incidence for the Seismic Wave Equation, *Geophysics*, **72** (5), SM155-SM167, doi 10.1190/1.2757586, 2007.
7. Meza-Fajardo, K.C., and A.S. Papageorgiou, A Nonconvolutional, Split-Field, Perfectly Matched Layer for Wave Propagation in Isotropic and Anisotropic Elastic Media: Stability Analysis, *Bull. Seis. Soc. Am.*, **98** (4), 1811-1836, doi: 10.1785/0120070223, 2008.
8. Musgrave, M.J.P., *Crystal Acoustics*, Holden Day, San Francisco, CA, 1970.
9. Preston, L., *Pararhombi: Parallel Implementation of 3-D Seismic Wave Propagation in Orthorhombic Media*, SAND2018-9477, Sandia National Laboratories, Albuquerque, NM, August 2018.
10. Tsvankin, I., Anisotropic Parameters and P-wave Velocity for Orthorhombic Media, *Geophysics*, **62** (4), 1292-1309, 1997.

A. PARANISO USAGE

This section describes the usage of Paraniso including the input earth model, receiver and source geometries, and provides the most important flags needed to run the algorithm.

A.1. Input Model File

The input model file is in NetCDF (https://www.unidata.ucar.edu/software/netcdf/) format. This binary format contains model dimension, elastic moduli, and density information along with all metadata associated with these data. NetCDF is a widely used format, with most common scripting and programming languages providing read and write capability, such as Matlab, python, C, C++, and Fortran. The input model file must contain either 1-D (in z) or 3-D point-by-point elastic moduli and density information. All 21 moduli given in Section 2 plus density must be in a single file, or, alternatively, each elastic modulus and density can be contained in 22 separate NetCDF files with a common base name. All input files must contain the dimensions:

NX: number of node points in the x-direction

NY: number of node points in the y-direction

NZ: number of node points in the z-direction

NT: number of time steps

numCoord: 4 (always: 3 space dimensions plus 1 time dimension)

Each file must also contain the following variables (the dimensions of each variable are given in parenthesis after the variable name):

minima: (numCoord) minimum values for each of the 3 spatial dimensions and time as [x0, y0, z0, t0]. Typically, these are all zeros.

increments: (numCoord) provides the node spacing and time step increment as [dx,dy,dz,dt].

x: (NX) x-axis values

y: (NY) y-axis values

z: (NZ) z-axis values

time: (NT) time axis values

For fully 3-D models the medium parameter variables are named: **cxxxx**, **cxxyy**, **cxxzz**, **cxxxy**, **cxxxz**, **cxxyz**, **cyyyy**, **cyyzz**, **cyyxy**, **cyyxz**, **cyyyz**, **czzzz**, **czzxy**, **czzxz**, **czzyz**, **cxyxy**, **cxyxz**, **cxyyz**, **cxzxz**, **cxzyz**, **cyzyz**, **rho**. Each variable has dimensions of (NZ,NY,NX). Again, these can be split into 22 files with a single base name ending with the variable name plus '.cdf', or all placed into a single input file. For 1-D in z models, considerable space can be saved by using 1-D arrays of size NZ for each of the medium parameters. In this case, the variable names begin with '**oneDModel**' plus the first letter capitalized version of the variable name, e.g., **oneDModelCxxxx**.

A.2. Receivers

The receivers can either be supplied on the command line for simple layouts or in a plain text file for more complicated geometries. The command line allows additions of single receivers or of a

uniform grid of receivers. Using a file allows completely arbitrary receiver placements for thousands of receivers, if desired. One important thing to note about receiver locations is that they should be placed 1 grid node below the local topography to achieve accurate results (if not using a free surface). This is due to the finite reach of the interpolation schemes used. See the `-Rl` flag below.

For the file method, simply define a flat text file with three columns: x, y, and z. Each line will be a new receiver and the x, y and z values will be in the model coordinate system. To include the file, add the following to the command line:

`-Rf3` ***`type filename.txt`***

where *type* is either "Pressure", "3C", "4C", "Velocity", "Vx", "Vy", "Vz", "Rx", "Ry", or "Rz" where 3C gives all three velocity components per receiver line and 4C also includes pressure. "R" types are rotation rate (curl of the velocity field about the specified axis) receivers. If the receiver type is "Velocity", then 3 additional columns must be appended for each receiver in the text file: bx, by, and bz, which specify the orientation of the receiver's positive axis as direction cosines.

To add individual receivers, add the following to the command line:

`-R` ***`type x y z`***

This adds one receiver of type (see above) at position x, y, z in model coordinates.

