applied sciences

MDPI

Article

Enhancement of In-Plane Seismic Full Waveform Inversion with CPU and GPU Parallelization

Min Bahadur Basnet [1,*,†], Mohammad Anas [2], Zarghaam Haider Rizvi [1,3,†], Asmer Hamid Ali [2,‡], Mohammad Zain [2,‡], Giovanni Cascante [3] and Frank Wuttke [1]

1 Geomechanics & Geotechnics, Kiel University, 24118 Kiel, Germany
2 Zakir Hussain College of Engineering & Technology, Aligarh Muslim University , Aligarh 202002, India
3 Department of Civil & Environmental Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada
* Correspondence: min.basnet@ifg.uni-kiel.de; Tel.: +49-431-880-2854
† Current address: GeoAnalysis Engineering GmbH, 24118 Kiel, Germany.
‡ These authors contributed equally to this work.

Abstract: Full waveform inversion is a widely used technique to estimate the subsurface parameters with the help of seismic measurements on the surface. Due to the amount of data, model size and non-linear iterative procedures, the numerical computation of Full Waveform Inversion are computationally intensive and time-consuming. This paper addresses the parallel computation of seismic full waveform inversion with Graphical Processing Units. Seismic full-waveform inversion of in-plane wave propagation in the finite difference method is presented here. The stress velocity formulation of the wave equation in the time domain is used. A four nodded staggered grid finite-difference method is applied to solve the equation, and the perfectly matched layers are considered to satisfy Sommerfeld's radiation condition at infinity. The gradient descent method with conjugate gradient method is used for adjoined modelling in full-waveform inversion. The host code is written in C++, and parallel computation codes are written in CUDA C. The computational time and performance gained from CUDA C and OpenMP parallel computation in different hardware are compared to the serial code. The performance improvement is enhanced with increased model dimensions and remains almost constant after a certain threshold. A GPU performance gain of up to 90 times is obtained compared to the serial code.

Keywords: seismic wave propagation; full waveform inversion; CUDA; GPU parallelisation; finite difference method

check for updates

Citation: Basnet, M.B.; Anas, M.; Rizvi, Z.H.; Ali, A.H.; Zain, M.; Cascante, G.; Wuttke, F. Enhancement of In-Plane Seismic Full Waveform Inversion with CPU and GPU Parallelization. *Appl. Sci.* **2022**, *12*, 8844. https://doi.org/10.3390/app12178844

Received: 3 May 2022
Accepted: 16 August 2022
Published: 2 September 2022

1. Introduction

The physical properties of the Earth's subsurface have always been of interest to humankind. Ranging from scientific development to industrial use, such as oil and gas industries and engineering applications, determining or approximating the physical properties is essential. Exploration geophysics is the branch of geophysics that uses physical methods such as seismic, acoustic, electromagnetic or electric measurements to extract information about the Earth's subsurface without physical disturbances. Full Waveform Inversion (FWI) is a wave inversion procedure where the high-quality estimate of the subsurface parameters is obtained by minimising the misfit between the observed and modelled data. Seismic and acoustic waveform inversion is widely in practice. Seismic or acoustic signals are applied at the surface, which gets propagated through the subsurface of interest, and the output signals are obtained at receiver points (see Figure 1). The wave phenomenon is modelled using an appropriate numerical technique, and the output is compared to that obtained from the field test. The model is updated with a mathematically calculated approximation function until we get a desired minimum value of the misfit to get a high-quality approximation of the subsurface properties [1]. This paper deals mainly with time-domain seismic wave inversion in a two-dimensional system using the finite difference method (FDM).

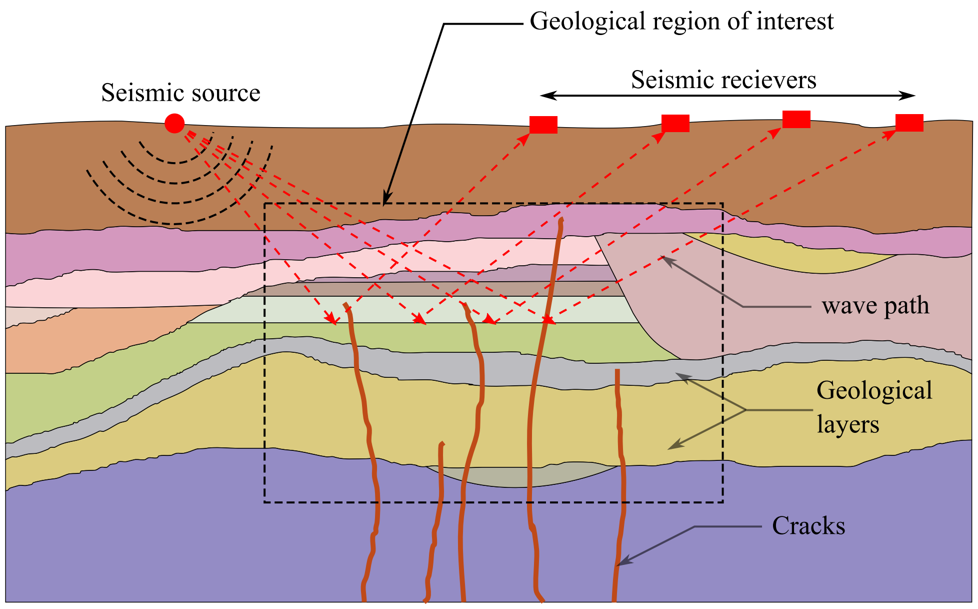

Figure 1. A schematic representation of geophysical investigation using seismic waveform inversion of a subsurface geological region with multiple layers.

1.1. The Seismic Full Waveform Inversion

The first introduction and numerical implementation of FWI was found to be developed in the 1980s; see [2–7]. Since its first implementation, many authors have implemented and modified it for different purposes and methods. The time domain FWI approach is applied in frequency domain approach in [8,9]. The performance of frequency domain approach of FWI is optimised with parallelisation by [10,11]. FWI is widely used for inversion of: (a) acoustic waves, e.g., [12–14] (b) seismic waves, e.g., [12,15–17]. With the advancement in modern day computers and parallel processing capabilities, FWI is gaining its popularity in exploration geophysics. A basic flow chart of Seismic FWI in shown in Figure 2.

The Finite Difference Method (FDM) is one of the most widely used inversion methods in the numerical implementation of FWI. The conventional FDM is based on the straightforward numerical implementation of Partial Differential Equations (PDE) for the given Boundary Value Problem (BVP). The partial differential terms are substituted with algebraic differences for a small argument x, such that $x \to 0$, which makes it easy to implement and understand (refer Equations (6) and (7)). The finite difference formulation for seismic waves can be found in: (a) [18–20] for forward wave propagation problems and (b) [12,21–23] for FWI problems. To satisfy Sommerfeld's radiation condition at infinity (i.e., the wave radiating towards infinity should not come back unless there are reflecting boundaries), absorbing boundaries or perfectly matched layers (PML) [12] is used. PML damps the outgoing wave field over a distance (number of grids in discrete form), making it more efficient and accurate than a normal absorbing boundary. Details about PML can be found in [24–26].

