Reduced-Precision Wave Propagation to Leverage AI Hardware in Seismic Modelling

Daniel Tompkins
Supervised by Professor Gerard Gorman and Dr. Edward Caunt

Department of Earth Science and Engineering, Imperial College London

Abstract

Modern seismic imaging techniques rely heavily on numerical wave equation solvers, which are typically implemented using finite-difference methods. These methods are memory-bound, making them well-suited for emerging computational hardware optimised for reduced-precision arithmetic, such as GPUs and TPUs designed for AI applications. However, directly solving wave equations in reduced-precision formats such as FP16 or BF16 can introduce severe numerical instabilities due to the restricted dynamic range and increased rounding errors. This study presents a novel, generalisable scaling method that transforms wave equations to fit within the representable range of reduced-precision formats, enabling stable and accurate numerical solutions. Unlike previous approaches that require equation-specific modifications, our method leverages dimensional analysis to redefine physical units, ensuring compatibility across a broad class of wave equations. We validate this approach using Devito-based simulations of the acoustic, tilted transversely isotropic (TTI), and elastic wave equations, demonstrating that scaled solutions remain within 0.002%–1.7% of their unscaled counterparts. Further, we investigate the impact of reduced precision on wave equation solvers using a 1D acoustic wave test case, systematically lowering the bit depth using arbitrary-precision arithmetic. Our results indicate that FP16 precision is viable for seismic modelling, while BF16 precision introduces excessive rounding error, and FP8 precision leads to numerical instability. These findings suggest that appropriately scaled wave equations can be solved efficiently on AI hardware using FP16, opening new possibilities for high-performance seismic imaging and broader applications in computational physics.

A report submitted in partial fulfillment of the requirements for the degree of Geophysics MSci at Imperial College London and Associateship of the Royal School of Mines. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed, provided that the source is explicitly acknowledged.

07 January 2025

This work contains 5476 words and was completed between July 2024 and January 2025, with UROP bursary funding from Devito Codes.
Generative AI tools were used to help in the production of figures and streamlining concepts.

Contents

1 Introduction

1.1 Partial Differential Equations and Finite-Difference Methods

Many physical systems can be described mathematically through partial differential equations (PDEs). These equations govern various natural and engineered phenomena, including fluid flow, modelled by the Navier-Stokes equations (Batchelor, 2000; Ershkov et al., 2021); wave propagation, described by seismic, acoustic, and electromagnetic wave equations (Aki and Richards, 2002; Kinsler et al., 2000; Griffiths, 2023); and financial modelling, where the Black-Scholes equation (Black and Scholes, 1973; Ankudinova and Ehrhardt, 2008) is used to price complex instruments. Efficiently solving PDEs is critical in numerous applications, enabling robust climate models, high-resolution subsurface imaging, and stable financial systems. These equations often lack analytical solutions or are too complex for real-world systems, requiring numerical methods.

Finite difference (FD) methods are widely used for numerically solving PDEs numerically due to their simplicity, efficiency, and low computational and memory requirements (Liu and Sen, 2009) compared to alternatives such as spectral, finite element, and finite volume methods (Virieux et al., 2011). FD methods replace continuous derivatives with approximations derived from a discretised grid of the physical domain. For example, the derivative of a function $u(x)$ at a point x_i can be approximated using the grid spacing Δx and neighbouring points x_{i+1} and x_{i-1}:

$$\left.\frac{du}{dx}\right|_{x=x_i} \approx \frac{u(x_{i+1}) - u(x_{i-1})}{2\Delta x} \tag{1}$$

Solving a PDE with an FD method replaces continuous derivatives with discretised approximations (Equation 1) to compute derivatives and update solutions locally.

FD methods are commonly memory-bound, with performance limited by data transfer rates rather than computational power (Micikevicius, 2009). This arises because FD methods rely on accessing grid values in memory repeatedly to update solutions, with simple arithmetic per data point. In contrast, a compute-bound system is constrained by the processor's speed, where performing calculations dominates the overall runtime (Hennessy and Patterson, 2011). Meanwhile, in communication-bound systems (e.g., MPI domain decomposition (Gropp et al., 1999; Smith, 1997; Bader and Zhu, 2024)) the primary bottleneck is the transfer of data between computational nodes, especially at subdomain boundaries. Understanding the memory-bound nature of FD methods is crucial as minimising memory requirements and optimising access patterns can substantially improve efficiency (Micikevicius, 2009). For example, reducing the working set (actively needed data) reduces memory bandwidth pressure, speeding up simulations (Micikevicius, 2009). This is particularly important in large-scale FD simulations, such as seismic modelling, where grids span millions of points, making memory access the key performance factor.

1.2 Seismic Modelling

Seismic modelling simulates seismic wave propagation through the Earth's interior, with key applications in seismic imaging, geohazard assessment, and resource exploration

(Warner et al., 2013; Symes et al., 2008). Imaging techniques such as reverse-time migration (RTM) and full-waveform inversion (FWI) improve subsurface characterisation by refining velocity models and enhancing resolution (Yilmaz, 2001; Virieux and Operto, 2009). RTM collapses wave energy to its source, improving geological interpretations (Zhou et al., 2018), while FWI iteratively matches synthetic and observed wavefields to better resolve subsurface structures (Semeniuk et al., 2017). These techniques are critical for locating hydrocarbons, assessing fault stability, and identifying CO_2 storage sites (Daramola et al., 2024; Hale, 2013; Papadopoulou et al., 2024).

Seismic wave propagation is governed by wave equations suited to different media. The acoustic wave equation describes fluids (Kozaczka and Grelowska, 2017), while the elastic wave equation captures both compressional and shear waves in solids (Virieux, 1986). In anisotropic media, where wave speeds vary by direction, anisotropic wave equations are required (Fletcher et al., 2009). FD schemes solve these equations efficiently, enabling high-resolution simulations for resource exploration, hazard assessment, and carbon storage monitoring (Mufti and Fou, 1989; Aochi et al., 2013; Jiang, 2011).

FD methods are well-suited for seismic modelling due to their efficiency, scalability, and compatibility with structured grids. Seismic surveys deploy sensors in regular grids (Stone, 1994), aligning naturally with FD discretisation. Large-scale seismic simulations, often spanning tens to hundreds of kilometres, require high-resolution grids to resolve subsurface details (Komatitsch and Tromp, 2002). FD methods, leveraging local stencils, efficiently handle such grids while maintaining computational feasibility.

1.3 Floating-Point Formats and Arithmetic

Floating-point representation is a method of encoding a broad range of real numbers using a finite number of bits on a computer (IEEE, 2008; Goldberg, 1991). It serves as the foundation for most scientific computing, enabling numerical calculations across various domains.

Floating-point formats must balance two key concepts: **dynamic range** and **precision**.

- **Dynamic range:** The range of values (maximum and minimum) that can be represented by a given floating-point format (IEEE, 2008).
- **Precision:** The number of distinct real numbers that can be represented within the dynamic range of a floating-point format. (IEEE, 2008).

Floating-point numbers can exceed their representable range, causing **overflow** (resulting in infinity) or **underflow** (truncating to zero). Lower precision formats, with fewer exponent bits, are more prone to this (Goldberg, 1991). The trade-off between dynamic range and precision is determined by the allocation of bits within the floating-point format. Since the total number of bits is finite, increasing the dynamic range reduces the precision, and vice versa. This balance is a fundamental design feature of any floating-point format (IEEE, 2008). An example illustrating the impact of limited precision on numerical accuracy is provided in Appendix A.

FP64 and FP32 are industry standards for scientific and engineering applications, such as seismic modelling, due to their high accuracy and broad dynamic range (Fabien-Ouellet, 2020). Lower precision formats, like FP16 and BF16, reduce memory usage and computational cost, making them ideal for machine learning tasks and memory-bound

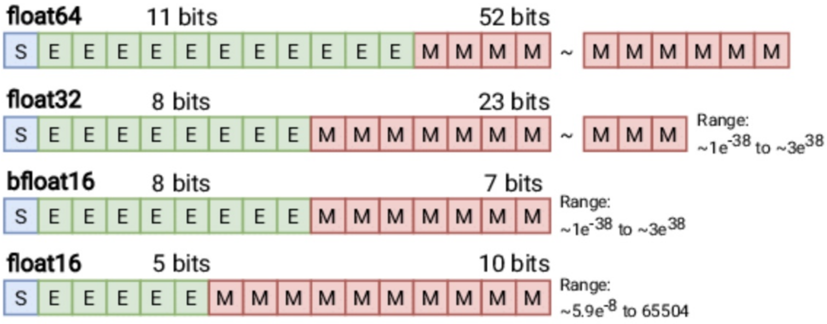
float64
11 bits
52 bits
S E E E E E E E E E E E M M M M ~ M M M M M M
float32
8 bits
23 bits
S E E E E E E E E M M M M M M M ~ M M M
Range:
~1e-38 to ~3e38
bfloat16
8 bits
7 bits
S E E E E E E E E M M M M M M M
Range:
~1e-38 to ~3e38
float16
5 bits
10 bits
S E E E E E M M M M M M M M M M
Range:
~5.9e-8 to 65504

Figure 1: Bit allocation for IEEE 754 standard FP64 (`float64`), FP32 (`float32`) and FP16 (`float16`) formats. These are referred to as double, single and half precision respectively (IEEE, 2008). BF16 (`bfloat16`), developed by Google Brain Cloud (2019), is also shown. Floating-point numbers are represented using three components: the sign bit (blue), which determines the number's positivity or negativity; the exponent (green), which controls the dynamic range; and the mantissa (red), which defines the precision of the number. Adapted from Haridas et al. (2022) and IEEE (2008).

computations (Carilli and Casper, 2021). See Figure 1 for the composition of common floating-point formats. Emerging formats like FP8 push this trade-off further, offering even lower precision to improve efficiency in deep learning applications (Micikevicius et al., 2022). FP8's limited precision restricts its use in accuracy-critical numerical methods, whereas FP64 and FP32 remain standard for high-precision applications like seismic simulations. In contrast, FP16, BF16, and FP8 are optimised for performance-critical, memory-constrained tasks such as machine learning.

1.4 Modern Hardware Development

Modern computational hardware is increasingly being developed to accommodate applications in artificial intelligence (AI) (Mojahidul Ahsan et al., 2024; Sentieys and Menard, 2022). AI applications are inherently resilient to noise and minor inaccuracies (Mojahidul Ahsan et al., 2024), allowing them to function effectively even with reduced-precision arithmetic. Exploiting this property, AI accelerators often use reduced-precision formats, trading a small degree of accuracy for substantial improvements in computational efficiency and memory usage (Mojahidul Ahsan et al., 2024).

CPUs, GPUs and TPUs (Armoni, 2024; Rodriguez and Bardos, 2024) typically support FP16 and BF16 floating-point formats in their architecture allowing for mixed precision arithmetic during computations (Sentieys and Menard, 2022). By reducing the number of bits used to represent numbers, lower precision formats significantly decrease the memory footprint of computations, enabling larger datasets and models to fit within fast-access memory.

FD methods are highly parallelisable due to their independent grid-based calculations, making them well-suited to GPUs, which optimise floating-point operations through dedicated processing cores like CUDA or Tensor Cores (Sun et al., 2022). Additionally, GPUs offer higher memory bandwidth, reducing bottlenecks in memory-bound applications, and enabling throughput gains of up to 12x compared to CPUs (Adams et al., 2007). The computational efficiency and memory advantages of modern hardware optimised for reduced-precision formats present significant opportunities for seismic modelling. While traditionally reliant on FP64 or FP32 for accuracy, seismic simulations that leverage reduced-precision formats could exploit the high throughput and memory bandwidth of AI hardware, enabling faster and more efficient large-scale FD computations.

1.5 Reduced-Precision in Seismic Modelling

Lower precision floating-point formats present two main challenges in seismic modelling:

- **Narrow dynamic range:** Shorter exponents increase susceptibility to overflow and underflow.
- **Rounding errors:** Reduced mantissa precision leads to higher rounding errors (IEEE, 2008).

These issues necessitate modifications to ensure numerical stability and accuracy (Gao, 2023).

Despite these challenges, reduced precision has been successfully applied in seismic modelling. Fabien-Ouellet (2020) implemented FP16 arithmetic in the 2D elastic wave equation (P-SV system), while Wan et al. (2024) extended this to the 3D elastic wave equation on curvilinear grids. Both studies achieved 1.7–2× speedups while maintaining numerical stability. Furthermore, Fabien-Ouellet (2020) showed that FP16 computations did not degrade FWI or RTM, demonstrating viability in imaging workflows.

To ensure values remained within FP16's dynamic range, Fabien-Ouellet (2020) applied logarithmic scaling to the right-hand side of the update equation. Wan et al. (2024) further introduced an additional scaling factor to maintain consistent orders of magnitude across variables, reducing rounding errors and improving numerical stability.

While reduced precision in seismic modelling has shown promise, existing approaches remain limited in scalability and generalisability. Fabien-Ouellet (2020) and Wan et al. (2024) tailored their methods to coupled first-order systems, relying on scaling parameters linked to specific wave equation properties. This dependence makes extension to broader PDEs challenging. Logarithmic scaling, though effective, lacks physical intuition for simpler wave systems. Additionally, Wan et al. (2024)'s extra scaling factor requires careful calibration to maintain numerical consistency, limiting automation and adaptability in large-scale simulations.

Building on these works, it is of interest to develop a method that generalises across a wider range of wave equations and mitigates reduced precision limitations without requiring extensive preconditioning or domain-specific adjustments.

1.6 Project Aims

This project aims to investigate approaches for solving wave equations in reduced precision and analyse the resulting impacts through two phases of investigation:

Scaling Wave Equations to Reduced Precision Formats

- **What:** Develop a method to scale wave equations such that their numerical representation fits within the dynamic range of reduced-precision floating-point formats.

- **Why:** Direct computation of wave equations in reduced precision often leads to errors due to overflow or underflow. Scaling ensures stability and accuracy by aligning values in the update equation to the representable range of the chosen floating-point format.
- **How:** We develop a methodology based on transforming the physical units of the system, independent of specific behaviours or inherent characteristics. This approach is validated by implementing scaled and unscaled cases of common equations in seismic modelling using the Devito DSL.

Assessing the Impact of Reduced Precision

- **What:** After scaling to the dynamic range of lower precision floating-point formats, evaluate the behaviour of wave equations as precision is incrementally reduced, focusing on solution accuracy and numerical stability.
- **Why:** Evaluating the limits of reduced precision is essential to determine the suitability of reduced precision formats for seismic modelling applications.
- **How:** We develop a 1D acoustic wave equation test case as a benchmark and compare the analytical solution to a numerical solution calculated via FD. Using the MPMath library in Python, we simulate reduced precision and quantify the effects on solution accuracy and error propagation across varying bit depths.

2 Introduction to Method

We first address the challenge of scaling wave equations for reduced-precision formats, aiming for a method applicable to all wave equations and PDEs. To validate this, we apply it to three seismic modelling equations:

- **Acoustic Wave Equation:** The acoustic wave equation describes the propagation of compressional waves, playing a central role in exploration seismology and underpinning numerous seismic processing techniques (Sotelo et al., 2021; Kosloff and Baysal, 1983).
- **Tilted Transversely Isotropic Wave Equation:** The tilted transversely isotropic (TTI) wave equation, as outlined in (Fletcher et al., 2009), approximates anisotropic elastic wave propagation without shear phases. This equation is particularly relevant in subsurface environments, such as those with tilted, layered geological formations like shales (Thomsen, 1986).
- **Elastic Wave Equation:** The elastic wave equation describes the propagation of both compressional and shear waves in solid media. These equations are fundamental to the modelling of the entire seismic wavefield and the development of wave propagation models for earthquake dynamics (Virieux, 1986; Madariaga, 1976).