To add a uniform grid of receivers, add the following to the command line:

`-Rg` ***`type x0:dx:xf y0:dy:yf z0:dz:zf`***

This adds a uniform grid of receivers of type (see above) on the grid defined by the Matlab style vectors. For example, *x0:dx:xf* means x ranging from x0 to xf at an increment of dx (note that dx here is independent of the model dx).

Other receiver command line options that may be useful are:

`-Rl` : use trilinear interpolation instead of the default cubic interpolation for receiver points. This is important to do for receivers within about 2 grid nodes of the earth's surface because a cubic interpolator will reach above the surface to obtain interpolated values, whereas trilinear interpolation is more localized.

`-Ra` : make acceleration traces instead of the default velocity traces

`-Rd` : make displacement traces instead of the default velocity traces

`-Rt` ***`N`***: write out traces to the output file every *N* time steps. In case something catastrophic occurs, trace data will be saved up to that increment in time. By default, it only writes to the output file at the end of the simulation.

`-Ro` ***`traceOutputFile.cdf`*** : the trace output from the receivers will be output into this netCDF file. There are several dimensions and variables in this file, but we will discuss only those most pertinent to reading the file.

The 'numReceivers' dimension gives the number of receivers in the file. Note that this number is the total number of components and receivers, so, for example, if you added 100 3C receivers, numReceivers would equal 300.

The following are pertinent variables:

receiverX: (numReceivers) receiver X position

receiverY: (numReceivers) receiver Y position

receiverZ: (numReceivers) receiver Z position

receiverBx: (numReceivers) x-component of receiver, between 0 and 1.

receiverBy: (numReceivers) y-component of receiver.

receiverBz: (numReceivers) z-component of receiver.

Note that a Vx receiver will have receiverBx=1.0 and receiverBy and Bz equal to zero, while a Vz receiver will have receiverBz=1.0 and the others equal to zero. Of course, for pressure receivers, these variables will be equal to zero and are not used.

receiverType: (numReceivers) coded type of receiver. A pressure receiver will have a value of 2 here, whereas other types will have a 1.

receiverData: (numReceivers,NT) a 2-D array containing all of the trace data. Each row is a full timeseries. Output units are in mks units, so velocities are in m/s and pressures are in Pascals.

A.3. Sources

Sources are added via the command line. Both explosion and arbitrarily oriented force sources are available. First, a source time function must be defined. A Gaussian wavelet, Ricker wavelet (doubly differentiated gaussian), and delta function (spike) source time functions can be added with command line flags or the user can specify a wavelet in a file. A Gaussian or Ricker wavelet is nice for visualization since they are compact in both time and frequency. A delta function source makes visualization impossible, but the output is very flexible, since the output of one model run can then be convolved with any number of source time functions, instead of having to run a new model for each source time function.

To make a Gaussian wavelet, add the following to the command line:

`-Sg` *`Fpeak`*

where *Fpeak* is the peak frequency of the desired Gaussian wavelet. Note that the 1% level is about 2.25 times this peak frequency.

Similarly, to add a Ricker wavelet, add the following to the command line:

`-Sr` *`Fpeak`*

where *Fpeak* is the peak frequency of the desired Ricker wavelet. Note that the 1% level is about 3 times this peak frequency.

To add a delta function wavelet, add the following to the command line:

```
-SD 0
```

This adds a spike at time zero (first time sample). The output from a delta function wavelet is useless without convolution with a reasonable source time function. A reasonable source time function is one that has its once (force) or twice (explosion) differentiated waveform at 1% of the peak amplitude spectrum at or below the maximum frequency that the model was designed for. Note: make sure that the model dt is multiplied into the convolution to obtain accurate amplitudes.

To add an arbitrary waveform, add the following to the command line:

`-Sw` ***`filename.txt`***

filename.txt is a plain text file containing two columns: t and amp. t is the time starting at t0 with samples every dt. amp is the amplitude of the source time function at that time. The amplitudes of the source time function are (usually) normalized so amp varies between -1 and +1. The length of file should be <=NT. If the file length is < NT, the source time function will be padded with zeros out to NT samples. Just as for any source time function convolved with a delta function, this source time function must be reasonable. In the far field, the source time function will be once differentiated for a force source and twice differentiated for an explosion source. These far field wavelets should have their 1% of peak amplitude spectrum at or below the maximum frequency that the model was designed for. Too much higher frequency energy leads to large numerical dispersion and inaccurate results.