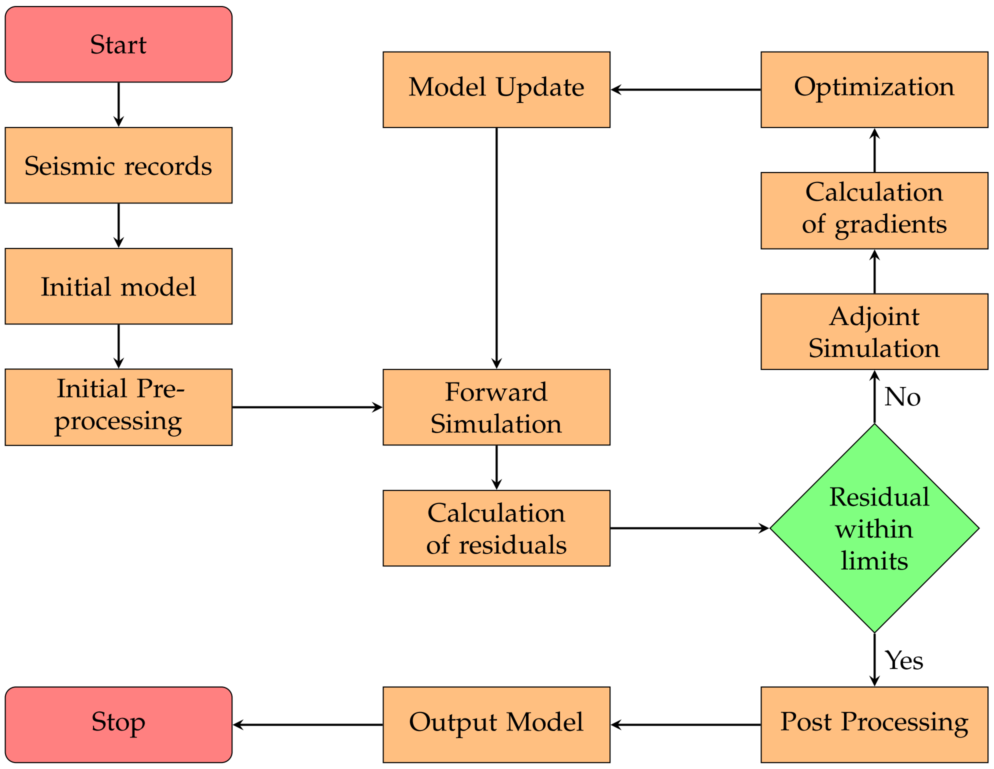

Figure 2. A flowchart of Seismic Full Waveform Inversion.

1.2. Parallel Computation

The finite difference for FWI is computationally intensive since, in each iteration step of FWI, the forward wave propagation model has to be computed several times and is both time and resource-consuming. Therefore, reducing the computation costs of FWI has been an important area of research for geophysical researchers. Parallel processing can efficiently perform seismic wave propagation and full-waveform inversion computations at a large scale. Processors can be used in parallel to achieve superior performance. To maximise the potential of parallel machines, the research challenge is to reframe the problem, develop parallel algorithms, and devise new computational strategies. The research objective is to reframe the problem, develop parallel algorithms, and invent alternative computing methodologies to maximise the benefits of parallel computing. Processing speeds on a single thread have stagnated in the past two decades. However, computer hardware and software advances have made parallel processing a realistic and appealing technology.

GPUs (Graphics Processing Units) are now among the most popular computing resources for parallel computation because of their high processing rates and low cost. Their capabilities have evolved from simple peripherals to powerful, programmable, and sophisticated processors in their own right. The potential of using inexpensive graphics technology for general-purpose computing has sparked great interest.

Since the early 2000s, seismic forward modelling has used parallelisation strategies based on software enhancements utilising APIs such as OpenMP and hardware improvements by porting codes to GPUs [27]. Jiang and Zhu [28] discuss the implementation of GPU-based 2D elastic FWI in the time domain on a single GPU card. The authors utilised a boundary-saving strategy to reconstruct forward wave fields to reduce the significant RAM utilisation. The work demonstrates that using shared memory can reduce the modelling time by about a third. Wang et al. [29] describes the development of a parallel scheme to speed up FWI on GPUs with Compute Unified Device Architecture (CUDA). Using the GTX480 GPU, the authors could speed up the FWI 80 times compared to the Central Processing Units (CPU) implementation. The acceleration of a 3-D finite-difference in the time domain wave propagation code on NVIDIA GPU graphics cards and the CUDA programming language is discussed in [30]. The authors also used Message Passing Interface

(MPI) to use multiple GPUs in parallel. The authors report the acceleration of the wave propagation code by a factor between 20 to 60 compared to a serial implementation.

It can be concluded from the literature review that the parallel programming of FWI in GPU has significantly improved the computational speed. However, the different physical methods have to be tested, and the computational efficiency needs to be further enhanced to adopt it for practical applicability. In this paper, we implemented the numerical model of seismic forward and full-waveform inversion models for P/SV wavefield in GPU and tested it against the serial and CPU parallel codes. As mentioned earlier, the work investigates the parallelisation of the method using the OpenMP and CUDA Application Programming Interface (API) on the CPU and GPU, respectively.

1.3. Importance, Scope and Limitations of the Study

FWI is a handy non-destructive method to obtain accurate information about the subsurface parameters of the earth of critical infrastructure, and its importance is still increasing. However, FWI primarily has two significant scopes for improvement to find widespread applicability in exploration geophysics and engineering. They are: (a) the improvement of the optimisation and parameter search algorithms, which is not the scope of this study, and (b) the improvement in computational efficiency. FWI is an iterative algorithm that demands high computational costs and high memory, which is generally out of the scope of general computers. If possible, the computation for the given hardware often takes days or months, which makes it practically inefficient.

The present study mainly focuses on enhancing in-plane seismic full waveform inversion with CPU and GPU parallelisation. The existing numerical algorithms are implemented in serial and parallel codes, and the computational efficiency achieved by parallelisation is studied. Thus, the computational enhancement by parallelisation is limited to the specific algorithm used in the programming. The study was conducted about in-plane elastic waves in the time domain, and the full waveform inversion was conducted using the step-gradient method (See [31]). Other advanced algorithms, where faster and more robust search methods enhance the FWI algorithms, are not within the scope of this study. Additionally, the test and validation are conducted for the limited number of hardware available; thus, the performance boost due to CPU and GPU parallelisation may vary depending upon the hardware configuration.

2. Mathematical Model

2.1. Stress Velocity Formulation of 2D Elastic Wave Equation

The stress velocity formulation of the elastodynamic equation, as given by [12] and substituted for two-dimensional in-plane wave motion (P-SV wavefield), can be written as in Equations (1)–(5).

$$\rho\frac{\partial v_x}{\partial t} = \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + f_x, \tag{1}$$

$$\rho\frac{\partial v_y}{\partial t} = \frac{\partial \sigma_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + f_y, \tag{2}$$

$$\frac{\partial \sigma_{xx}}{\partial t} = (\lambda + 2\mu)\frac{\partial v_x}{\partial x} + \lambda\frac{\partial v_y}{\partial y} + \frac{\partial \sigma_{xx0}}{\partial t}, \tag{3}$$

$$\frac{\partial \sigma_{yy}}{\partial t} = \lambda\frac{\partial v_x}{\partial x} + (\lambda + 2\mu)\frac{\partial v_y}{\partial y} + \frac{\partial \sigma_{yy0}}{\partial t}, \tag{4}$$

$$\frac{\partial \sigma_{xy}}{\partial t} = \frac{1}{2}\left(\tfrac{\partial v_x}{\partial y} + \tfrac{\partial v_y}{\partial x}\right) + \frac{\partial \sigma_{xy0}}{\partial t} \tag{5}$$

where x and y are two-dimensional spatial variables, t is time variable, v_x and v_y are particle velocities along x and y directions, $\partial\sigma_{xx}$, $\partial\sigma_{yy}$ and $\partial\sigma_{xy}$ are principle and shear stress variables, $\partial\sigma_{xx0}$, $\partial\sigma_{yy0}$ and $\partial\sigma_{xy0}$ are body stresses and, $\frac{\partial}{\partial t}$, $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ are partial differential operators to the respective variables.

2.2. Numerical Implementation in FDM

For the numerical implementation of Elastodynamic Wave Equations (1)–(5), they are discretized in finite time and space over a two-dimensional grid. The finite-difference operators are based on simple differences over a small interval of space and time as given in Equations (6) and (7).

$$\frac{\partial P^+}{\partial x} \approx \frac{P[i+1]-P[i]}{dh} \text{ Forward operator.} \tag{6}$$

$$\frac{\partial P^-}{\partial x} \approx \frac{P[i]-P[i-1]}{dh} \text{ Backward operator} \tag{7}$$

P is the given function, i is the grid index, and dh is a small increment in the space or time variable x. A combination of forward and backward operators is also possible.