We use the Python package Devito (Louboutin et al., 2019; Luporini et al., 2020) to implement both unscaled and scaled versions of the equations outlined above, comparing the resulting wavefields quantitatively and qualitatively. Devito is a domain-specific language that facilitates specification of FD models using high-level symbolic Python objects. These objects define FD operators, which are then translated into highly optimised

C++ code at runtime through a multi-stage compilation process. For each implementation, we use a $2\,\text{km} \times 2\,\text{km}$ grid, discretised into 201 points in each direction. We inject a Ricker wavelet source (Ricker, 1953; Hao et al., 2024) with a frequency of 30Hz into the centre of the grid, and set the simulation length at 0.25s

After validating the scaling method for adapting wave equations to the dynamic range of lower precision floating-point formats, we investigate the impact of reduced precision on the accuracy of wave equation solutions. A test case is devised using the 1D acoustic wave equation, with a numerical solver implemented in Python using a FD scheme. To simulate an incremental reduction in precision, from FP32 equivalent to FP8 equivalent, we use the Python package MPMath (mpmath development team, 2023). MPMath enables arbitrary-precision arithmetic by specifying significant digits and representing numbers as extended precision data types. By gradually reducing mantissa bits, we assess rounding and truncation effects, evaluating reduced-precision formats for seismic modelling.

3 Scaling Wave Equations to Dynamic Range

3.1 Introduction to Scaling

Wave equations often involve physical parameters that span large and small magnitudes, leading to overflow or underflow issues when using reduced precision formats. To illustrate this, consider the 1D acoustic wave equation:

$$\frac{\partial^2 u(x,t)}{\partial t^2} = c^2 \frac{\partial^2 u(x,t)}{\partial x^2} + s(x,t), \tag{2}$$

where $u(x,t)$ is the displacement field, c is the wave speed, $s(x,t)$ is a source term, x is the spatial coordinate, and t is time.

The FD stencil for Equation 2 using second-order approximations is:

$$\frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{\Delta t^2} = c^2 \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} + s(x,t), \tag{3}$$

where u_i^n represents the displacement at spatial index i and time step n, Δt is the time step size, Δx is the spatial step size, and c is the wave speed. Rearranging for u_i^{n+1}, the update equation becomes:

$$u_i^{n+1} = 2u_i^n - u_i^{n-1} + \frac{c^2 \Delta t^2}{\Delta x^2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) + \Delta t^2 s(x,t). \tag{4}$$

For a reasonable wave speed of $c = 3000\,\text{m/s}$, squaring this as described in Equation 4 results in $9,000,000\,\text{m/s}$, far exceeding $65,504$, the maximum representable value of FP16. Similarly, a time step size of $\Delta t = 0.003\,\text{s}$ squared yields $0.000009\,\text{s}$, which is significantly smaller than 0.000061, the minimum representable value of FP16. Naively solving Equation 2 numerically using Equation 4 in half precision would lead to overflows and underflows at every time step.

To address these challenges, we employ a dimensional analysis approach akin to selecting alternative units, such as km/s instead of m/s. The goal is to define units so that the physical parameters of the problem are approximately ~ 1, avoiding the risks of overflow and underflow. Unlike the methods proposed by Fabien-Ouellet (2020) and Wan et al. (2024), our approach works directly with the problem's physical units rather than introducing complex mathematical transformations.

By focusing on base physical units, this method can be extended to a wide range of computational physics problems involving measurable physical parameters. It avoids dependence on specific characteristics of an equation, such as coupling, making it broadly applicable to any problem governed by physical units.

3.2 General Methodology

We propose a method for defining new units for wave equations as shown in Figure 2.

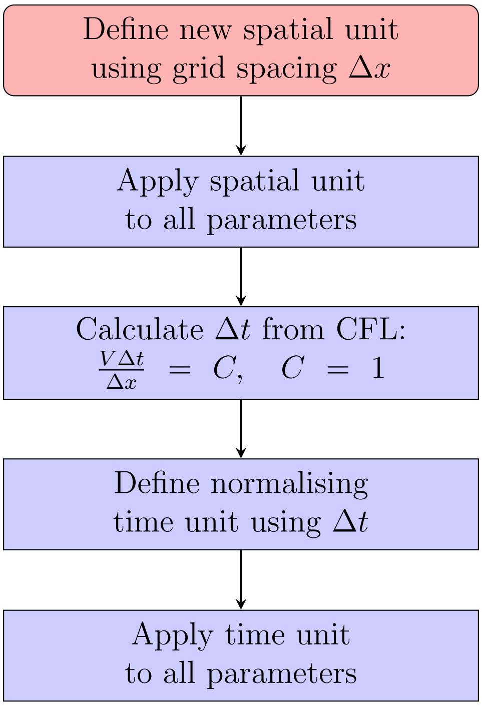
Define new spatial unit using grid spacing Δx
Apply spatial unit to all parameters
Calculate Δt from CFL: $\frac{V\Delta t}{\Delta x} = C, \quad C = 1$
Define normalising time unit using Δt
Apply time unit to all parameters

Figure 2: Flowchart of the scaling process. The Courant-Friedrichs-Lewy (CFL) condition (Courant et al., 1967; De Moura and Kubrusly, 2013) ensures numerical stability in finite-difference time stepping. Here, we determine Δt using the CFL condition with Courant number $C = 1$, ensuring $\frac{V\Delta t}{\Delta x} = 1$. See Appendix B for a detailed derivation of the method. To illustrate this general method in practice, consider an area discretised with a grid spacing Δx of 10 m and a constant velocity model of 3000 m/s throughout. The spatial unit is redefined so that the new grid spacing becomes 1 (10 m), and the maximum velocity V is rescaled accordingly to 300 (10 m/s). The time step is determined using this velocity and grid spacing in the CFL condition as described in Figure 2, yielding a value of $\frac{1}{300}$ s. This is subsequently used to define a new normalising time unit of $\frac{1}{300}$ s and thus our time step Δt becomes $1(\frac{1}{300}\,\text{s})$. Applying this time unit to all time-dependent parameters ensures that the normalised maximum velocity becomes $1\,(10\,\text{m}/\frac{1}{300}\,\text{s})$. The frequency of 30 Hz is adjusted to 0.1 Hz, and the total simulation time is scaled up by a factor of 300. By expressing physical parameters in this rescaled form, operations such as squaring a velocity or time step remain within the numerical dynamic range.

3.3 Application to Isotropic Acoustic Wave Equation and Results

The 2D acoustic wave equation is given by:

$$\frac{\partial^2 P(x,z,t)}{\partial t^2} = c^2(\frac{\partial^2 P(x,z,t)}{\partial x^2} + \frac{\partial^2 P(x,z,t)}{\partial z^2}) + s(x,z,t), \tag{5}$$

where $P(x,z,t)$ is the pressure field, c is the wave speed, $s(x,z,t)$ is a source term, x and z are spatial coordinates and t is time.

Discretising the temporal and spatial derivatives in Equation 5 with a 2nd-order FD scheme gives:

$$\frac{P_{i,j}^{n+1} - 2P_{i,j}^{n} + P_{i,j}^{n-1}}{\Delta t^2} = c^2 \left(\frac{P_{i+1,j}^{n} - 2P_{i,j}^{n} + P_{i-1,j}^{n}}{\Delta x^2} + \frac{P_{i,j+1}^{n} - 2P_{i,j}^{n} + P_{i,j-1}^{n}}{\Delta z^2} \right) + s(x,z,t) \tag{6}$$

To reach our final update equation, we rearrange Equation 6 for $P_{i,j}^{n+1}$. This yields an expression for the pressure at the next time step given by:

$$P_{i,j}^{n+1} = 2P_{i,j}^{n} - P_{i,j}^{n-1} + \Delta t^2 c^2 (\frac{P_{i+1,j}^{n} - 2P_{i,j}^{n} + P_{i-1,j}^{n}}{\Delta x^2} + \frac{P_{i,j+1}^{n} - 2P_{i,j}^{n} + P_{i,j-1}^{n}}{\Delta z^2}) + \Delta t^2 s(x,z,t). \tag{7}$$

We use Equation 7 to solve Equation 5 for the pressure field $P(x,z,t)$ using a fourth-order discretisation scheme in Devito. With the change of units, the $\Delta t^2 s(x,z,t)$ term is normalised to 1, as the peak amplitude of our Ricker wavelet is 1. We implement both a constant and variable velocity case, with the variable model used shown in Appendix C. The results of the scaling procedure are presented in Figure 3, Figure 4, and Table 1.

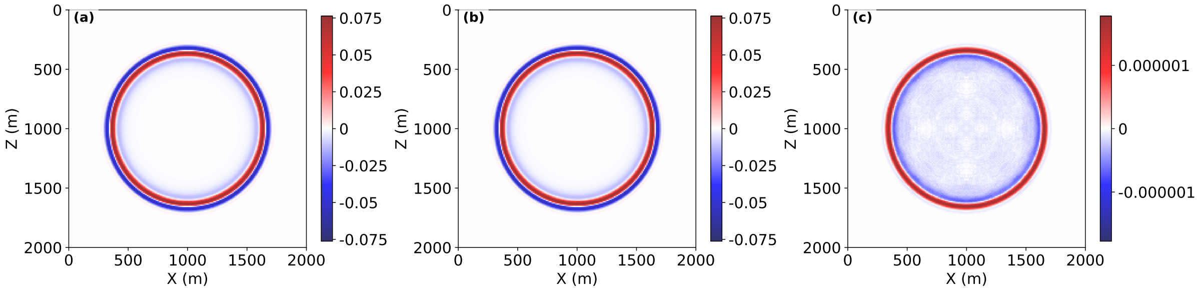

Figure 3: Plot **(a)** shows the baseline unscaled pressure field given by Equation 7 for a constant velocity model with $V_{max} = 3000\text{m/s}$. Plot **(b)** shows the pressure field produced by Equation 7 with our scaling method applied, note that **(a)** and **(b)** are both visually and numerically indistinguishable. Plot **(c)** shows the difference between the two wavefields at the end of the simulation.

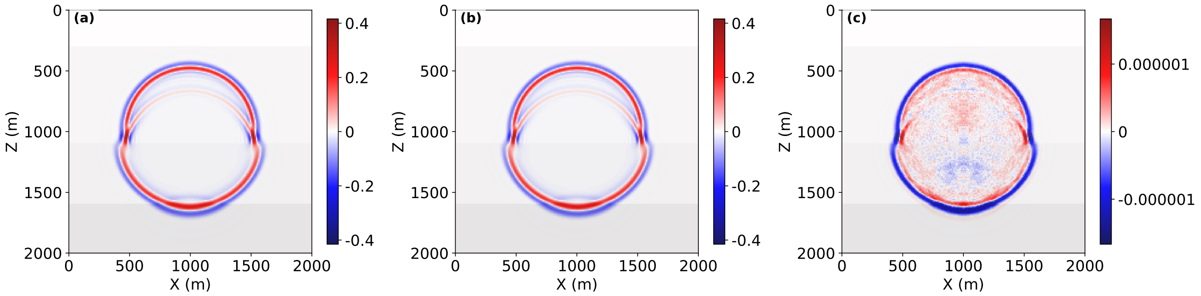

Figure 4: Plot **(a)** shows the baseline unscaled pressure field given by Equation 7 for a layered velocity model as shown in Appendix C. Plot **(b)** shows the scaled pressure field for the same velocity model, note that again the fields are both visually and numerically indistinguishable. Plot **(c)** shows the difference between the two wavefields at the end of the simulation.

Velocity Case	% Change in Max Abs	% Change in Range
Constant Velocity	$-5.276\,734 \times 10^{-4}$	$1.826\,799 \times 10^{-3}$
Variable Velocity	$-1.291\,285 \times 10^{-4}$	$4.442\,672 \times 10^{-4}$

Table 1: We calculate percentage changes in the maximum absolute value and range of values of the acoustic pressure field to quantitatively evaluate our scaling method. The constant velocity row shows the percentage changes observed in Figure 3, and the variable velocity row displays the results observed in Figure 4. Percentage changes are generally on the order of 10^{-3} to 10^{-4}, meaning that the first 3 to 4 decimal digits of the unscaled and scaled maximum absolute and range of values remained the same.

3.4 Application to Tilted Transversely Isotropic Wave Equation and Results

Fletcher et al. (2009) use a P-SV TTI dispersion relation to derive a coupled system of equations to describe wave propagation an anisotropic medium:

$$\begin{aligned} \frac{\partial^2 P}{\partial t^2} &= v_{px}^2 H_2 P + \alpha v_{pz}^2 H_1 Q + v_{sz}^2 H_1 (P - \alpha Q), \\ \frac{\partial^2 Q}{\partial t^2} &= \frac{v_{pn}^2}{\alpha} H_2 P + v_{pz}^2 H_1 Q - v_{sz}^2 H_2 \left(\frac{1}{\alpha} P - Q \right), \end{aligned} \tag{8}$$

where P is the pressure field, Q is an auxillary field, v_{pz} is the P wave velocity in the direction normal to the symmetry plane, v_{pn} is the P-wave normal moveout (NMO) velocity relative to the symmetry plane given by $v_{pn} = v_{pz}\sqrt{1 + 2\delta}$, v_{px} is the P-wave velocity in the symmetry plane given by $v_{px} = v_{pz}\sqrt{1 + 2\epsilon}$, v_{sz} is the SV velocity normal to the symmetry plane, δ and ϵ are dimensionless anisotropy parameters defined by Thomsen (1986), α is a non-zero scalar and H_1 and H_2 are derivative operators given by:

$$\begin{aligned} H_1 &= \sin^2\theta \frac{\partial^2}{\partial x^2} + \cos^2\theta \frac{\partial^2}{\partial z^2} + \sin 2\theta \frac{\partial^2}{\partial x \partial z}, \\ H_2 &= \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2} - H_1. \end{aligned} \tag{9}$$

Due to the complexity of Equation 8, we leverage the capabilities of Devito to define the update equations for the pressure and auxillary fields as:

$$
\begin{aligned}
P_{i,j}^{n+1} = (\Delta t)^2 \Bigg(& \frac{2P_{i,j}^{n} - P_{i,j}^{n-1}}{\Delta t^2} \\
& + 7.5\frac{P_{i+1,j}^{n} - 2P_{i,j}^{n} + P_{i-1,j}^{n}}{\Delta x^2} \\
& + 7.5\frac{P_{i,j+1}^{n} - 2P_{i,j}^{n} + P_{i,j-1}^{n}}{\Delta z^2} \\
& + 3.66\frac{Q_{i+1,j}^{n} - 2Q_{i,j}^{n} + Q_{i-1,j}^{n}}{\Delta x^2} \\
& + 3.66\frac{Q_{i,j+1}^{n} - 2Q_{i,j}^{n} + Q_{i,j-1}^{n}}{\Delta z^2} \\
& - 11.64\frac{P_{i+1,j+1}^{n} - P_{i+1,j-1}^{n} - P_{i-1,j+1}^{n} + P_{i-1,j-1}^{n}}{4\Delta x\Delta z} \\
& + 7.32\frac{Q_{i+1,j+1}^{n} - Q_{i+1,j-1}^{n} - Q_{i-1,j+1}^{n} + Q_{i-1,j-1}^{n}}{4\Delta x\Delta z} \Bigg),
\end{aligned}
\tag{10}
$$

$$
\begin{aligned}
Q_{i,j}^{n+1} = (\Delta t)^2 \Bigg(& \frac{2Q_{i,j}^{n} - Q_{i,j}^{n-1}}{\Delta t^2} \\
& + 4.56\frac{P_{i+1,j}^{n} - 2P_{i,j}^{n} + P_{i-1,j}^{n}}{\Delta x^2} \\
& + 4.56\frac{P_{i,j+1}^{n} - 2P_{i,j}^{n} + P_{i,j-1}^{n}}{\Delta z^2} \\
& + 5.34\frac{Q_{i+1,j}^{n} - 2Q_{i,j}^{n} + Q_{i-1,j}^{n}}{\Delta x^2} \\
& + 5.34\frac{Q_{i,j+1}^{n} - 2Q_{i,j}^{n} + Q_{i,j-1}^{n}}{\Delta z^2} \\
& - 9.12\frac{P_{i+1,j+1}^{n} - P_{i+1,j-1}^{n} - P_{i-1,j+1}^{n} + P_{i-1,j-1}^{n}}{4\Delta x\Delta z} \\
& + 7.32\frac{Q_{i+1,j+1}^{n} - Q_{i+1,j-1}^{n} - Q_{i-1,j+1}^{n} + Q_{i-1,j-1}^{n}}{4\Delta x\Delta z} \Bigg).
\end{aligned}
\tag{11}
$$

The system described in Equation 8 is solved using Equation 10 and Equation 11 using a eighth-order discretisation scheme. The results of the scaling procedure are presented in Figure 5, Figure 6, and Table 2.