Now, the type and location of the source must be specified. To add an explosion source, add the following to the command line:

`-Se` ***`x y z amp`***

This adds an explosion source of amplitude amp (N-m) at position x, y, z.

To add a force type source, add the following to the command line:

`-Sfz` ***`x y z amp`***

This adds a vertical force source of amplitude amp (N) at position x, y, z. To specify an x-directed or y-directed for source use **`-Sfx`** or **`-Sfy`**, respectively. For an arbitrarily directed force source, use the command line flag

`-Sf` ***`x y z amp theta phi`***

where theta is the polar angle relative to the z-axis and phi is the azimuthal angle relative to the x-axis. Default angles are in degrees.

A general moment tensor can be specified with

`-Sm` ***`x y z amp mxx mxy mxz myx myy myz mzx mzy mzz`***

where *mxx*, etc., are the 9 components of an arbitrary moment tensor. Note that typically the moment tensor is symmetric (i.e., *mxy* = *myx*, etc.). Non-symmetric moment tensors imply torque sources.

A.4. Running Paraniso

Once the model, sources, and receivers are prepared, one is ready to run a simulation. Pananiso must be executed with mpirun or mpiexec. It cannot be run on its own even for single processor runs. The most important flags not already provided in the source or receiver sections above are:

modelFile.cdf : The model name is provided directly after the executable with no flag preceding it. If using 22 separate model files (one for each medium parameter), the argument will be *modelFileBase* without a '.cdf'.

`-p px py pz`: (required) this gives the domain decomposition of the model. There will be *px* processors in the x-direction, *py* in the y, and *pz* in the z. Note that NX/*px*, NY/*py*, and NZ/*pz* must all be > 8, i.e., there must be more than 8 grid nodes on each processor in each dimension. The code uses a master-slave node approach to decomposition, so the total number of processors requested for mpirun is *px***py***pz*+1 = np. To maximize cache efficiency, use $px \leq py \leq pz$.

`-T t0:dt:tf` : (optional) re-define the time vector for the simulation in Matlab vector notation

The following is the absorbing boundary conditions to damp unwanted reflections from the computational domain boundary and should be provided:

`-bpm n R alph k xfac`: multi-axial convolutional PML with a thickness of *n* nodes, with theoretical reflection coefficient *R*, corner frequency *alph*, stretch factor *k*, and cross factor *xfac*. *n* is typically 10; *R* should be 0.001 or less, *alph* should be pi*Fpeak (Fpeak is approximately the dominant frequency of the source waveform), *k* should be 1, and *xfac* should be initially chosen to be 0. If the model becomes unstable, then increase *xfac*; however, *xfac* should be as small as stability allows. See Section 3.4 for more information.

To activate the stress free boundary condition at the Zmin boundary, include the flag

`-bF`

No topography should be in the model file in this case. See Section 2.2 for information on this boundary condition.

For visualization purposes, slices can be made with the following flags:

`-En N type plane pos` : This will output *N* snapshots of *type* ground motion on the given *plane* at position *pos* evenly spaced in time. *type* can be "Pressure", "Vx", "Vy" or "Vz". *plane* can be "XY", "XZ" or "YZ". So, for example, -En 51 Pressure XZ 0, will output 51 snapshots of the pressure field on the XZ plane at y=0. Multiple **`-En`** lines are allowed per command line.

`-Eo sliceFile.cdf` : output the slices (snapshots) to the netCDF file *sliceFile.cdf*. All slices are stored in this file, so this file can become very large for big models with many snapshots. In order to split up output slices into multiple files based on slice type, provide the alternative

`-Eos sliceFileBase`

where *sliceFileBase* will form the base file name. Output files will be called *sliceFileBase* plus *plane* name plus *type*, e.g., sliceFileBasexzPressure.cdf.

Each slice is stored in an appropriately named variable in the file. The variable names are given as '*planeType*', so the variable named 'xzPressure' would refer to pressure on the xz plane. These variables are 3-D arrays of dimension (*N*,planeDim1,planeDim2), where *N* is the number of

slices, planeDim1 is the size of first of the plane dimensions and planeDim2 is the size of the second plane dimension. So, 'xzPressure' from the `-En` example above would have dimension (51,NZ,NX). Note that the ordering of the dimension sizes are the same as for the 3-D geophysical parameters. A second useful variable in the cdf file has the same name at the slice variable above, but with 'Time' appended. This variable is of length N and gives the time at which the snapshot was taken.

Output files (slices and/or receivers) are in netCDF format and can be read by most modern scripting and programming languages for visualization.

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Technical Library	01177	libref@sandia.gov

This page left blank

This page left blank

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.