A standard staggered grid (SSG), suggested by [32,33], is widely implemented by other researcher for the computation of two-dimensional seismic wave propagation problems. The standard staggered grid, as implemented in [12] and shown in Figure 3 is used for numerical implementation in the paper. An arithmetic mean is used to obtain Density ρ, and a harmonic mean is used to obtain Lamis' Constant parameter μ at the half grid points in the staggered grid. A detail of Finite Difference Discretization of Seismic FWI Problem and computational implementation can be found in [12,31].

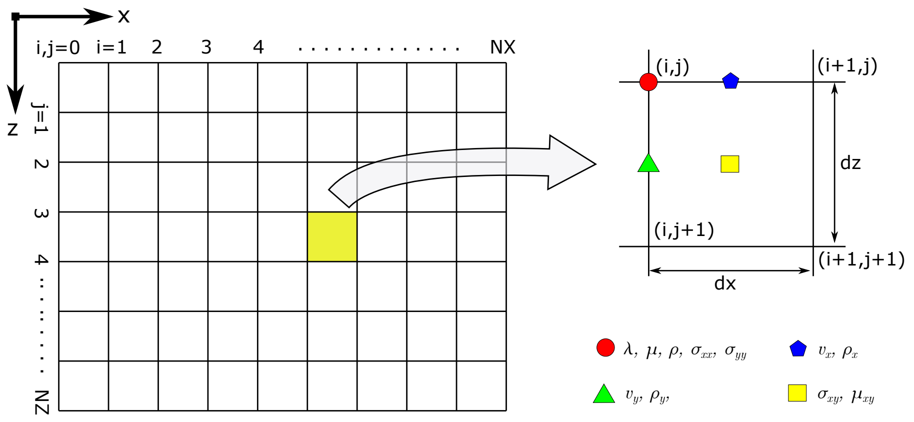
X
i,j=0 i=1 2 3 4 . . . . NX
Z
j=1 2 3 4 . . . . NZ
(i,j)
(i+1,j)
dz
(i,j+1)
(i+1,j+1)
dx
λ, μ, ρ, σ_{xx}, σ_{yy}
v_x, ρ_x
v_y, ρ_y,
σ_{xy}, μ_{xy}

Figure 3. A staggered grid configuration for use in seismic finite difference modelling [12,31–33].

Perfectly Matched Layers (PML), as implemented in [31], is used as a boundary condition. The perfectly matched layer frame is added to the finite difference grid, where the wave equation is solved in the frequency domain with coordinate stretching in this region (see Figure 4). The arrangement absorbs the outgoing waves and creates a reflectionless boundary frame (see [24]) and the Sommerfeld's radiation condition is numerically implemented.

For FWI, a start model has to be initialised in the first step. A start model could be informed guesswork with the information available about the ground. The mathematical model is solved for the given seismic source, and the output signals are recorded at the receiver locations. This part is called the forward model, and the modelled output at the receivers is called modelled data (u^{mod}). The seismic recordings that we have from the field recordings are referred to as observed data (u^{obs}). The difference between them is referred to as residual data and can be measured using vector norm. For this case, we use L_2 norm, given by Equation (8). The Gradient-based method given in [12] is implemented in parallel FWI computation in the paper.

$$| L |_2 = \left(\sum_{i=1}^{n} | u^{mod} - u^{obs} |^2\right)^{1/2} \tag{8}$$

where n is the number of receivers. L_2 norm is related to energy, which is used to calculate the energy gradients, and with proper step length, we gradually minimise the misfit with every iteration; see [31].

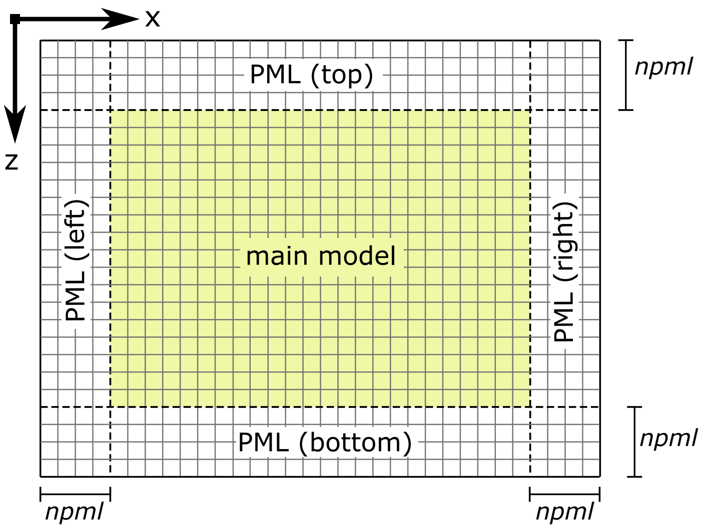

Figure 4. Perfectly matched layers in finite difference grid (*npml* = Number of grids in perfectly matched layers).

3. Parallel Computational Model

In terms of CPU clock speed, current generation computing has reached saturation, primarily due to the enormous increase in cost and power requirement for increasing clock frequency further from current standards [34]. Until 2004 Moore's law was in full force, with chip densities doubling every two years. It also increased clock speed as the chip dimension decreased due to Dennard scaling. According to this principle, the power needed to run transistors in a given volume remains constant regardless of the number of transistors. Because transistors do not scale with size, power density increases as transistors get smaller. Increasing clock speeds also implies a higher voltage, which increases power consumption. Since the scaling was performed assuming most of the power was dynamic, with a smaller transistor, the static power, namely leakage, began to dominate. In addition, more heat is generated with increasing clock speeds, requiring more robust cooling solutions.

The processor shifted from a single-core to a multicore to combat the abovementioned issue. Power and frequency are fundamentally linked in multicore processors. By incorporating multiple cores, the frequency of each core can be lowered, which allows the power generally given to a single core to be distributed among multiple cores. Thus, a multicore processor produces significantly higher performance [35] and results in parallel processing. By focusing more on increasing the core count rather than the clock speed, parallel computing is one of the most optimal methods of achieving high performance from a CPU [36]. The comparison between OpenMP and CUDA is given in Table 1.

Table 1. Major features of OpenMP and CUDA [37].

	OpenMP	**CUDA**
Parallelism	-Data Parallelism -Asynchronous task parallelism -Host and device	-Data Parallelism -Asynchronous task parallelism -Device only
Architecture abstraction	-Memory hierarchy -Data and computation binding -Explicit data mapping and movement	-Memory hierarchy -Explicit data mapping and movement
Synchronisation	-Barrier -Reduction -Joint	-Barrier
Framework Implementation	-Compiler directives for C/C++ and Fortran	C/C++ extensions

3.1. OpenMP

OpenMP is a parallel programming model for shared memory and distributed shared memory multiprocessors. The OpenMP API supports multi-platform shared-memory parallel programming in C/C++ and Fortran. The OpenMP API defines a portable, scalable model with a simple and flexible interface for developing parallel applications on platforms from the desktop to the supercomputer. OpenMP supports the parallelisation of small portions of a program at a time rather than all at once, a method also known as incremental parallelisation. As a result, it is possible to observe the impact of parallelisation of individual functions on the application [38]. Through OpenMP, the compiler guides the parallelisation of code through directives, easing the burden on programmers of doing it manually [39]. The steps to parallelise the given code using OpenMP have been elaborated in Figure 5. An important aspect is identifying which functions take the longest to execute. The second step involves identifying the code blocks in the function that could be parallelised. This is conducted by ensuring that operations in parallel are not accessed in the middle of their execution and that each result is calculated independently of the others in parallel. After declaring sections as parallel, define variables, as each thread may require a variable only accessible to them. Finally, several test cases must be run to check that the results are identical to sequential code and that there is no race condition.

3.2. GPU

GPUs are programmable processors among the most powerful computing devices available in terms of performance and price. GPUs are high-density processors known to accelerate graphics applications compared to CPUs due to their multi-core design and high memory bandwidth. Because of the GPU’s parallel architecture, it has sustained a steady performance boost. A current-generation high-end GPU exceeds a high-end general-purpose CPU in terms of processing power. On a GPU, a shared memory pool available to all threads in the block performs substantially better than a global memory pool (See Figure 6. The programming model is based on the hierarchy of hardware parallelism. The GPU kernel divides all threads into equal-sized static workgroups of several hundred. Workgroups effectively exchange state and synchronise, allowing for coordinated data processing. A workgroup’s threads are all scheduled to run on a single core simultaneously. Serial execution of the code takes place on the CPU. On the GPU, parallelism is represented via a kernel function that runs many threads simultaneously.