Parameter Case	**% Change in Max Abs**	**% Change in Range**
Constant Parameter	$1.850\,217 \times 10^{-2}$	$1.695\,510$
Variable Parameter	$-1.073\,405 \times 10^{-4}$	$7.135\,709 \times 10^{-4}$

Table 2: Percentage changes in the maximum absolute value and range of values for the pressure field as solved for by Equation 10.

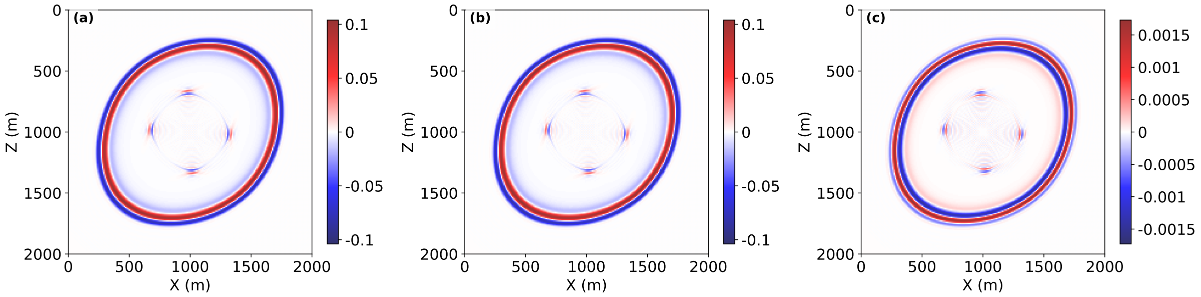

Figure 5: Plot **(a)** shows the pressure field as found by Equation 10 using unscaled parameters. Plot **(b)** shows the scaled pressure field, and Plot **(c)** shows the difference between the two. For this constant velocity case, we use parameters $V_p = 3000$m/s, $\epsilon = 0.24$, $\delta = 0.1$, $\alpha = 1$ and $\theta = \frac{\pi}{4}$ as outlined in Fletcher et al. (2009).

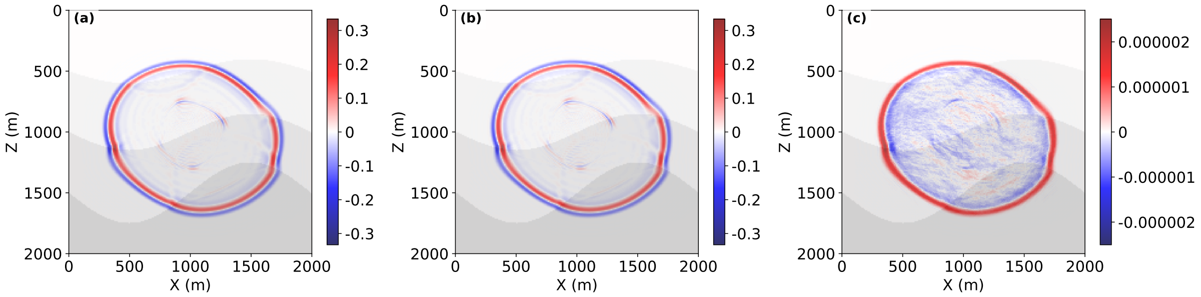

Figure 6: Plot **(a)** shows the pressure field as found by Equation 10 using an unscaled variable parameter model. Plot **(b)** shows the scaled pressure field through the same variable model, and Plot **(c)** shows the difference between the two. We hold α constant at 1 and implement the variable parameter model shown in Appendix C.

3.5 Application to Elastic Wave Equation and Results

Virieux (1986) outlines a velocity-stress formulation of the elastic wave equation to model P-SV wave propagation as follows:

$$\begin{aligned}
\frac{\partial v_x}{\partial t} &= b\left(\frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xz}}{\partial z}\right), \\
\frac{\partial v_z}{\partial t} &= b\left(\frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{zz}}{\partial z}\right), \\
\frac{\partial \tau_{xx}}{\partial t} &= (\lambda + 2\mu)\frac{\partial v_x}{\partial x} + \lambda\frac{\partial v_z}{\partial z}, \\
\frac{\partial \tau_{zz}}{\partial t} &= (\lambda + 2\mu)\frac{\partial v_z}{\partial z} + \lambda\frac{\partial v_x}{\partial x}, \\
\frac{\partial \tau_{xz}}{\partial t} &= \mu\left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x}\right),
\end{aligned} \tag{12}$$

Where $(v_x,\ v_z)$ is the velocity vector, $(\tau_{xx},\ \tau_{zz},\ \tau_{xz})$ are components of the stress tensor, b is the buoyancy defined as $\frac{1}{\rho}$ where ρ is density, and λ and μ are the Lame parameters (Aki and Richards, 2002). We can parameterise the Lame coefficients using the definitions of P and S wave velocity (Aki and Richards, 2002) leading to a new set of equations as follows:

$$
\begin{aligned}
\frac{\partial v_x}{\partial t} &= \frac{1}{\rho}\left(\frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xz}}{\partial z}\right), \\
\frac{\partial v_z}{\partial t} &= \frac{1}{\rho}\left(\frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{zz}}{\partial z}\right), \\
\frac{\partial \tau_{xx}}{\partial t} &= V_p^2 \rho \frac{\partial v_x}{\partial x} + \rho\left(V_p^2 - 2V_s^2\right)\frac{\partial v_z}{\partial z}, \\
\frac{\partial \tau_{zz}}{\partial t} &= V_p^2 \rho \frac{\partial v_z}{\partial z} + \rho\left(V_p^2 - 2V_s^2\right)\frac{\partial v_x}{\partial x}, \\
\frac{\partial \tau_{xz}}{\partial t} &= V_s^2 \rho \left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x}\right),
\end{aligned}
\tag{13}
$$

where V_p is the P wave velocity and V_s is the S wave velocity. Discretising and re-arranging the system described in Equation 13 leads to a set of update equations:

$$
\begin{aligned}
v_{x(i,j)}^{n+1} &= v_{x(i,j)}^{n-1} + \frac{2\Delta t}{\rho}\left(\frac{\tau_{xx(i+1,j)}^{n} - \tau_{xx(i-1,j)}^{n}}{2\Delta x} + \frac{\tau_{xz(i,j+1)}^{n} - \tau_{xz(i,j-1)}^{n}}{2\Delta z}\right), \\
v_{z(i,j)}^{n+1} &= v_{z(i,j)}^{n-1} + \frac{2\Delta t}{\rho}\left(\frac{\tau_{xz(i+1,j)}^{n} - \tau_{xz(i-1,j)}^{n}}{2\Delta x} + \frac{\tau_{zz(i,j+1)}^{n} - \tau_{zz(i,j-1)}^{n}}{2\Delta z}\right), \\
\tau_{xx(i,j)}^{n+1} &= \tau_{xx(i,j)}^{n-1} + 2\Delta t\left(V_p^2\rho(\frac{v_{x(i+1,j)}^{n} - v_{x(i-1,j)}^{n}}{2\Delta x}) + \rho(V_p^2 - V_s^2)(\frac{v_{z(i,j+1)}^{n} - v_{z(i,j-1)}^{n}}{2\Delta z})\right), \\
\tau_{zz(i,j)}^{n+1} &= \tau_{zz(i,j)}^{n-1} + 2\Delta t\left(V_p^2\rho(\frac{v_{z(i,j+1)}^{n} - v_{z(i,j-1)}^{n}}{2\Delta z}) + \rho(V_p^2 - V_s^2)(\frac{v_{x(i+1,j)}^{n} - v_{x(i-1,j)}^{n}}{2\Delta x})\right), \\
\tau_{xz(i,j)}^{n+1} &= \tau_{xz(i,j)}^{n-1} + 2\Delta t V_s^2 \rho\left(\frac{v_{x(i,j+1)}^{n} - v_{x(i,j-1)}^{n}}{2\Delta z} + \frac{v_{z(i+1,j)}^{n} - v_{z(i-1,j)}^{n}}{2\Delta x}\right).
\end{aligned}
\tag{14}
$$

We use a fourth-order discretisation scheme to solve Equation 14 using Devito, testing both constant and variable models for V_p, V_s and ρ.

Parameter Case	**Field**	**% Change in Max Abs**	**% Change in Range**
Constant	v_x	$6.200\,037 \times 10^{2}$	$6.200\,037 \times 10^{2}$
	v_z	$6.200\,009\,8 \times 10^{2}$	$6.200\,009\,8 \times 10^{2}$
	τ_{xx}	$1.682\,920\,1 \times 10^{-4}$	$3.099\,073\,6 \times 10^{-4}$
	τ_{zz}	$1.234\,141\,6 \times 10^{-4}$	$2.893\,035\,6 \times 10^{-4}$
	τ_{xz}	$1.590\,241\,0 \times 10^{-4}$	$2.697\,730\,2 \times 10^{-4}$
Variable	v_x	$1.099\,998 \times 10^{3}$	$1.099\,998 \times 10^{3}$
	v_z	$1.099\,997 \times 10^{3}$	$1.099\,997 \times 10^{3}$
	τ_{xx}	$-1.734\,003 \times 10^{-4}$	$3.318\,421 \times 10^{-4}$
	τ_{zz}	$-2.488\,594 \times 10^{-4}$	$3.694\,171 \times 10^{-4}$
	τ_{xz}	$-1.923\,861 \times 10^{-4}$	$3.773\,728 \times 10^{-4}$

Table 3: Percentage changes in the maximum absolute value and range of values for velocity and stress tensor components for constant and variable parameter fields.

The results of the scaling process are presented in Figure 7, Figure 8 and Table 3.

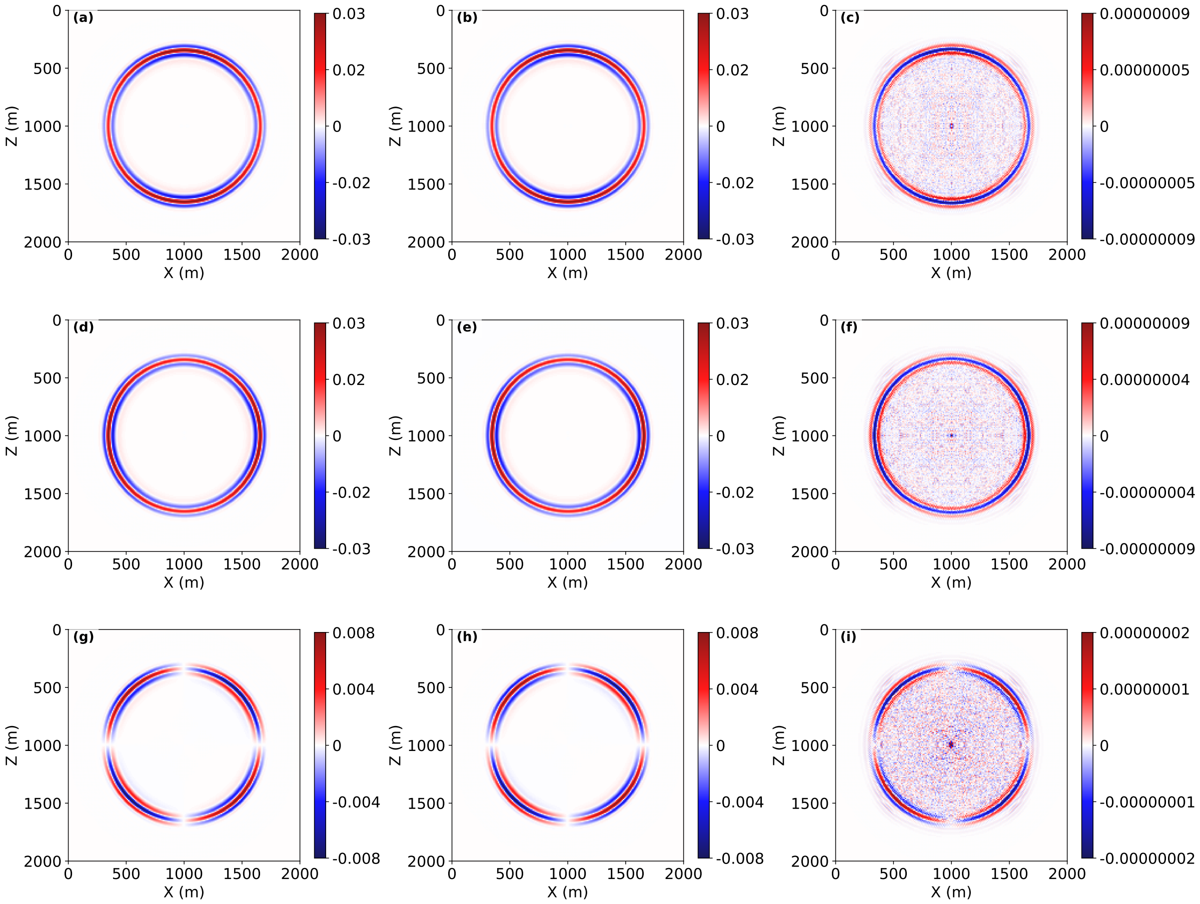

Figure 7: Normal and shear stress fields, found using Equation 14 through a constant parameter model, with $V_p = 3000\text{m/s}$, $V_s = 1500\text{m/s}$ and $\rho = 2400\text{kg/m}^3$. The top row shows the unscaled field for normal stress τ_{xx} in Plot **(a)**, the scaled field in Plot **(b)** and the difference between the two in Plot **(c)**. Plot **(d)** shows the unscaled field for normal stress τ_{zz}, Plot **(e)** shows the scaled field, and Plot **(f)** shows the difference. The bottom row displays the results of scaling on the shear stress τ_{xz}. Plots **(g)**, **(h)** and **(i)** show the unscaled stress, scaled stress and difference respectively.

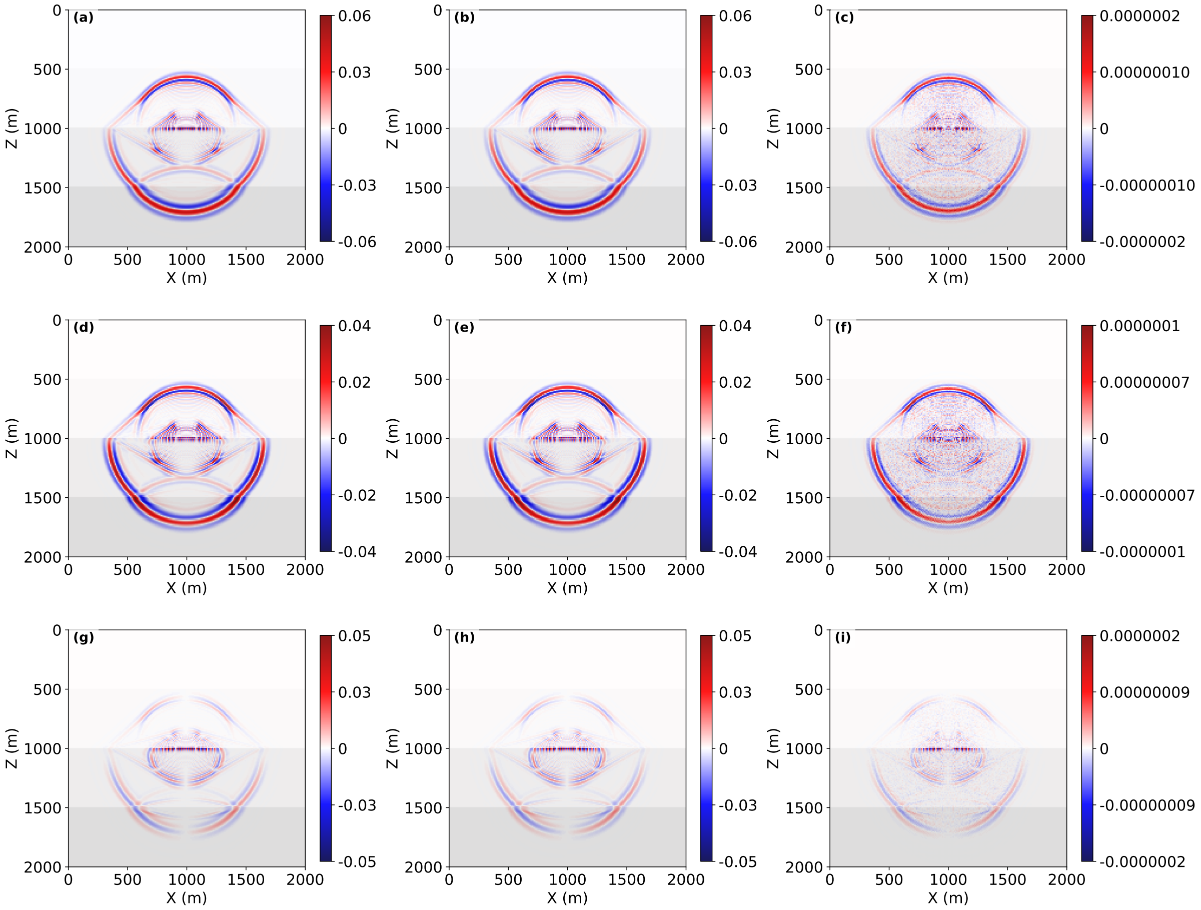

Figure 8: Normal and shear stress fields, found using Equation 14 through a variable parameter model detailed in Appendix C. The top row shows the unscaled field for normal stress τ_{xx} in Plot **(a)**, the scaled field in Plot **(b)** and the difference between the two in Plot **(c)**. Plot **(d)** shows the unscaled field for normal stress τ_{zz}, Plot **(e)** shows the scaled field, and Plot **(f)** shows the difference. The bottom row displays the results of scaling on the shear stress τ_{xz}. Plots **(g)**, **(h)** and **(i)** show the unscaled stress, scaled stress and difference respectively.