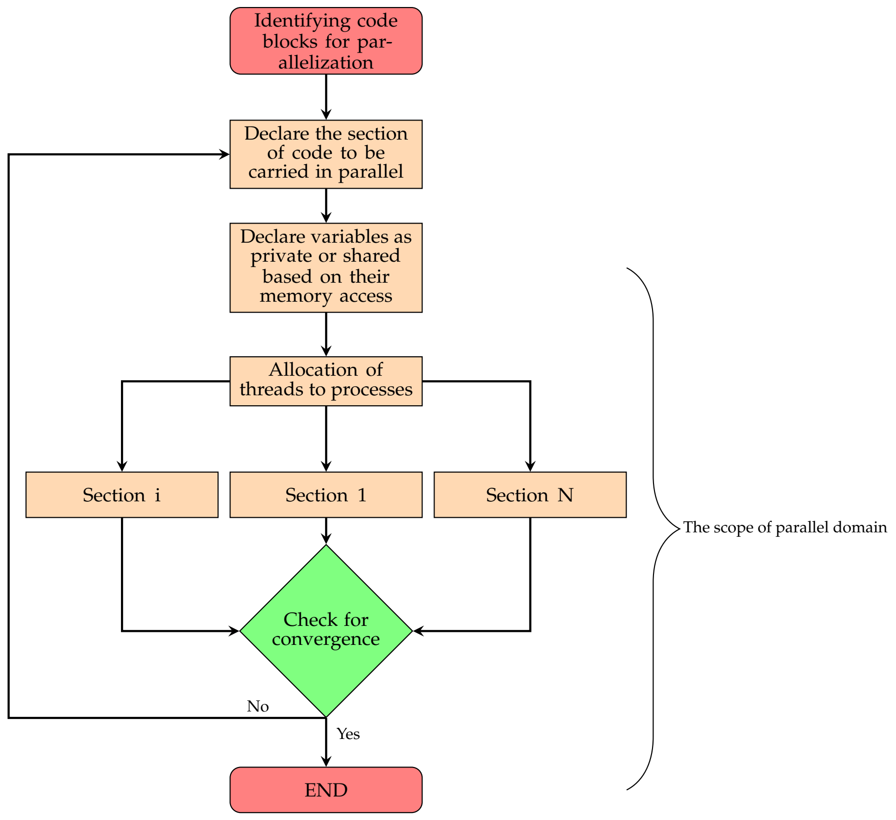
Identifying code blocks for parallelization
Declare the section of code to be carried in parallel
Declare variables as private or shared based on their memory access
Allocation of threads to processes
Section i
Section 1
Section N
Check for convergence
No
Yes
END
The scope of parallel domain

Figure 5. A flowchart of parallel computation using OpenMP.

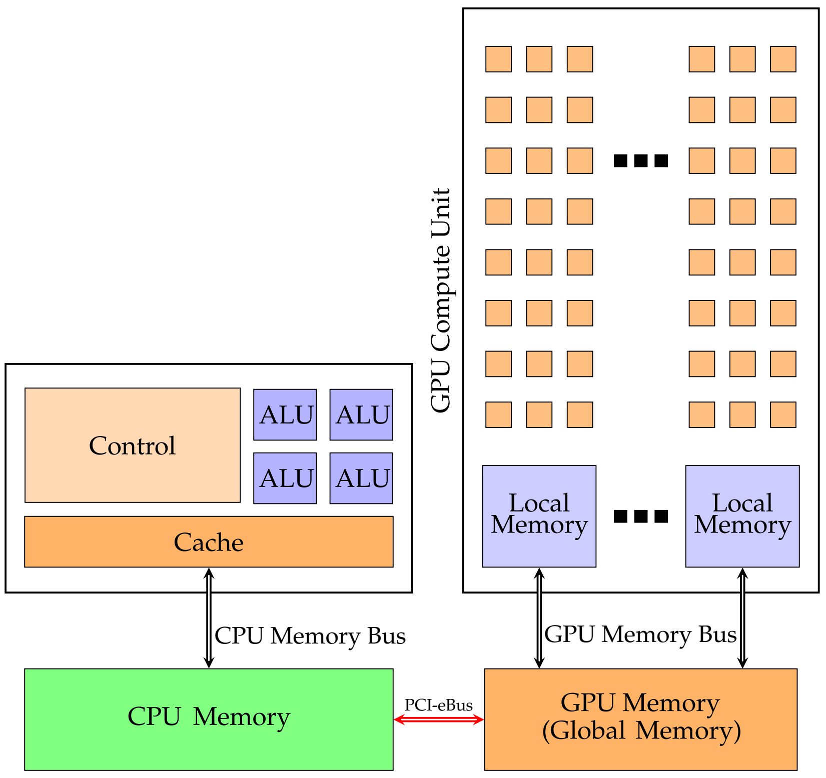

Figure 6. CPU and GPU Architecture.

Compute Unified Device Architecture (CUDA)

The introduction of NVIDIA's CUDA allowed it to be used to accelerate non-graphical applications. CUDA programming on GPUs has drastically increased computing performance, making it a popular parallelism option. CUDA is a C-based language designed to use GPUs' massive parallelism. Nvidia's CUDA framework comprises an expanded C/C++ programming language and optimised libraries (cuBLAS, cuFFT) for common mathematical tasks. CUDA provides a basic abstract programming model that may be used on several GPU generations in the future.

The steps to parallelise the code using CUDA on the GPU have been elaborated in Figure 7. The prepossessing is conducted on the CPU using Python programming language and is saved as binary files. The files are read in C++ host code, from which the necessary variables are transferred to the GPU device for the parallel computation of forward and FWI simulations. The GPU is used for the rest of the calculation. The forward simulations are run over the initial input parameters. Then the output from the forward simulations is compared with the field measurement records to calculate the data misfit. If the data misfit is less than the minimum, the parameters (Lami's constants: λ, μ and Density: ρ) are estimated to the desired accuracy. Suppose the data misfit is more than the minimum; the simulation is forwarded to an adjoint simulation where the gradients of the parameters are calculated. The computed gradients are then smoothed and optimised, and the best fitting step length is calculated using a parabolic line search algorithm. With the gradients and the step length the medium parameters are updated as $m_{i+1} = l\,\partial m + m_i$, where i is the iteration step, m_i is the medium parameters at i^{th} iteration step, ∂m the gradient of the parameter and l the computed step length. The forward simulation is run in the next iteration with the updated medium parameters. The loop runs until the misfit is obtained within the desired limits. After the desired misfits are obtained, the latest obtained medium parameters are copied to the host CPU as binary files, where the post-processing of the results continues in Python.

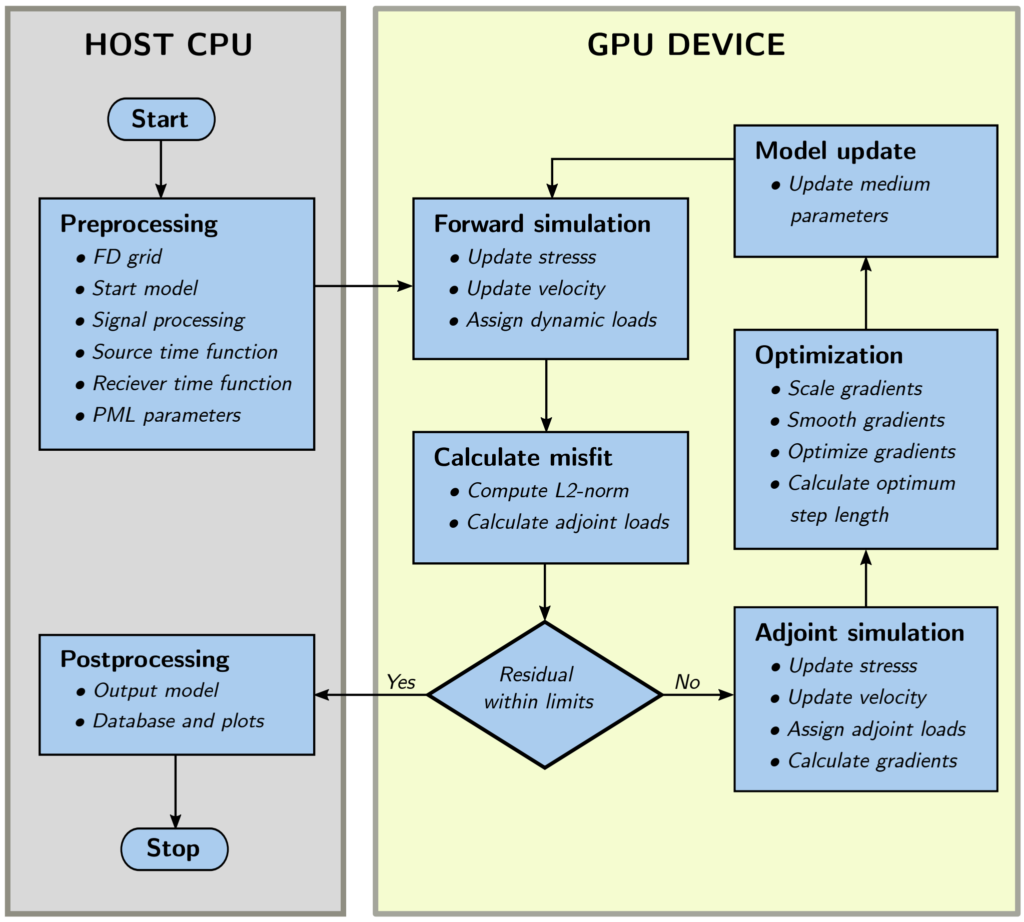

Figure 7. A flow chart of parallel computation of seismic FWI in GPU.

4. Numerical Simulations

The section presets the implementation of seismic Forward and FWI simulations in parallel computation based on GPU and compare the performance to that of parallel computation in CPU described in Section 3. The hardware used for computation is as shown in Table 2. Two different simulations are considered to study the efficiency of parallel computation in GPU. To study the computational efficiency of the forward-only model, a scenario of seismic wave propagation in a water reservoir dam is presented in Section 4.1. For the seismic FWI, a spherical inclusion in full space is shown in Section 4.2.

Table 2. CPU and GPU hardware used for computation in this study.

Hardware	CPU/GPU	Memory	OS
CPU1	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40 GHz	8 GB	Ubuntu 20.04
CPU2	AMD Ryzen Threadripper 3970X(2.9 GHz)	64 GB	Ubuntu 20.04
GPU1	Nvidia GeForce GTX 1650	4 GB	Ubuntu 20.04
GPU2 (installed with CPU2)	Nvidia GeForce GTX 2080 Ti	8 GB	Ubuntu 20.04

4.1. Seismic Forward Model

For the numerical simulation of the forward wave propagation problem, we consider an earthen water reservoir dam with the dimensions as shown in Figure 8. The depth of the water on the upstream slope of the dam is 3.0 m. We consider a line crack (5.6 cm wide and 10 m long) starting at 1.5 m below the centre of the free surface of the dam. The crack extends towards the far-field at an angle of 45 degree to the vertical. The crack is filled with extremely soft saturated soil. The water table inside the dam is represented by the draw-down curve, as shown in the figure. A realistic range of wave propagation is considered by adopting the elastic medium parameters as suggested in [40]. The medium parameters for the model are shown in Table 3 and the effect of the air-water and the air-solid interface is considered a free surface. A synthetic time signal (Ricker wavelet) of unit amplitude and centre frequency of 0.8 kHz acting vertically is considered. The seismic source fired at the top of the upstream slope of the dam. Twenty-five seismic receivers are positioned along the dam's upstream and downstream slope with a spacing of 0.53 m ($dx = 0.5$ m, $dz = 0.17$ m). A source function of unit amplitude is considered; thus, the seismograms recorded at the receivers are the unitless values normalized by the excitation function. The dam is initially (at $t = 0$) considered at rest.

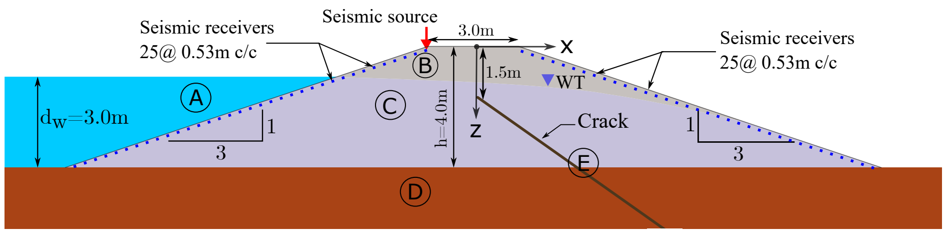

Figure 8. A physical model of dam including a crack and soil-water interface to perform forward seismic wave simulation. The material parameters are as given in Table 3 and 55 seismic receivers are located along the dam surface with spacing of 0.53 m.

Table 3. Medium parameters for forward seismic modelling.

Medium	**Code**	**P-Wave Velocity** (c_1)	**S-Wave Velocity** (c_2)	**Density** (ρ)
Water	A	1482 m/s	0.0	1000 $\mathrm{kg/m^3}$
Unsaturated soil in the dam	B	800 m/s	400 m/s	1700 $\mathrm{kg/m^3}$
Saturated soil in the dam	C	1450 m/s	400 m/s	1950 $\mathrm{kg/m^3}$
Subsurface layers	D	1900 m/s	700 m/s	2100 $\mathrm{kg/m^3}$
Crack filler	E	1600 m/s	100 m/s	1000 $\mathrm{kg/m^3}$

The model is discretized with $nx = 1493$, $nz = 393$ grid points along the respective directions with grid spacing $dx = dz = 0.02$ m. Twenty perfectly matched layers are used in all four directions. The particle velocities normalized to their peak amplitudes at different receiver positions are shown in Figure 9. The output corresponds to the input Ricker wavelet signals of 0.8kHz central frequency as shown in Figure 10. The particle velocities v_z and v_x in the dam are shown in Figure 11 and Figure 12, respectively.

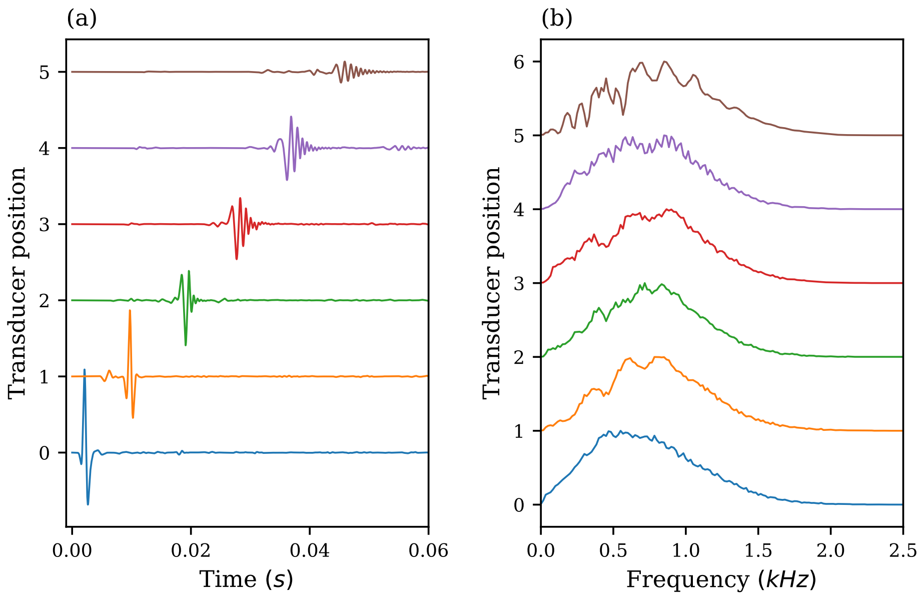

Figure 9. Vertical components of particle velocity, normalized by the peak amplitude of the input signal, recorded in the receivers located at the surface of the dam at different the transducer locations $x = [-1.5\text{ m}, \; 1.5\text{ m}, 4.5\text{ m}, 7.5\text{ m}, 10.5\text{ m}, 13.5\text{ m}]$: (**a**) Time signal, (**b**) The frequency amplitudes normalized to the peak amplitude of the same signal. (The source is located at $x = -1.5$ m).