3.6 Discussion of Scaling and Merits of Approach

Our method successfully scales wave equations to fit within the dynamic range of lower precision formats. The resulting wavefields are visually indistinguishable from their unscaled counterparts. In the best case, the maximum absolute value of the scaled wavefield differs by approximately $-5 \times 10^{-5}\%$. Given FP32's 7 decimal digits of precision, the scaled and unscaled maximum absolute values match to six decimal places, diverging only in the seventh. For FP16 (4 decimal digits) and BF16 (2 decimal digits), this difference is numerically insignificant. In general, the scaled update equation produces values differing from the unscaled field by at most $10^{-2}\%$ to $10^{-4}\%$, ensuring numerical equivalence within the precision limits of FP16 and BF16.

A key result of the scaling process is its robustness across a range of physical parameters. We demonstrate its effectiveness for equations involving velocities, densities, angles, and anisotropic factors, highlighting potential applications beyond seismic modelling. While this work focuses on seismic wave equations, many of these parameters also appear in other PDEs. For example, velocity and density are central to the Navier-Stokes equations in fluid dynamics, the heat equation in convection modelling (Bluman and Cole, 1969; Recktenwald, 2004), and the Vlasov equation in plasma physics (Vlasov, 1968; Cheng and Knorr, 1976). If the method holds for these parameters in wave equations, it could extend to other linear PDEs. Unlike past approaches (Fabien-Ouellet, 2020; Wan et al., 2024) tailored to specific equations, this generalised method provides a scalable framework applicable across multiple domains, enhancing computational efficiency across disciplines.

Our method is shown to be effective for stress and pressure fields, while the velocity field undergoes a consistent change of units as part of the scaling process. The only case in which the scaled update equation produces a field that is materially different from the unscaled case is for the particle velocity fields in Virieux (1986)'s elastic wave equation formulation. While the velocity fields appear visually indistinguishable, numerical analysis reveals significantly higher values in the scaled case. This discrepancy arises because velocity is explicitly coupled with time, meaning that scaling the unit of time in the simulation directly scales the velocity field. Importantly, this transformation is consistent and systematically brings the velocity field closer to unity, making it a direct consequence of our method rather than a flaw. This has broader implications for other equations where parameters are coupled with time. For instance, similar scaling effects would be expected in other velocity-stress formulations of wave equations, as well as in PDEs that solve for velocity fields beyond seismic modelling.

Our method has key limitations affecting its general applicability to computational physics. Since it operates on physical units, it does not modify dimensionless parameters. In applying it to the TTI system of Fletcher et al. (2009), the anisotropic parameters δ and ϵ remain unchanged. For our case, with $\delta \sim 0.1$ and $\epsilon \sim 0.24$, this is not problematic, as they are constant, not exponentiated, and well within the dynamic range of lower precision formats. However, for larger or smaller dimensionless values, additional steps may be required to prevent overflow or underflow. Certain visco-acoustic and visco-elastic wave equations incorporate the quality factor Q_p (Li et al., 2017; Yang and Zhu, 2018), which can range from 10 to 1000 (de Castro Nunes et al., 2011; Dobrynina et al., 2011). If unscaled, extreme values may compromise numerical stability. For instance, Li et al. (2017) describe a visco-elastic wave equation with a $\frac{1}{Q_p^2}$ term, where $Q_p > 256$ causes Q_p^2 to exceed FP16's maximum representable value.

We neglect explicit scaling of angles, such as those defining tilt in TTI systems (Fletcher et al., 2009). While trigonometric functions of these angles remain unchanged, they are typically computed outside the main time loop in FD solvers. This enables mixed-precision arithmetic (Micikevicius et al., 2017; Baboulin et al., 2009) to retain high precision for precomputed values while using reduced precision in time stepping, reducing computational overhead.

Another limitation of our method is its reliance on the linearity of the scaling process. A function is linear if it satisfies additivity, $f(x_1 + x_2) = f(x_1) + f(x_2)$, and homogeneity, $f(kx) = kf(x)$ for a scalar k. By homogeneity, scaling a linear PDE by k scales its solution by the same factor, provided all terms—including field values and forcing terms—are scaled consistently. A unit transformation (e.g., 2 kPa → 2000 Pa) preserves the equation's structure, whereas arbitrarily scaling values (e.g., 2 kPa → 4 kPa) alters system behaviour. Additionally, achieving uniform scaling across all variables may not always be feasible, particularly in equations with multiple coupled quantities. Non-linear PDEs, which do not satisfy homogeneity, exhibit solutions that do not scale proportionally, limiting the direct applicability of our method to such cases.

4 Reduced-Precision Solutions

4.1 Introduction to Reduced-Precision Solutions

The number of mantissa bits a floating-point value has manifests itself in the number of significant decimal digits that values can be quoted to. Refer to Figure 1 for the allocation of bits across each component of the common floating-point formats.

Format	Mantissa Bits	Decimal Precision
FP64	52	15 digits
FP32	23	7 digits
FP16	10	4 digits
BF16	7	2–3 digits
FP8	4	1 digit

Table 4: Mantissa bits and approximate decimal precision of common floating-point formats.

Table 4 summarises the mantissa bits and corresponding approximate precision for common floating-point formats. The reduction in precision between floating-point types has implications for the accumulation of rounding error in computations. For example, consider the accumulation of rounding error when summing $N = 1,000,000$ values, each equal to 0.123456789:

True Value: The exact sum, without rounding, is:

$$\text{True Sum} = N \times 0.123456789 = 1,000,000 \times 0.123456789 = 123,456.789$$

FP32 Representation: The value 0.123456789 is stored as 0.1234568 in FP32, rounded to 7 significant digits. The total sum is:

$$\text{Sum}_{\text{FP32}} = N \times 0.1234568 = 1,000,000 \times 0.1234568 = 123,456.8$$

The rounding error is:

$$\text{Error}_{\text{FP32}} = \text{Sum}_{\text{FP32}} - \text{True Sum} = 123,456.8 - 123,456.789 = 0.011$$

FP16 Representation: The value 0.123456789 is stored as 0.1235 in FP16, rounded to 4 significant digits. The total sum is:

$$\text{Sum}_{\text{FP16}} = N \times 0.1235 = 1,000,000 \times 0.1235 = 123,500$$

The rounding error is:

$$\text{Error}_{\text{FP16}} = \text{Sum}_{\text{FP16}} - \text{True Sum} = 123,500 - 123,456.789 = 43.211$$

Implications: In FP32, the accumulated rounding error is minimal (0.011), whereas in FP16, the reduced precision introduces a significantly larger error (43.211). For a format like BF16, which offers only 2–3 decimal digits of precision, the rounding error would be even greater. While reduced precision formats provide substantial computational advantages, such as improved efficiency and lower memory requirements, they do so at the expense of increased numerical error. Understanding this trade-off is crucial for evaluating the suitability of formats like FP16 and BF16 in scientific computing applications.

4.2 Test Problem

To investigate the effects of reduced precision on wave equation solutions, we devise a test case using the 1D acoustic wave equation:

$$\frac{\partial^2 u(x,t)}{\partial t^2} = c^2 \frac{\partial^2 u(x,t)}{\partial x^2}. \tag{15}$$

This is similar to Equation 2, but the source term is omitted in Equation 15 as an initial condition for displacement is implemented instead.

We define the test case as follows:

$$\frac{\partial^2 u(x,t)}{\partial t^2} = c^2 \frac{\partial^2 u(x,t)}{\partial x^2}, \quad x \in [-L, L],\ t > 0,$$

where $u(x,t)$ is the displacement field, and c is the wave speed.

Boundary Conditions: The solution satisfies homogeneous Dirichlet boundary conditions:

$$u(-L,t) = 0, \quad u(L,t) = 0, \quad t > 0.$$

Initial Conditions: The initial displacement and velocity conditions are:

$$u(x,0) = \sin\left(\frac{2\pi x}{L}\right), \quad u_t(x,0) = 0, \quad x \in [-L, L].$$

The solution is derived using the method of separation of variables. Assuming $u(x,t) = X(x)T(t)$, the wave equation separates into two ordinary differential equations (ODEs):

$$\frac{T''(t)}{c^2 T(t)} = \frac{X''(x)}{X(x)} = -k^2,$$

where k is a separation constant.

The spatial ODE is:

$$X''(x) + k^2 X(x) = 0.$$

With the boundary conditions $u(-L,t) = 0$ and $u(L,t) = 0$, the solution is:

$$X_n(x) = \sin\left(\frac{n\pi x}{L}\right), \quad k = \frac{n\pi}{L},\ n = 1, 2, 3, \ldots$$

The temporal ODE is:

$$T''(t) + \left(\frac{n\pi c}{L}\right)^2 T(t) = 0.$$

The general solution is:

$$T_n(t) = C_n \cos\left(\frac{n\pi c}{L}t\right) + D_n \sin\left(\frac{n\pi c}{L}t\right),$$

where C_n and D_n are constants determined by the initial conditions.

Combining the spatial and temporal solutions, the general solution is:

$$u(x,t) = \sum_{n=1}^{\infty} \left[C_n \cos\left(\frac{n\pi c}{L}t\right) + D_n \sin\left(\frac{n\pi c}{L}t\right)\right] \sin\left(\frac{n\pi x}{L}\right).$$

The initial conditions are used to determine the coefficients C_n and D_n:

- From $u(x,0) = \sin\left(\frac{2\pi x}{L}\right)$, we find $C_n = 1$ for $n = 2$, and $C_n = 0$ for $n \neq 2$.
- From $u_t(x,0) = 0$, we find $D_n = 0$ for all n.

Thus, the analytical solution is:

$$u(x,t) = \cos\left(\frac{2\pi ct}{L}\right) \sin\left(\frac{2\pi x}{L}\right). \tag{16}$$

The analytical solution in Equation 16 describes a sinusoidal standing wave propagating along the x-axis with a speed c m/s. To simplify verification of the results, we select c and L such that the period of the solution is 1 s. Specifically, we set the half-length $L = 1000$ m and the wave speed $c = 1000$ m/s to achieve this.

4.3 Implementation

To solve Equation 15 numerically, we implement a forward-Euler (Biswas et al., 2013; Estep, 2002) leapfrog time-stepping scheme. Using a 2nd-order central difference approximation, we discretise the equation as:

$$\frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{\Delta t^2} = c^2 \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}, \tag{17}$$

where u_i^n is the numerical solution for displacement $u(x,t)$ at time index n and spatial index i, with time step Δt and grid spacing Δx. Rearranging for u_i^{n+1} yields:

$$u_i^{n+1} = 2u_i^n - u_i^{n-1} + \frac{c^2 \Delta t^2}{\Delta x^2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right). \tag{18}$$

We implement a loop in Python to solve Equation 18 for a given simulation length using the MPMath package to simulate varying precision levels.

MPMath facilitates arbitrary precision arithmetic, allowing control over mantissa bits to simulate reduced precision. In our implementation, all numerical operations updating the solution use MPMath arithmetic, with parameters Δt, Δx, and c cast as MPMath objects before simulation. This prevents overflow and underflow by using unlimited exponent bits, ensuring values remain within dynamic range.

We solve both scaled and unscaled cases of the 1D acoustic wave problem, comparing solutions to test the scaling method and identify discrepancies as precision decreases. Precision is reduced incrementally from 23 mantissa bits (FP32 equivalent) to 4 bits (FP8 equivalent). At each bit depth, we record the displacement field after 5 and 10 periods, comparing it to the analytical solution, Equation 16. Additionally, we track error over time and compute the Fourier transform of the wavefield at the beginning and end of the simulation to analyse frequency content.

4.4 Results

One key result of implementing both a scaled and unscaled case of our test problem was that the results for both were indistinguishable at every bit depth. In this section, we will only show the results of the unscaled case as this offers the purest insight into the behaviour of our test as precision is reduced.

We show numerical solutions, error through time and frequency spectra for levels of precision equivalent to FP32, FP16, BF16 and FP8, based on the number of mantissa bits as described in Table 4.

4.4.1 Solutions

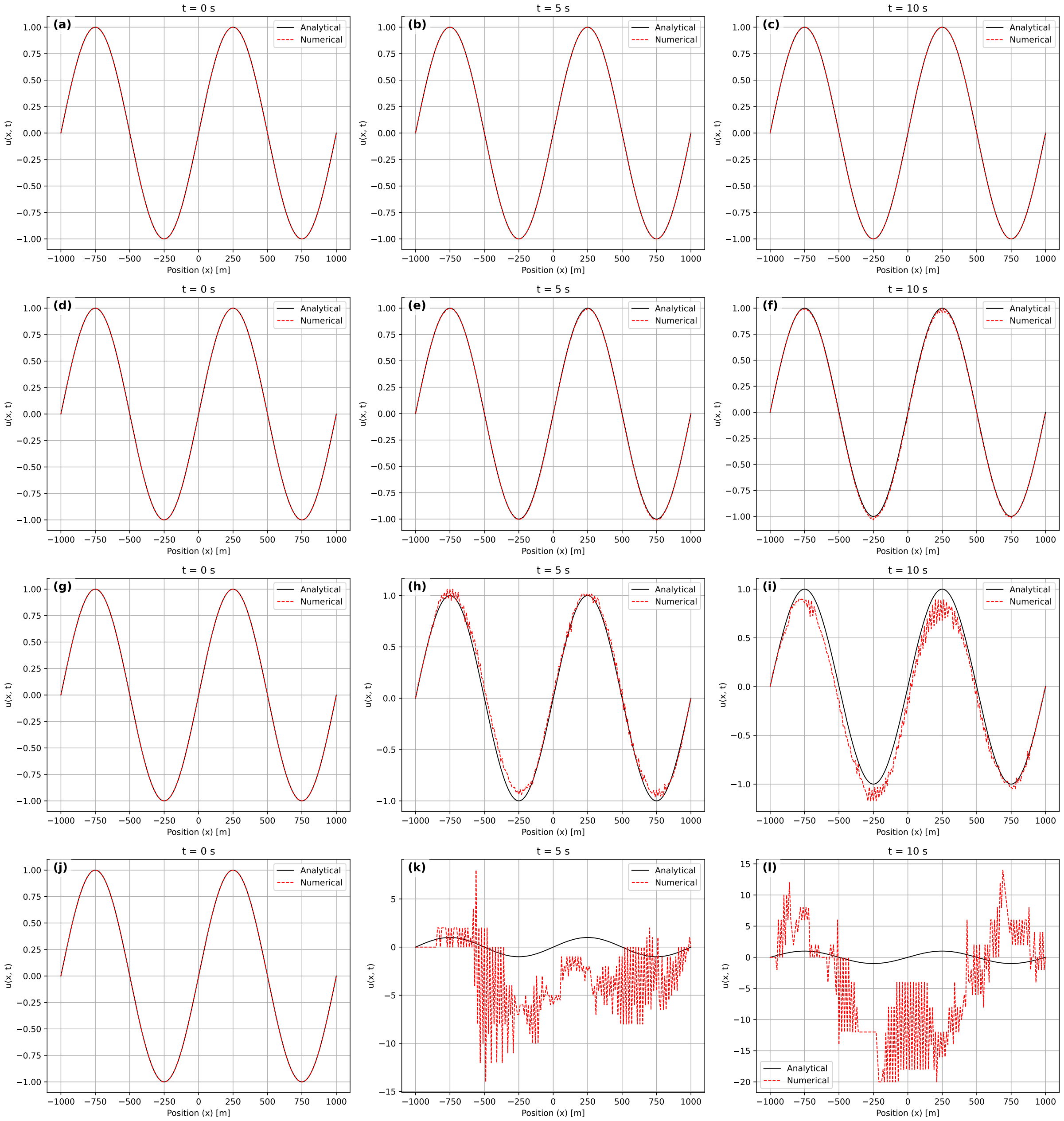

Figure 9: We show the numerical solution for our test case found using Equation 18, plotted with our analytical solution given by Equation 16. In Plots **(a)**, **(b)** and **(c)**, we show the numerical solution for 23 mantissa bits (FP32 equivalent). Note that at 23 bits, our numerical solution is indistinguishable to the analytical solution. In Plots **(d)**, **(e)** and **(f)**, we show the numerical solution for 10 mantissa bits (FP16 equivalent). Some distortion is first visible in the peaks and troughs of the wave after 10 periods ($t = 10$s). In Plots **(g)**, **(h)** and **(i)**, we show the numerical solution for 7 mantissa bits (BF16 equivalent). We observe signficant distortion in the wavefield here with significant high frequency artefacts visible after 5 and 10 periods. In Plots **(j)**, **(k)** and **(l)**, we show the numerical solution for 4 mantissa bits (FP8 equivalent). At this level of precision, the solution is unstable, reaching values well outside the range of ±1 where sinsuoidal functions are bound.