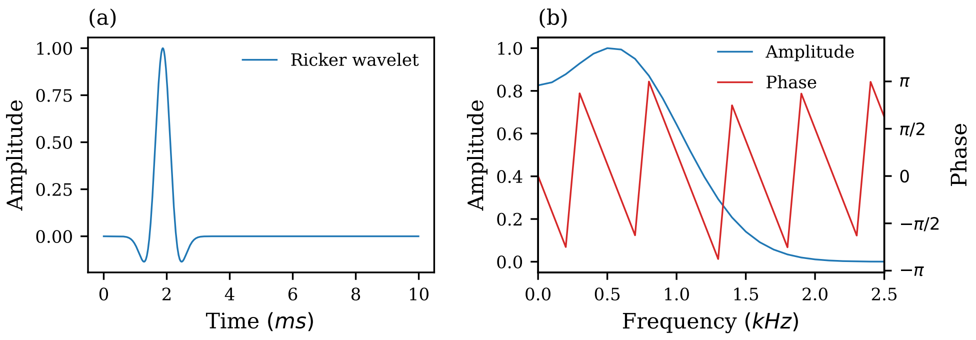

Figure 10. Ricker wavelet of unit amplitude with center frequency 0.8 kHz used as velocity source for the excitation in forward seismic simulation model: (**a**) normalized amplitude in time domain, (**b**) normalized amplitudes in frequency domain.

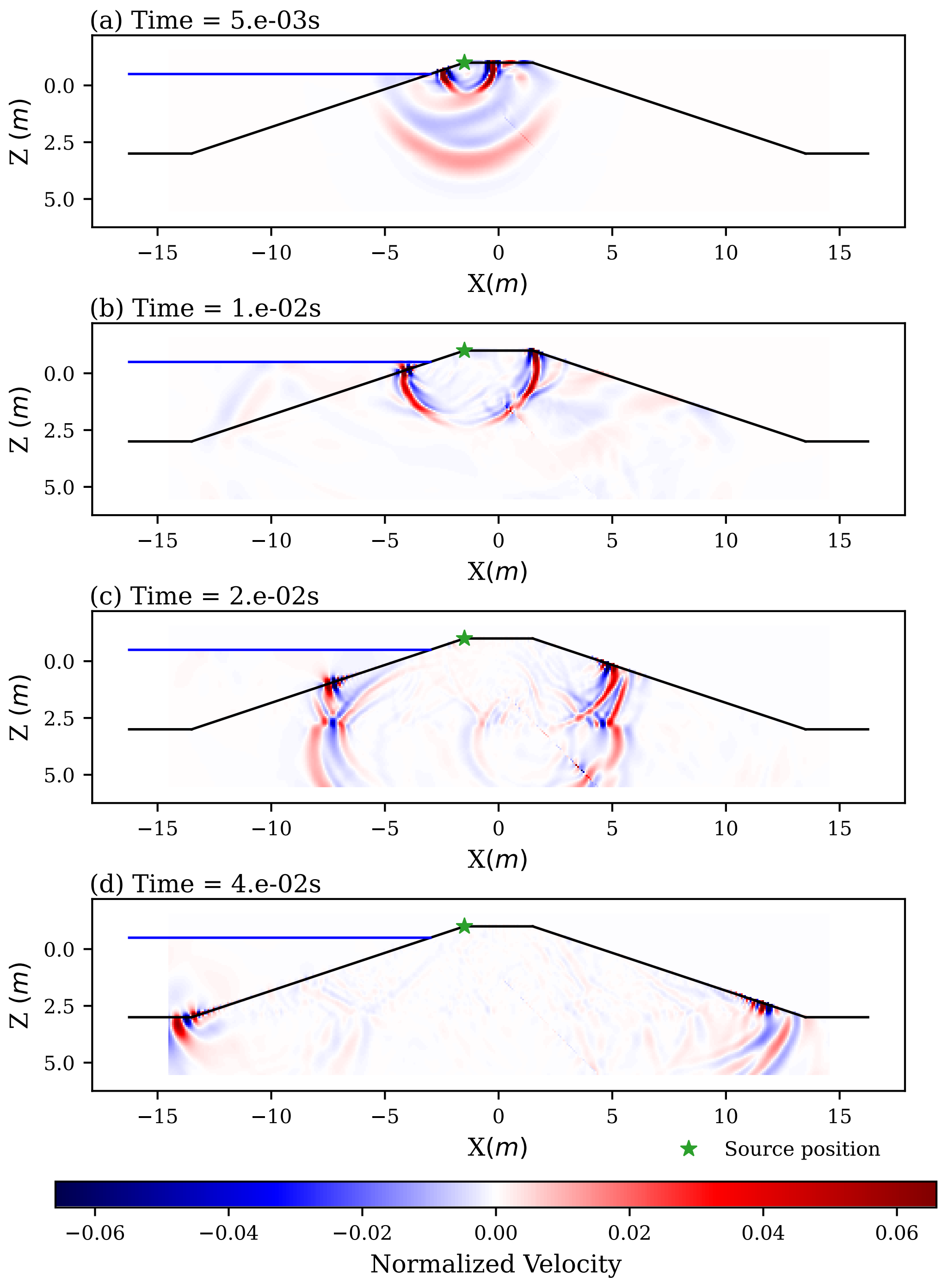

Figure 11. Vertical component of particle velocity (v_z), normalized to the velocity wavelet at the excitation point, in the forward seismic model.

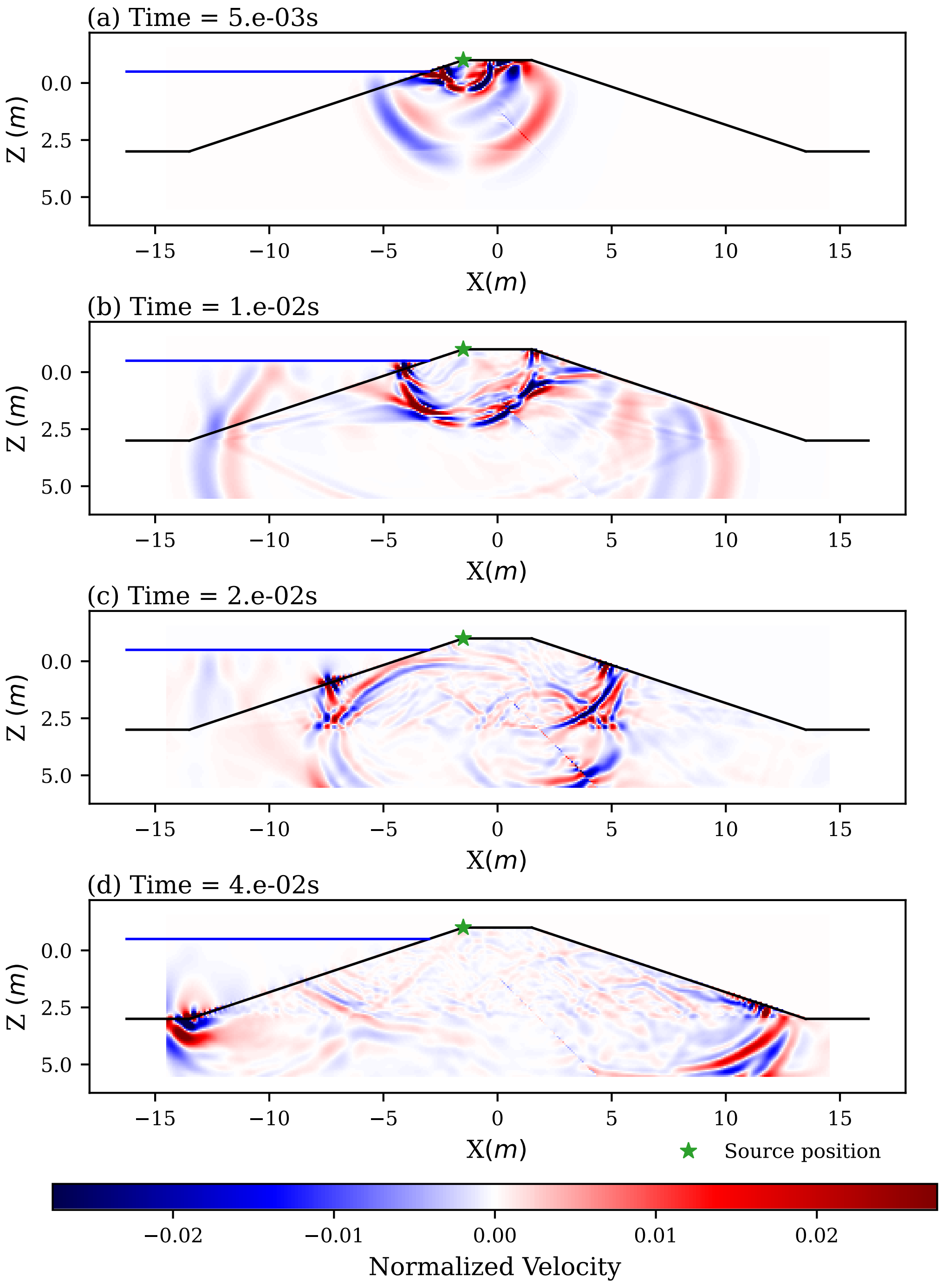

Figure 12. Horizontal component of particle velocity (v_x), normalized to the velocity wavelet at the excitation point, in the forward seismic model.

4.2. Seismic FWI Model

For FWI simulations, a simple spherical inclusion model, as used by [12] for acoustic FWI is considered. A rectangular homogeneous elastic region in range $x = (0, 15)$ m and $z = (0, 30)$ m in a homogeneous full-plane with density $\rho = 1500$ kg/m^3 and wave velocities $c_1 = 500$ m/s, $c_2 = 300$ m/s with a circular inclusion of radius $r = 3$ m at the center of the grid with density $\rho = 1700$ kg/ m ^{3} and wave velocities $c_1 = 800$ m/s, $c_2 = 400$ m/s is considered (Figure 13). Three seismic sources are used at coordinates $x = 1$ m and $z = 7.5$ m, 15 m, 22.5 m. Fifty five receivers are used at $x = 14$ m and between $y = 1.5$ m to $y = 28.5$ m in interval of 0.5 m, as shown in Figure 13. The model is discretized with $nx = 301$ and $nz = 601$ number of grids in x- and z-directions, respectively, with grid spacing $dx = dz = 0.05$ m. 10 convolutional PML layers are used in all four boundaries.

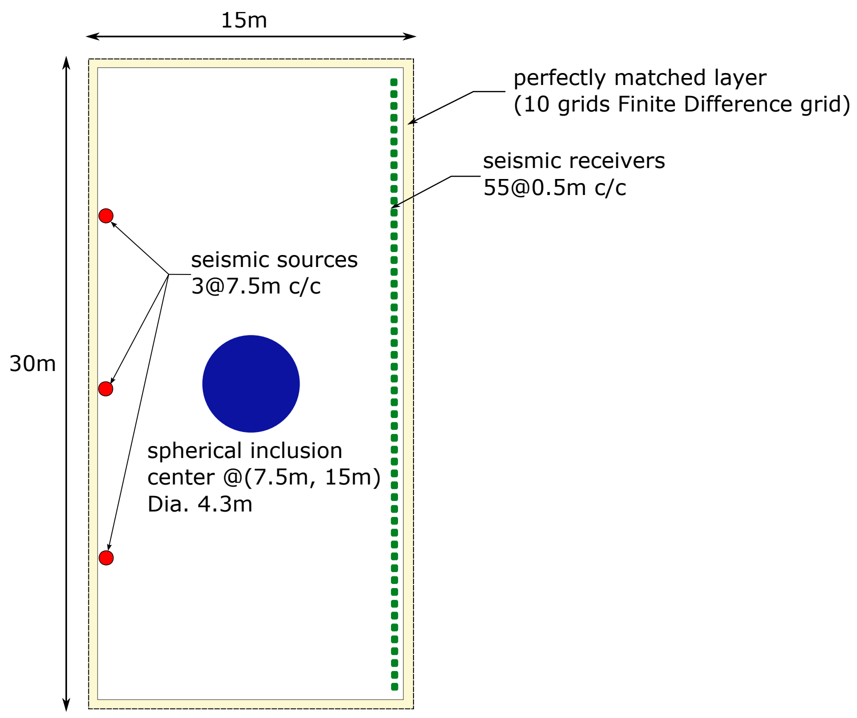

Figure 13. A spherical inclusion in a rectangular finite difference grid considered for FWI simulation.

A synthetic data set is considered as observed seismic records. A Ricker wavelet with peak frequency 2.50 kHz and amplitude 1.0 as shown in Figure 14, are fired simultaneously as input velocities in the z-direction, and the observed data is generated by the forward modelling of the true model. A seismic FWI simulation is performed with the developed parallel codes. The start model is a homogeneous full plane with the same material properties as the true model without spherical inclusion. The material updates in different iteration steps are shown in Figures 15 and 16 for longitudinal and shear wave velocity models, respectively. The convergence of the inversion model is represented by the change in L2-norm over the iteration steps, shown in Figure 17. The obtained results show a fair inversion of the given true model.

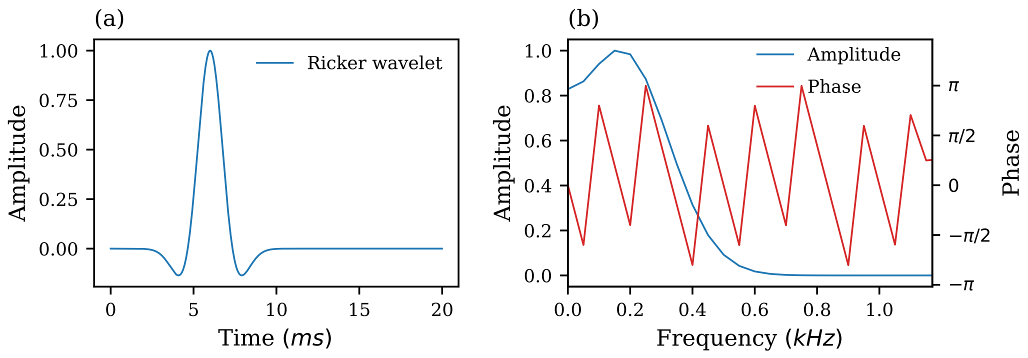

Figure 14. Ricker wavelet of unit amplitude with center frequency 250 Hz used as velocity source for the excitation in FWI model: (**a**) normalized amplitude in time domain, (**b**) normalized amplitudes in frequency domain.