4.4.2 Error Through Time

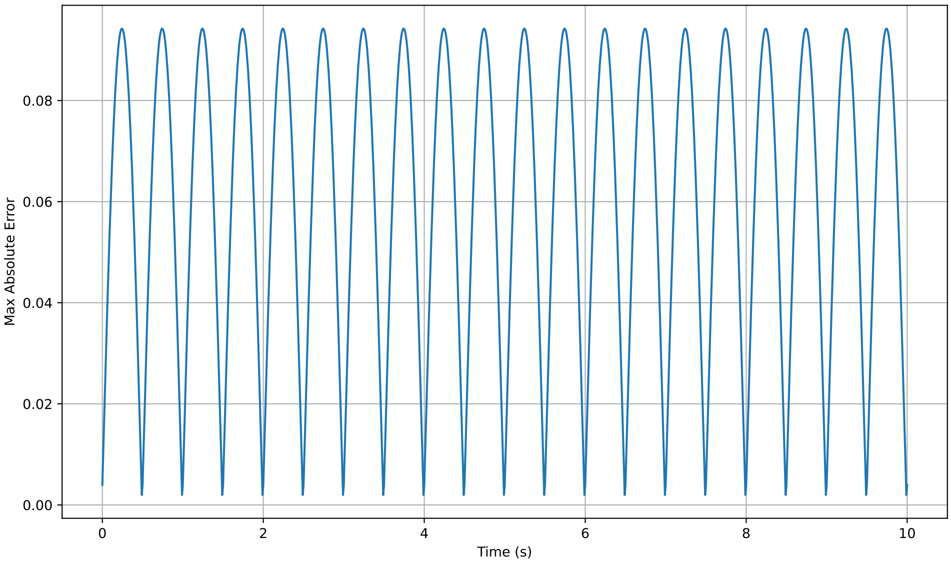

Figure 10: Maximum absolute error through time for 23 mantissa bits (FP32 equivalent). Note here that the profile is periodic, with the error being cancelled over time and stable.

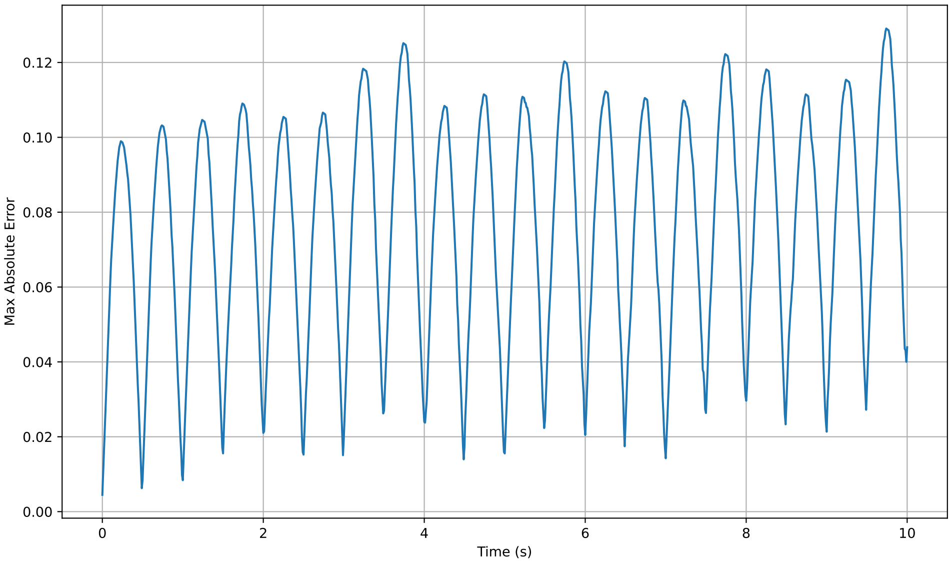

Figure 11: Maximum absolute error through time for 10 mantissa bits (FP16 equivalent). At this point, a zero-offset in the error is clear, with the profile behaving as a sinusoid with an added constant component that introduces a non-zero amplitude at zero frequency.

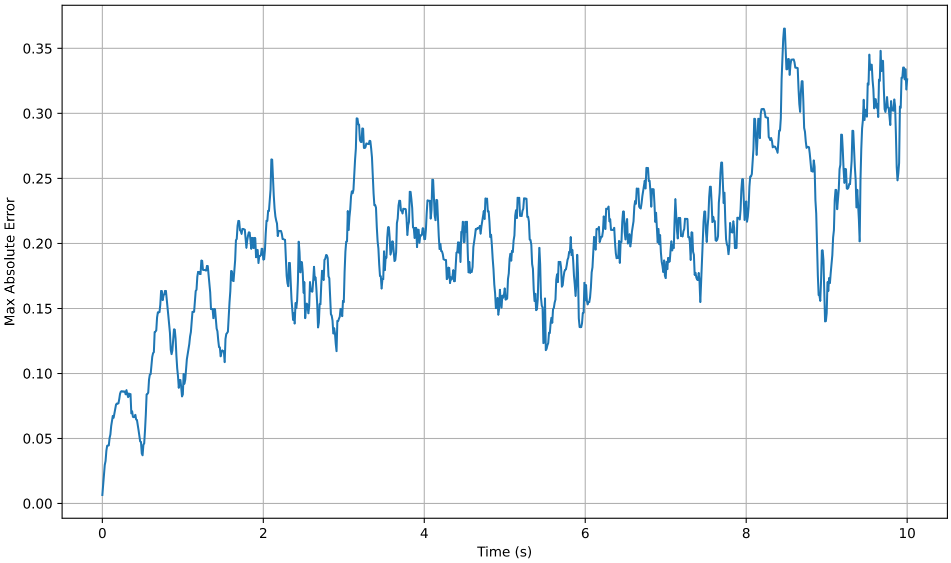

Figure 12: Maximum absolute error through time for 7 mantissa bits (BF16 equivalent). The profile has lost the periodicity that was still visible at 10 mantissa bits and is now beginning to shift towards an exponential accumulation of error. It is worth nothing the magnitude of error at this precision level, it is 3 times that of FP16.

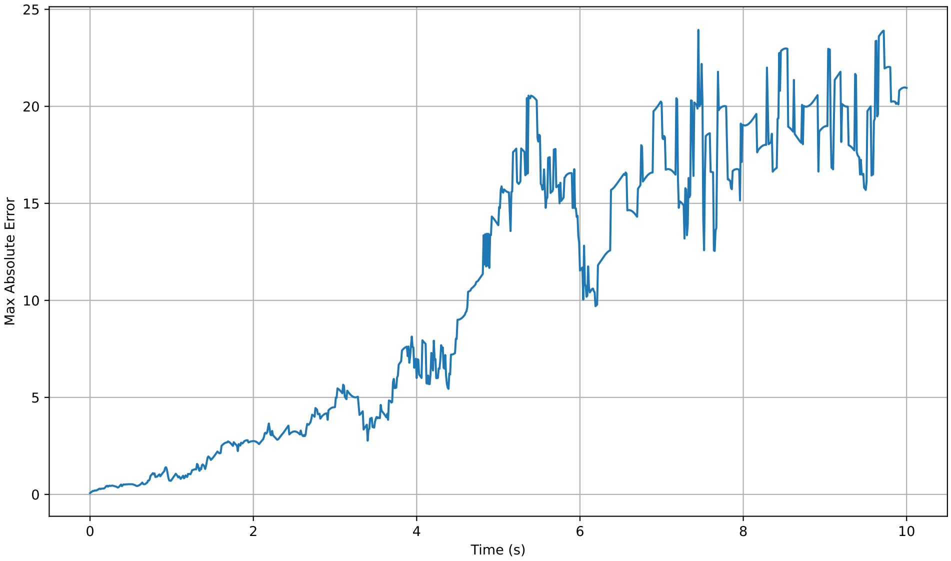

Figure 13: Maximum absolute error through time for 4 mantissa bits (FP8 equivalent). At this level of precision, the solution is unstable and this is reflected in the behaviour of the error. The curve is broadly exponential with error values two orders of magnitude higher than that of FP16 and three higher than FP32.

4.4.3 Frequency Spectra

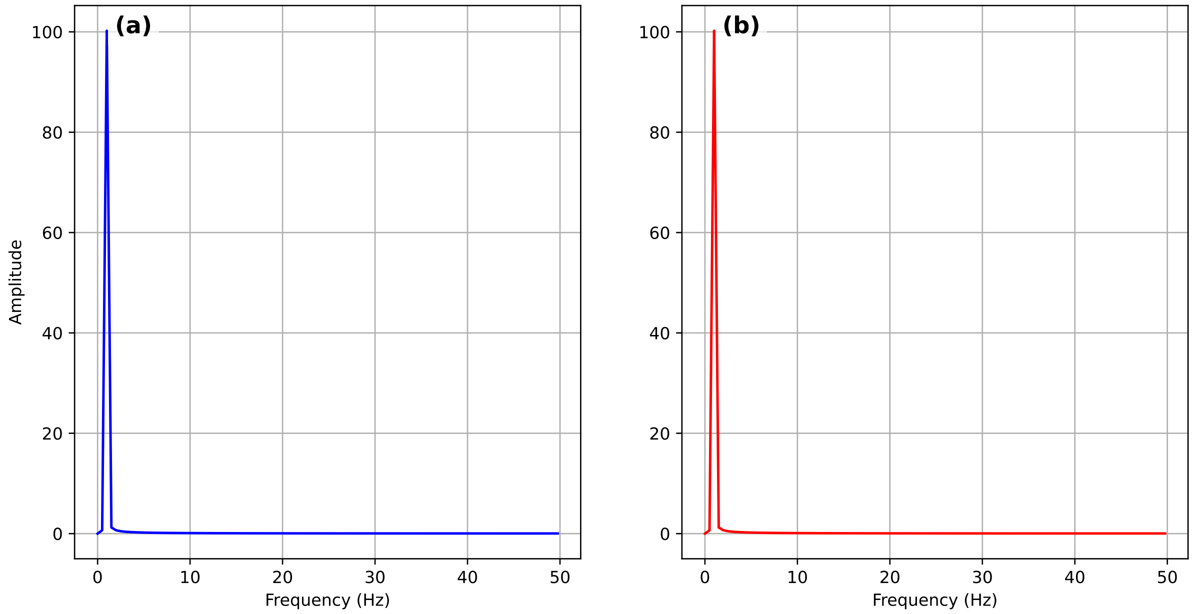

Figure 14: Frequency spectrum for 23 mantissa bits (FP32 equivalent). Plot **(a)** shows the result of a Fourier Transform of the wavefield at the beginning of the simulation. In Plot **(b)** we show the frequency spectrum at the end of the simulation.

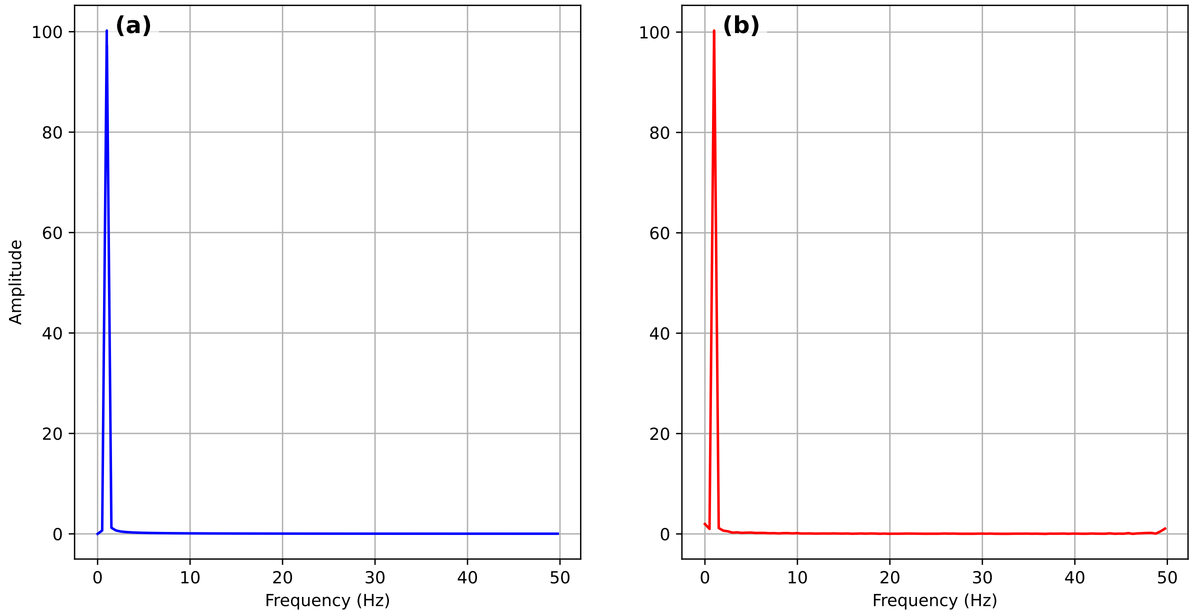

Figure 15: Frequency spectrum for 10 mantissa bits (FP16 equivalent). Plot **(a)** shows the spectrum at the beginning of the simulation. In Plot **(b)** we show the frequency spectrum at the end of the simulation. Note the small upticks at the extremes of the frequency spectrum. The high frequency artefact is of particular significance as it opposes the dominant frequency of the system, indicating this introduction of noise is a result of the precision reduction.

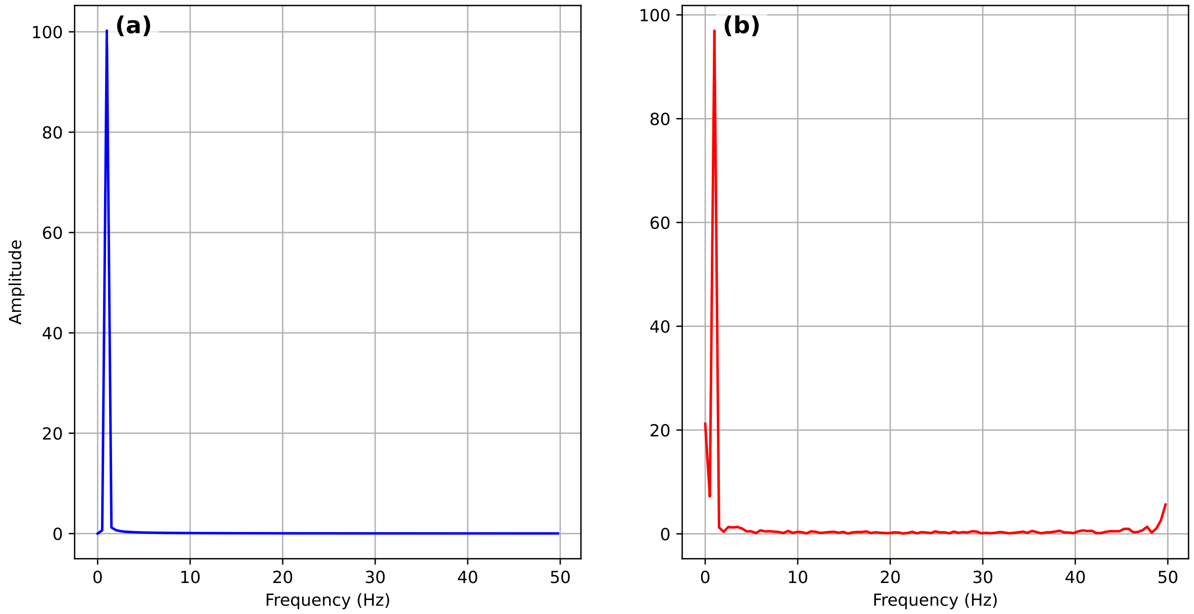

Figure 16: Frequency spectrum for 7 mantissa bits (BF16 equivalent). Plot **(a)** shows the spectrum at the beginning of the simulation. In Plot **(b)** we show the frequency spectrum at the end of the simulation. At this bit depth we observe a loss of the main mode of frequency, with the amplitude dropping during the simulation. We see additional harmonics propagating through the spectrum and significant high and low frequency noise.

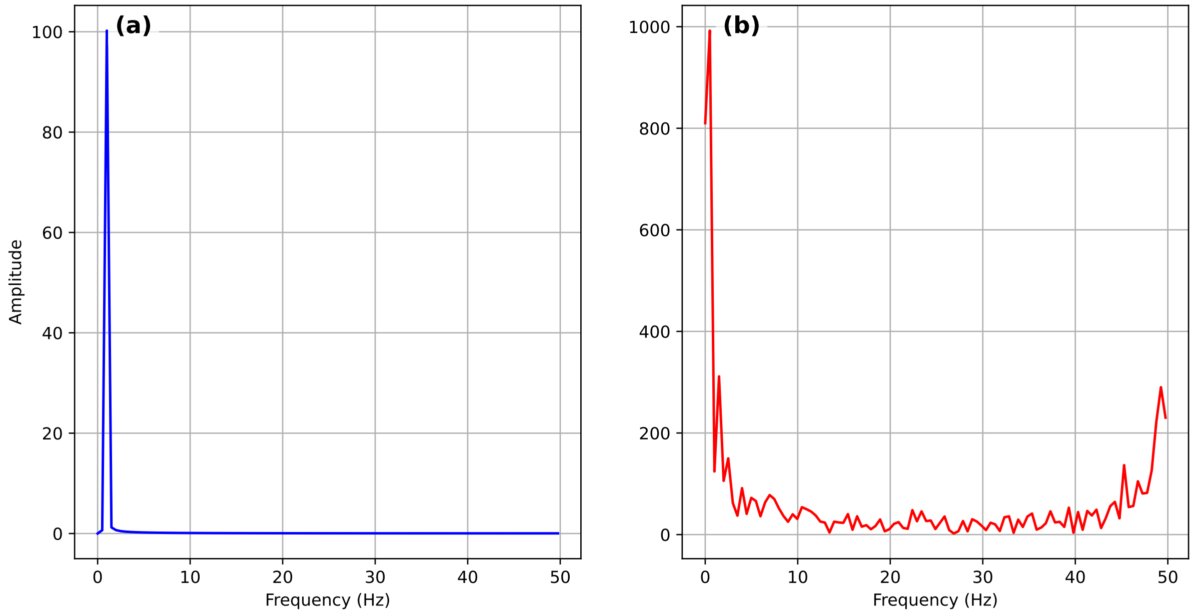

Figure 17: Frequency spectrum for 4 mantissa bits (FP8 equivalent). Plot **(a)** shows the spectrum at the beginning of the simulation. In Plot **(b)** we show the frequency spectrum at the end of the simulation. We observe many spurious harmonics in this frequency spectrum, but the key result here is that the loss of stability is present in the frequency domain. While before, our maximum amplitude was 100, in Plot **(b)** it is as high as 1000, indicating a breakdown of the solution's stability.

4.5 Discussion of Reduced-Precision Results

Stable solutions are observed at precision levels equivalent to FP32, FP16, and BF16, with accuracy degrading as precision decreases. At FP32, numerical and analytical solutions are visually indistinguishable. At FP16, minor amplitude distortions emerge but remain within an acceptable error range. At BF16 and lower, significant high-frequency artifacts appear, particularly at later times, increasing maximum absolute error by 287.6% relative to FP32. At FP8 precision, the numerical solution becomes unstable. As precision decreases, we note the loss of the main frequency mode, accompanied by the emergence of high- and low-frequency noise. Below FP16 equivalent, additional harmonics become prominent in the frequency spectra. Furthermore, for precision levels below BF16, the Courant number used in the FD scheme begins to influence the error accumulation over time, effectively introducing an additional stability condition on the simulation. To clarify, subsequent discussion of bits refers to the number of mantissa bits only.

The first signs of distortion in our numerical solution appear at 10 mantissa bits as shown in Figure 9. The distortion propagates from the peaks and troughs through the limbs of the wave as precision is reduced further. At 7 mantissa bits, distortion is evident across all parts of the wave after both 5 and 10 periods. The distortion observed becomes increasingly random over time, dominated by high-frequency regions. Unlike coherent high-frequency signals from seismic sources, which convey valuable subsurface information, the noise in our simulations is non-physical and spatially random. This noise can mask subtle geological features, degrade the signal-to-noise ratio, and interfere with migration and inversion algorithms. Moreover, in FD solvers, high-frequency noise contributes to numerical dispersion by introducing phase velocity errors that distort wave propagation, further compromising seismic imaging accuracy (Tam and Webb, 1993).

The error through time for each simulation increases as the number of mantissa bits is decreased, as expected. The key result, however, is the evolution of the error from the periodic curve shown in Figure 10 to the exponential shape of Figure 13. The error first begins to acquire a zero-offset, as shown in Figure 18.

The distinct zero-offset observed at 10 mantissa bits signifies the point at which the reduced-precision solution starts deviating from the analytical baseline, with errors no longer balancing over time. This has significant implications for the numerical modelling of wave equations that naturally oscillate about zero, such as those governing seismic and acoustic wave propagation. A persistent offset introduces an unphysical bias, distorting wave propagation and potentially leading to long-term stability issues in reduced-precision simulations. The evolution of the error from periodic at 10 mantissa bits to exponential at 5 bits is shown in Figure 19. This behaviour aligns with findings by Gao (2023), who observed that reduced-precision computations introduce cumulative numerical errors that deviate from the analytical baseline, manifesting as exponential error growth at lower precisions in our case. The transition from periodic to exponential error profiles underscores the limitations of low-precision arithmetic, particularly below 10 mantissa bits, where the solution fidelity deteriorates rapidly. Such behaviour has implications for applications requiring high accuracy, as the trade-off between computational efficiency and solution stability becomes critical.

The numerical artifact seen in Figure 15 marks the onset of spurious harmonics in the frequency spectrum of the wavefield. Subsequent spectra, such as those shown in Figures 16 and 17, clearly demonstrate the accumulation of these harmonics, which become increasingly pronounced at lower precisions. This behaviour can be attributed

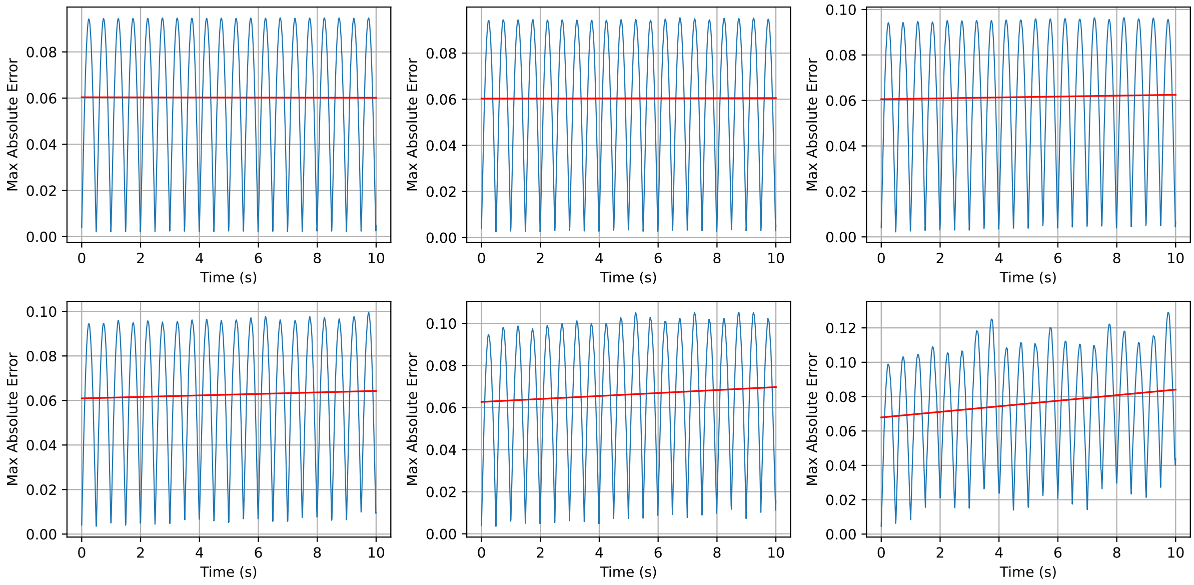

Figure 18: Plots **(a)**, **(b)**, **(c)**, **(d)**, **(e)** and **(f)** show the error profile for our numerical solution at 15, 14, 13, 12, 11 and 10 bits respectively. We show a least-squares trend line in red to highlight the acquisition of zero-offset in the error. The trend first starts to develop in Plot **(c)**, gradually increasing through to the clear sloped trend line observed in Plot **(f)**.

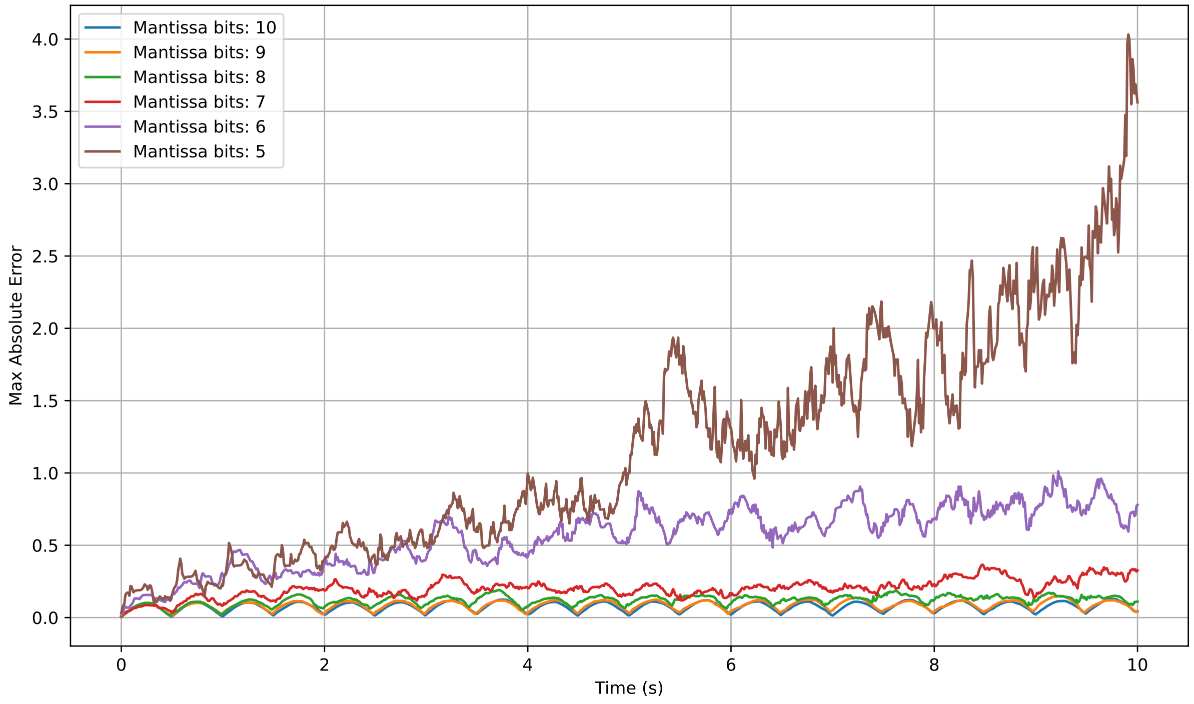

Figure 19: We show error profiles through time for 10, 9, 8, 7, 6, 5 mantissa bits to illustrate the shift from periodic to exponential growth in error. While some periodicity is maintained at precisions below 10 bits, the zero-offset continues to grow. At 6 mantissa bits, the error profile begins to exhibit an exponential trend, and by 5 bits, the exponential behaviour is unmistakable.

to rounding errors that compound during the iterative updates of the wavefield under reduced precision arithmetic. The distribution of numerical artefacts in the frequency spectra is significant, as they initially appear concentrated at the high and low ends of the spectrum before propagating to produce the additional harmonics observed. High-frequency noise contributes to numerical dispersion, further amplifying errors. As Li et al. (2024) highlight, high-frequency artifacts can lead to wavefield instabilities due to their exponential amplification during the compensation process. These instabilities can significantly degrade seismic imaging quality, necessitating additional processing steps to mitigate this.

Our study of the impact of reduced precision on wave equation solutions highlights the importance of selecting floating-point formats with bit allocations that align with the specific requirements of the application.

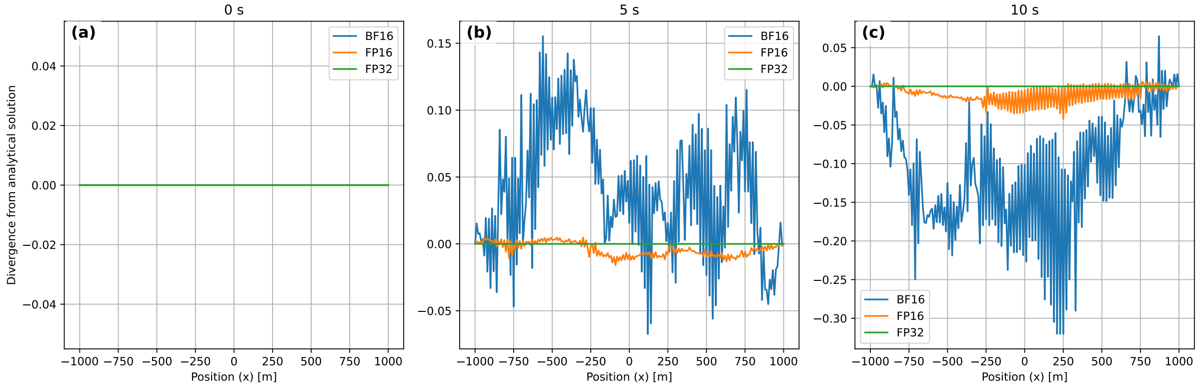

Figure 20: Difference between analytical solution and numerical solution for FP32, FP16, and BF16 equivalent levels of precision. We observe a significant discrepancy in the deviation from the analytical solution for our BF16 equivalent precision level in comparison to FP16. Note that while both FP16 and BF16 use 16 bits in total, BF16 allocates 7 bits to the mantissa and 8 bits to the exponent, whereas FP16 allocates 10 bits to the mantissa and 5 bits to the exponent. This allocation gives BF16 a larger dynamic range, but at the cost of reduced precision in comparison to FP16, a fact that is clearly shown by the difference in solutions.

As shown in Figure 20, solutions obtained with FP16 equivalent precision are significantly closer to those of the industry-standard FP32. In contrast, the BF16 equivalent precision deviates markedly from both FP32 and FP16, particularly at later simulation times. This inadequacy is further evidenced in Figure 21, which shows the error progression over time. Previous studies, such as Fabien-Ouellet (2020) and Wan et al. (2024), have successfully employed FP16 for seismic modelling with reduced-precision arithmetic. Our findings further support the conclusion that FP16 is the most suitable reduced-precision format for seismic applications. Beyond seismic modelling, our results suggest that BF16 precision may be unsuitable for a broader range of scientific computing tasks. Unlike wave equations, which are not chaotic systems and thus accumulate error in a relatively controlled manner, chaotic systems such as fluid dynamics are far more sensitive to error accumulation.

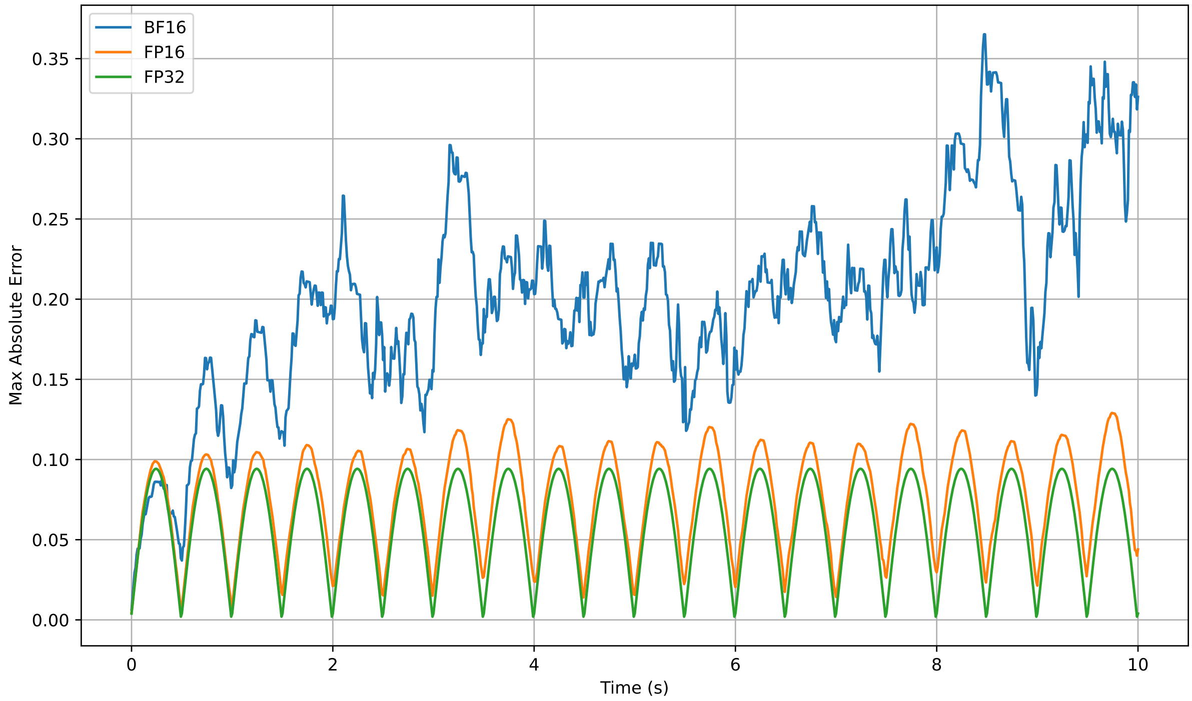

Figure 21: Maximum absolute error profile over time for FP32, FP16, and BF16 equivalent levels of precision. We observe similar, periodic behaviour for both FP32 and FP16, with FP16 displaying a significant zero-offset as shown in Figure 18. The error profile for BF16 is clearly distinct, lacking periodicity and increasing in a comparatively exponential fashion.

5 Conclusions and Future Work

5.1 Conclusions

In this work, we propose a two-step approach to leveraging modern computational hardware in seismic modelling:

- **Scaling wave equations**: Normalising physical parameters to ~ 1 ensures wave equations fit within the dynamic range of reduced precision floating-point formats.

- **Solving in reduced precision**: Once scaled, wave equations can be accurately solved using FP16 arithmetic, leveraging AI-optimized hardware.

5.1.1 Scaling Wave Equations

Our scaling method proves effective across various seismic modelling contexts. It works best for equations governing quantities not explicitly coupled with time, such as the acoustic wave equation, the TTI equations of Fletcher et al. (2009), and stress tensor components from Virieux (1986). However, caution is required for time-dependent quantities. In Virieux (1986), the velocity vector undergoes a systematic shift due to the new time unit, producing a numerically distinct yet qualitatively similar solution. More broadly, our method mitigates overflow and underflow by removing extreme values while preserving system behaviour. As it is based on physical units, it extends beyond geophysics to other PDEs and numerical methods. This generality suggests potential applications in diverse computational domains requiring efficient PDE solutions.

5.1.2 Solving Wave Equations in Reduced Precision

Our investigation of reduced precision arithmetic yields key insights for seismic modelling and computational physics. FP16 precision proves suitable for solving wave equations, in agreement with Fabien-Ouellet (2020) and Wan et al. (2024). In contrast, BF16's bit allocation results in three times the error of FP16, making it unsuitable for seismic applications despite identical memory savings. Given FP16's superior accuracy, BF16 is likely unsuitable for other scientific computing tasks, particularly in chaotic systems sensitive to error accumulation. Like FP8, BF16 appears better suited to AI and machine learning, whereas FP16 remains the reduced-precision format of choice for scientific computing. These findings underscore the importance of aligning numerical methods with modern hardware capabilities, paving the way for more efficient and accurate simulations across scientific disciplines.

5.2 Future work

This work leaves a few open questions that could be addressed by future studies:

- **Validating scaling method on modern computational hardware**: A logical next step is to implement a similar test case to the 1D acoustic wave system described in this work, applying the proposed scaling method and solving it using hardware optimised for reduced precision arithmetic. Such a study would serve to validate the accuracy of the scaling method in real-world computational environments while also quantifying the performance gains achievable through reduced precision. This investigation would not only advance the field of seismic modelling but also provide insights into the broader applicability of the method across other memory-bound numerical methods, such as finite-volume and particle-based approaches, demonstrating their potential to leverage modern computational hardware effectively.

- **Leveraging Fourier analysis to select units**: Future studies could refine the scaling method by investigating the problem in Fourier space. Rather than defining a time unit based solely on the CFL condition with a Courant number of 1, as in this work, a Fourier-based approach could involve taking the Fourier transform of the wavefield to identify its dominant frequency. This frequency could then inform a more natural choice of time unit, better aligned with the energy propagation characteristics of the system. Such an approach could enhance the accuracy and robustness of the scaling method, particularly for systems with complex frequency content.

- **Integrate Kahan summation to accommodate chaotic systems**: Future work could explore the implementation of the Kahan summation algorithm to preserve numerical accuracy at lower precisions (Gao, 2023). While scaling alone may suffice for stable wave equations, more chaotic systems, such as those in fluid dynamics, may require additional techniques to prevent divergence. Furthermore, the Kahan summation could enhance the feasibility of sub-FP16 precision formats, such as FP8, by mitigating the accumulation of rounding errors. The improved numerical accuracy provided by this technique may partially compensate for the increased susceptibility to precision loss inherent in these lower-bit-depth representations.

- **Applications of mixed precision**: An effective way to balance computational efficiency with numerical accuracy is through mixed-precision arithmetic. Heterogeneous precision strategies—where critical computations such as time-stepping and summations use FP32/FP64, while spatial derivatives and stencil operations leverage FP16/FP8—have been successfully applied in deep learning accelerators (Micikevicius et al., 2017). Extending this approach to wave equation solvers could enhance performance, particularly in large-scale seismic simulations where memory bandwidth is a limiting factor.

Appendices

A Why is Precision Important?

To further illustrate the importance of precision in numerical methods, consider a simplified weather model where temperature, pressure, and velocity fields are updated iteratively. Suppose the temperature field is updated using an equation like:

$$T_{n+1} = T_n + \Delta T \cdot k, \tag{19}$$

where $\Delta T = 0.001$ and k is a scaling factor. Using FP32, ΔT may round to 0.00100000005, introducing a negligible initial error. However, after 10^6 iterations, the total error accumulated by the system increases dramatically:

$$\text{Total Error} = \text{Iterations} \times \text{Rounding Error}. \tag{20}$$

Insufficient precision can also lead to a loss of significant digits and, consequently, information when subtracting similar values. Consider a pressure gradient ΔP given by:

$$\Delta P = P_2 - P_1, \tag{21}$$

where $P_2 \approx P_1$. In a lower precision floating-point format, the difference ΔP may round to zero due to insufficient significant digits. This loss of information would propagate through the system, affecting any calculations dependent on ΔP, potentially leading to unphysical results or numerical instability.

B Derivation of Scaling Method

Consider an arbitrary grid discretised in SI units, with spacing Δx in the x direction and Δy in the y direction. A spatial unit is first defined as $\Delta x\,\text{m}$, under the assumption that $\Delta x = \Delta y$. For irregular grids where $\Delta x \neq \Delta y$, the spatial unit is defined using the larger of the two spacings. This results in a new grid spacing of:

$$\Delta x = 1\,\Delta x\,\text{m} \tag{22}$$

We then make use of the Courant-Friedrichs-Lewy (CFL) condition to ensure the stability FD schemes (Courant et al., 1967; De Moura and Kubrusly, 2013). This condition is expressed as:

$$\frac{V_{max}\Delta t}{\Delta x} = C, \quad C \leq 1, \tag{23}$$

Here, V_{max} is the highest velocity in the system (e.g., the maximum velocity in a given velocity model), Δx is the grid spacing, Δt is the time step, and C is the Courant number. The physical interpretation of Equation 23 is that, to ensure stability in the FD scheme, information in the system must not propagate faster than the numerical time-stepping scheme allows. Setting $C = 1$ corresponds to the condition for maximum stability in the FD scheme.

Substituting V_{max} in $\Delta \text{x}\,\text{m/s}$ into Equation 23 with $C = 1$ gives:

$$\frac{V_{max}\Delta t}{1} = 1 \tag{24}$$

re-arranging Equation 24 for Δt yields a value of:

$$\Delta t = \frac{1}{V_{max}}\mathrm{s} \tag{25}$$

We use the result in Equation 25 to define a new time unit of $\frac{1}{\mathrm{V_{max}}}$ s with V_{max} given in Δxm/s.

This time unit is then applied to all time-dependent parameters in the system, including time step, velocity, frequency, and the length of the simulation. This transformation normalises V_{max} to $1\,(\Delta\mathrm{xm})/(\frac{1}{\mathrm{V_{max}}}\mathrm{s})$ and scales all other values in the velocity model to be less than or equal to 1. Using Equation 23 to calculate Δt with this transformed velocity yields a time step equal to the selected Courant number, ensuring it is always less than or equal to 1. Scaling the frequency also scales the source term, as the source term is defined using frequency. Applying the new time unit increases the simulation length by a factor of V_{max}; however, a very large simulation length is not problematic since this value is not used to update the wavefield directly—it only defines the number of time steps to iterate over, however this does not increase the number of time steps.

C Velocity and Parameter Models

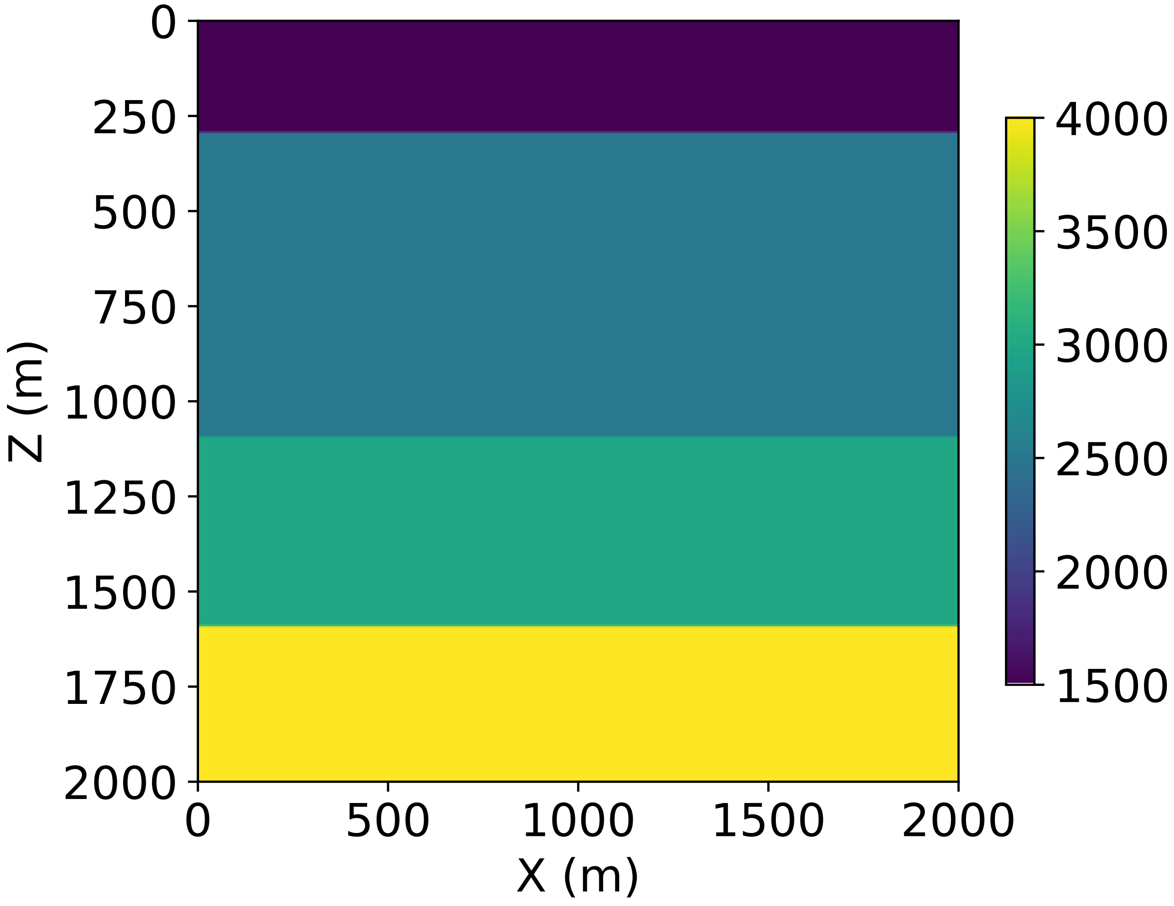

Figure 22: Layered velocity model used for the variable test case of the 2D Acoustic Wave Equation in Devito.

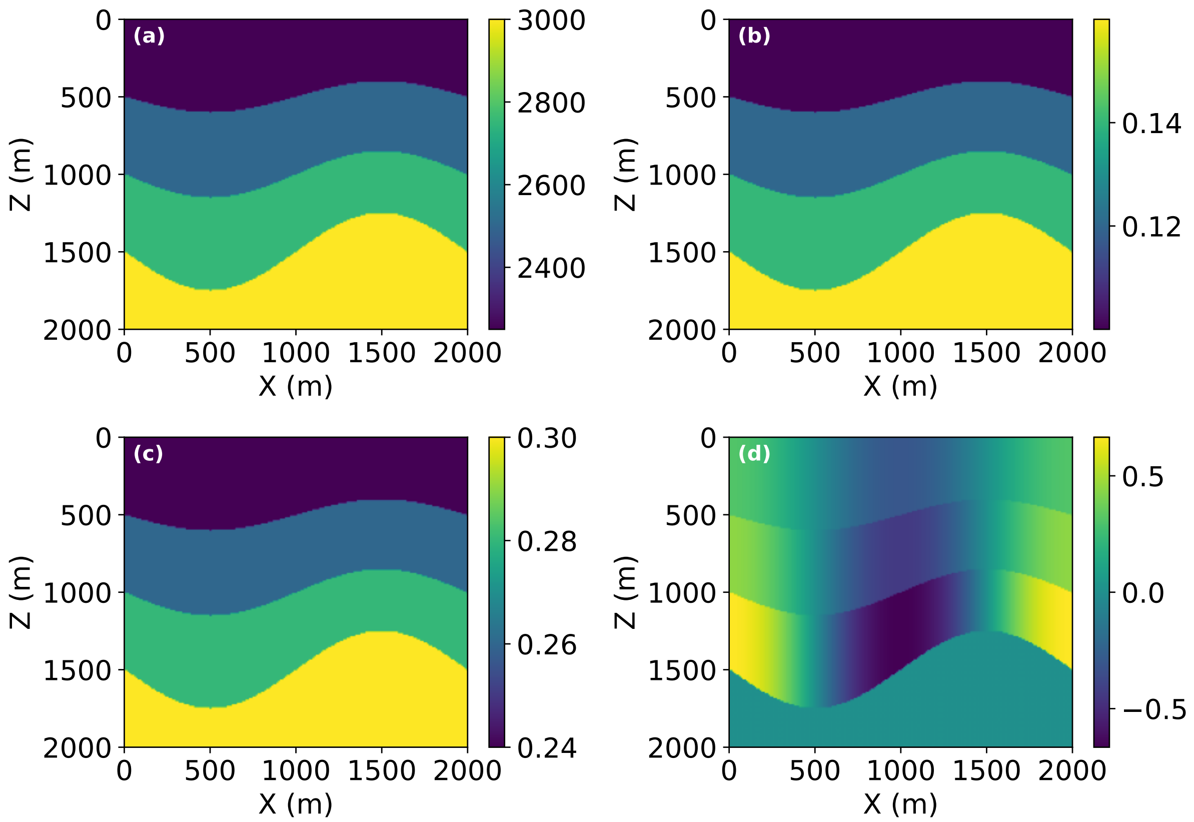

Figure 23: Variable parameter models used for our test case of Fletcher et al. (2009)'s TTI system. Plot **(a)** shows our model for V_p, this is used to calculate subsequent velocities in the system. Plots **(b)** and **(c)** show our models for the dimensionless anisotropic factors δ and ϵ respectively. Plot **(d)** shows our field for the tilt angle θ.

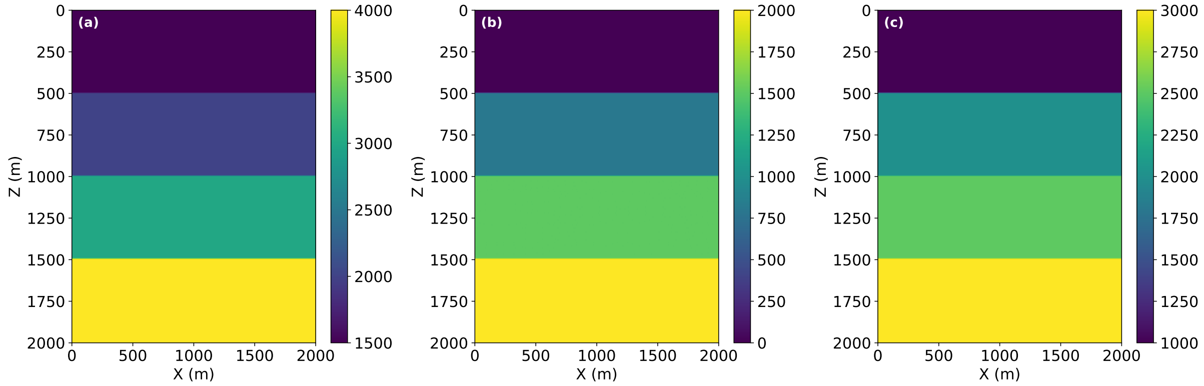

Figure 24: Variable V_p, V_s and ρ models used for our test case of the elastic wave equation as outlined by Virieux (1986). Plot **(a)** shows our model for V_p. Plots **(b)** and **(c)** show our models for V_s and ρ respectively.

References

Samuel Adams, Jason Payne, and Rajendra Boppana. Finite difference time domain (fdtd) simulations using graphics processors. In *2007 DoD High Performance Computing Modernization Program Users Group Conference*, pages 334–338. IEEE, 2007.

Keiiti Aki and Paul G Richards. *Quantitative seismology*. University Science Books,U.S., 2002.

Julia Ankudinova and Matthias Ehrhardt. On the numerical solution of nonlinear black–scholes equations. *Computers & Mathematics with Applications*, 56(3):799–812, 2008.

Hideo Aochi, Thomas Ulrich, Ariane Ducellier, Fabrice Dupros, and David Michea. Finite difference simulations of seismic wave propagation for understanding earthquake physics and predicting ground motions: Advances and challenges. In *Journal of Physics: Conference Series*, volume 454, page 012010. IOP Publishing, 2013.

Marco Armoni. Tensor processing units (tpu): A technical analysis and their impact on artificial intelligence. *Tech4Future Reports*, 2024. URL https://tech4future.info/wp-content/uploads/2024/11/Tensor-Processing-Units-TPU-Paper-ENG.pdf.

Marc Baboulin, Alfredo Buttari, Jack Dongarra, Jakub Kurzak, Julie Langou, Julien Langou, Piotr Luszczek, and Stanimire Tomov. Accelerating scientific computations with mixed precision algorithms. *Computer Physics Communications*, 180(12):2526–2533, 2009.

Shujaut H Bader and Xiaojue Zhu. Afid-mhd: a finite difference method for magnetohydrodynamic flows. *Journal of Computational Physics*, page 113658, 2024.

George Keith Batchelor. *An introduction to fluid dynamics*. Cambridge university press, 2000.

BN Biswas, Somnath Chatterjee, SP Mukherjee, and Subhradeep Pal. A discussion on euler method: A review. *Electronic Journal of Mathematical Analysis and Applications*, 1(2):2090–2792, 2013.

Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654, 1973.

George W Bluman and Julian D Cole. The general similarity solution of the heat equation. *Journal of mathematics and mechanics*, 18(11):1025–1042, 1969.

Michael Carilli and Jared Casper. What every user should know about mixed precision training in pytorch, 2021. https://pytorch.org/blog/what-every-user-should-know-about-mixed-precision-training-in-pytorch/.

Chio-Zong Cheng and Georg Knorr. The integration of the vlasov equation in configuration space. *Journal of Computational Physics*, 22(3):330–351, 1976.

Google Cloud. Bfloat16: The secret to high performance on cloud tpus, 2019. https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus.

Richard Courant, Kurt Friedrichs, and Hans Lewy. On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, 11(2):215–234, 1967.

Gideon Oluseyi Daramola, Boma Sonimiteim Jacks, Olakunle Abayomi Ajala, and Abiodun Emmanuel Akinoso. Enhancing oil and gas exploration efficiency through ai-driven seismic imaging and data analysis. *Engineering Science & Technology Journal*, 5(4): 1473–1486, 2024.

Bonnie Ives de Castro Nunes, Walter Eugenio De Medeiros, Aderson Farias do Nascimento, and José Antonio de Morais Moreira. Estimating quality factor from surface seismic data: A comparison of current approaches. *Journal of Applied Geophysics*, 75 (2):161–170, 2011.

Carlos A De Moura and Carlos S Kubrusly. The courant–friedrichs–lewy (cfl) condition. *AMC*, 10(12):45–90, 2013.

AA Dobrynina, VV Chechel'nitskii, and VA San'kov. Seismic quality factor of the lithosphere of the southwestern flank of the baikal rift system. *Russian Geology and Geophysics*, 52(5):555–564, 2011.

Sergey V Ershkov, Evgeniy Yu Prosviryakov, Natalya V Burmasheva, and Victor Christianto. Towards understanding the algorithms for solving the navier–stokes equations. *Fluid Dynamics Research*, 53(4):044501, 2021.

Donald Estep. The forward euler method. *Practical Analysis in One Variable*, pages 583–604, 2002.

Gabriel Fabien-Ouellet. Seismic modeling and inversion using half-precision floating-point numbers. *Geophysics*, 85(3):F65–F76, 2020.

Robin P. Fletcher, Xiang Du, and Paul J. Fowler. Reverse time migration in tilted transversely isotropic (tti) media. *Geophysics*, 74(6):WCA179–WCA187, 2009. doi: 10.1190/1.3269902.

Longfei Gao. Compensated sum and delayed update for time dependent wave simulations at half precision. *arXiv preprint arXiv:2310.00236*, 2023.

David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM computing surveys (CSUR)*, 23(1):5–48, 1991.

David J Griffiths. *Introduction to electrodynamics.* Cambridge University Press, 2023.

William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI: portable parallel programming with the message-passing interface*, volume 1. MIT press, 1999.

Dave Hale. Methods to compute fault images, extract fault surfaces, and estimate fault throws from 3d seismic images. *Geophysics*, 78(2):O33–O43, 2013.

Yaju Hao, Duowen Yin, Peng Zhang, and Hongjing Zhang. Seismic decomposition method using ricker wavelet dictionary and its applications for q-value estimation. *Acta Geophysica*, 72(4):2425–2445, 2024.

Akash Haridas, Nagabhushana Rao Vadlamani, and Yuki Minamoto. Deep neural networks to correct sub-precision errors in cfd. *Applications in Energy and Combustion Science*, 12:100081, 2022.

John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011.

IEEE. Ieee standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–70, 2008. doi: 10.1109/IEEESTD.2008.4610935.

Xi Jiang. A review of physical modelling and numerical simulation of long-term geological storage of co2. *Applied energy*, 88(11):3557–3566, 2011.

Lawrence E Kinsler, Austin R Frey, Alan B Coppens, and James V Sanders. *Fundamentals of acoustics*. John wiley & sons, 2000.

Dimitri Komatitsch and Jeroen Tromp. Spectral-element simulations of global seismic wave propagation—i. validation. *Geophysical Journal International*, 149(2):390–412, 2002.

Dan D Kosloff and Edip Baysal. Migration with the full acoustic wave equation. *Geophysics*, 48(6):677–687, 1983.

Eugeniusz Kozaczka and Grażyna Grelowska. Theoretical model of acoustic wave propagation in shallow water. *Polish maritime research*, 24(2):48–55, 2017.

Fei Li, Qiang Mao, Juan Chen, Yan Huang, and Jianping Huang. Stable q-compensated reverse time migration in tti media based on a modified fractional laplacian pure-viscoacoustic wave equation. *Journal of Geophysics and Engineering*, page gxae066, 2024.

Jing Li, Gaurav Dutta, and Gerard Schuster. Wave-equation qs inversion of skeletonized surface waves. *Geophysical Journal International*, 209(2):979–991, 2017.

Yang Liu and Mrinal K Sen. A practical implicit finite-difference method: examples from seismic modelling. *Journal of Geophysics and Engineering*, 6(3):231–249, 2009.

M. Louboutin, M. Lange, F. Luporini, N. Kukreja, P. A. Witte, F. J. Herrmann, P. Velesko, and G. J. Gorman. Devito (v3.1.0): an embedded domain-specific language for finite differences and geophysical exploration. *Geoscientific Model Development*, 12 (3):1165–1187, 2019. doi: 10.5194/gmd-12-1165-2019. URL https://www.geosci-model-dev.net/12/1165/2019/.

Fabio Luporini, Mathias Louboutin, Michael Lange, Navjot Kukreja, Philipp Witte, Jan Hückelheim, Charles Yount, Paul H. J. Kelly, Felix J. Herrmann, and Gerard J. Gorman. Architecture and performance of devito, a system for automated stencil computation. *ACM Trans. Math. Softw.*, 46(1), apr 2020. ISSN 0098-3500. doi: 10.1145/3374916. URL https://doi.org/10.1145/3374916.

Raul Madariaga. Dynamics of an expanding circular fault. *Bulletin of the Seismological Society of America*, 66(3):639–666, 1976.

Paulius Micikevicius. 3d finite difference computation on gpus using cuda. In *Proceedings of 2nd workshop on general purpose processing on graphics processing units*, pages 79–84, 2009.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, et al. Fp8 formats for deep learning. *arXiv preprint arXiv:2209.05433*, 2022.

SM Mojahidul Ahsan, Anurag Dhungel, Mrittika Chowdhury, Md Sakib Hasan, and Tamzidul Hoque. Hardware accelerators for artificial intelligence. *arXiv e-prints*, pages arXiv–2411, 2024.

The mpmath development team. *mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 1.3.0)*, 2023. `http://mpmath.org/`.

Irshad R Mufti and Joseph T Fou. Applications of three-dimensional finite-difference seismic modeling in oil field exploitation and simulation. *International Journal of Imaging Systems and Technology*, 1(1):28–32, 1989.

Myrto Papadopoulou, Samuel Zappalà, Alireza Malehmir, Kristina Kucinskaite, Michael Westgate, Ulrik Gregersen, Thomas Funck, Florian Smit, and Henrik Vosgerau. Advancements in seismic imaging for geological carbon storage: study of the havnsø structure, denmark. *International Journal of Greenhouse Gas Control*, 137:104204, 2024.

Gerald W Recktenwald. Finite-difference approximations to the heat equation. *Mechanical Engineering*, 10(01), 2004.

Norman Ricker. Wavelet contraction, wavelet expansion, and the control of seismic resolution. *Geophysics*, 18(4):769–792, 1953.

Richard C Rodriguez and Jonah Elijah P Bardos. Fast object detection with a machine learning edge device. *arXiv preprint arXiv:2410.04173*, 2024.

Chris Semeniuk, Chris Elders, and Fabio Mancini. Pluto reservoir characterisation: Exploiting advances in full-waveform inversion technology. *CEED Seminar Proceedings 2017*, 2017.

Olivier Sentieys and Daniel Menard. Customizing number representation and precision. In *Approximate Computing Techniques: From Component-to Application-Level*, pages 11–41. Springer, 2022.

Barry F Smith. Domain decomposition methods for partial differential equations. In *Parallel numerical algorithms*, pages 225–243. Springer, 1997.

Edith Sotelo, Marco Favino, and Richard L Gibson Jr. Application of the generalized finite-element method to the acoustic wave simulation in exploration seismology. *Geophysics*, 86(1):T61–T74, 2021.

Dale G Stone. *Designing seismic surveys in two and three dimensions*. Society of exploration geophysicists, 1994.

Wei Sun, Ang Li, Tong Geng, Sander Stuijk, and Henk Corporaal. Dissecting tensor cores via microbenchmarks: Latency, throughput and numeric behaviors. *IEEE Transactions on Parallel and Distributed Systems*, 34(1):246–261, 2022.

William W Symes, Bertrand Denel, Adam Cherrett, Eric Dussaud, Paul Williamson, Paul Singer, and Laurent Lemaistre. Computational strategies for reverse-time migration. In *SEG International Exposition and Annual Meeting*, pages SEG–2008. SEG, 2008.

Christopher KW Tam and Jay C Webb. Dispersion-relation-preserving finite difference schemes for computational acoustics. *Journal of computational physics*, 107(2):262–281, 1993.

Leon Thomsen. Weak elastic anisotropy. *Geophysics*, 51(10):1954–1966, 1986.

Jean Virieux. P-sv wave propagation in heterogeneous media: Velocity-stress finite-difference method. *Geophysics*, 51(4):889–901, 1986.

Jean Virieux and Stéphane Operto. An overview of full-waveform inversion in exploration geophysics. *Geophysics*, 74(6):WCC1–WCC26, 2009.

Jean Virieux, Henri Calandra, and René-Édouard Plessix. A review of the spectral, pseudo-spectral, finite-difference and finite-element modelling techniques for geophysical imaging. *Geophysical Prospecting*, 59(Modelling Methods for Geophysical Imaging: Trends and Perspectives):794–813, 2011.

AnatoliĬ Aleksandrovich Vlasov. The vibrational properties of an electron gas. *Soviet Physics Uspekhi*, 10(6):721, 1968.

Jialiang Wan, Wenqiang Wang, and Zhenguo Zhang. Enhancing computational efficiency in 3-d seismic modelling with half-precision floating-point numbers based on the curvilinear grid finite-difference method. *Geophysical Journal International*, 238(3): 1595–1611, 2024.

Michael Warner, Andrew Ratcliffe, Tenice Nangoo, Joanna Morgan, Adrian Umpleby, Nikhil Shah, Vetle Vinje, Ivan Štekl, Lluís Guasch, Caroline Win, et al. Anisotropic 3d full-waveform inversion. *Geophysics*, 78(2):R59–R80, 2013.

Jidong Yang and Hejun Zhu. A time-domain complex-valued wave equation for modelling visco-acoustic wave propagation. *Geophysical journal international*, 215(2):1064–1079, 2018.

O Yilmaz. Seismic data analysis: processing, inversion and interpretation of seismic data. *Society of Exploration Geophysicists*, 463, 2001.

Hua-Wei Zhou, Hao Hu, Zhihui Zou, Yukai Wo, and Oong Youn. Reverse time migration: A prospect of seismic imaging methodology. *Earth-science reviews*, 179:207–227, 2018.