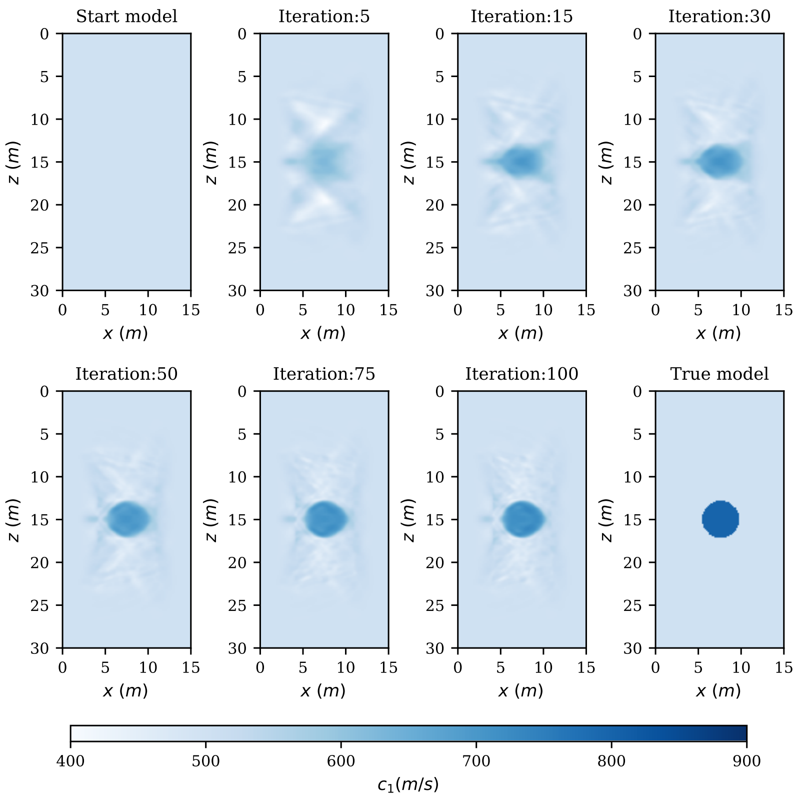

Figure 15. Longitudinal wave velocity inversion at different iteration steps in seismic FWI of a spherical inclusion in full-plane.

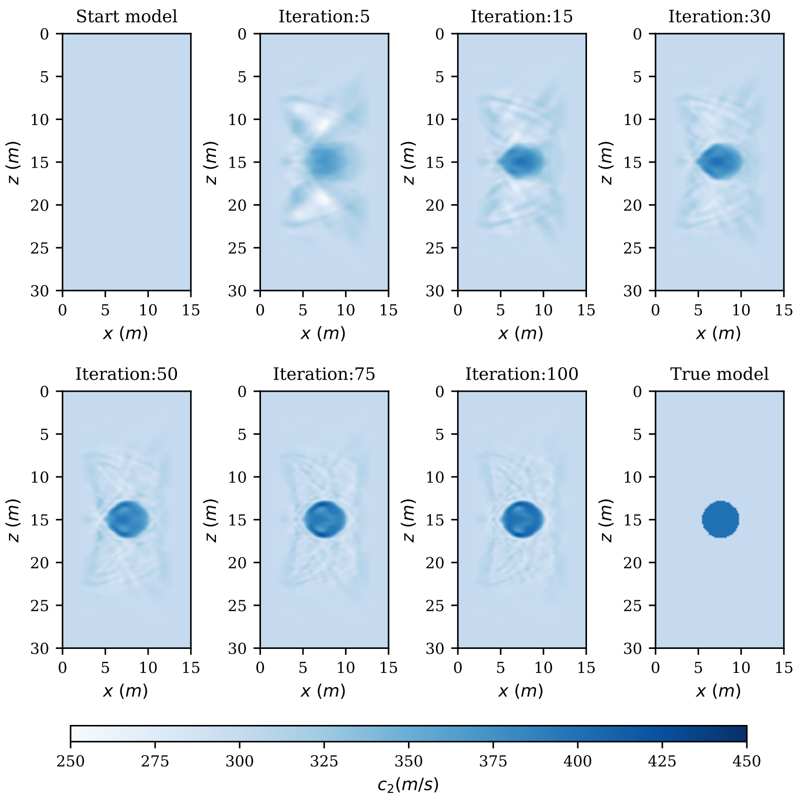

Figure 16. Shear wave velocity inversion at different iteration steps in seismic FWI of a spherical inclusion in full-plane.

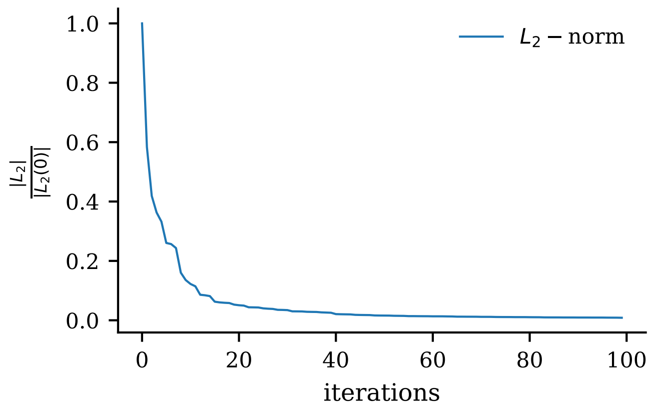

Figure 17. L_2 norm in the first 40 steps, normalized to the L_2 norm in the first step, in Seismic FWI a spherical inclusion in full-plane.

5. Results and Discussion

A series of studies are performed to show the enhancement of in-plane seismic full waveform inversion using CPU and GPU parallelization. Initially, the forward model is studied with an example of the wave propagation in a dam in Section 4.1. The validated forward model is further developed into the full waveform inversion model in Section 4.2 using low-frequency anamoly for a circular inclusion in full space. The developed FWI codes are programmed for serial computation in CPU, parallel computation in CPU using openMP parallel modules and parallel computation in GPU using CUDA C programming languages. Same physical parameters and mathematical models, with a variation in time and space grids, are used for comparative performance analysis of the developed codes over the available hardware.

The seismic wave propagation in the dam shows that the presence of cracks in the dam significantly alters the propagation of elastic waves (Figures 11 and 12). Similarly, a minor disturbance can also be observed due to the presence of a water table. The effective wavelength of P and SV waves in saturated soil inside the dam is measured in Figures 11 and 12 to be approximately 2.0 m and 0.7 m, respectively. The measured wavelengths correspond to the medium parameters and the central frequency of the input signal, which validates the seismic simulation codes in CPU and GPU. It can also be observed from Figure 9b that the peak frequency response shifts to slightly higher frequencies with the distance of wave propagation. From the perspective of computational efficiency, similar results are obtained from serial computation, parallel computation using openMP and parallel computation in GPU. Figures 18–20 show the computational time and performance boost in forward simulations. It can be seen that the performance in the most capable GPU that we used is 60 to 80 times that compared of the sequential code. The CPU parallel computation using OpenMP with 16 threads only shows an improvement by 2 to 4 times in the performance. For the GPU1, the simulation was out of memory after 3800-time steps for the given spatial grid; see Figure 18.

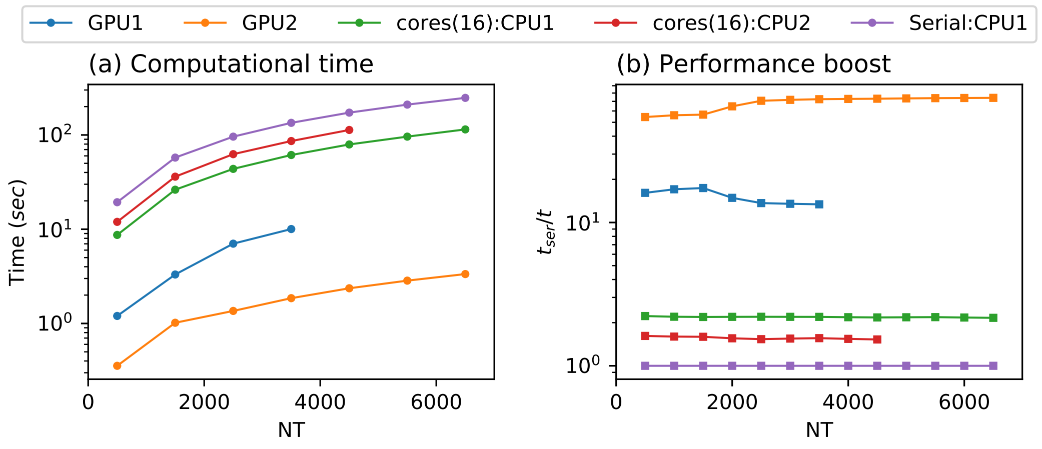

Figure 18. Comparison of computational time and performance for variation of temporal grid size NT keeping spatial grid size constant ($NX = 201$, $NZ = 401$) in forward simulations.

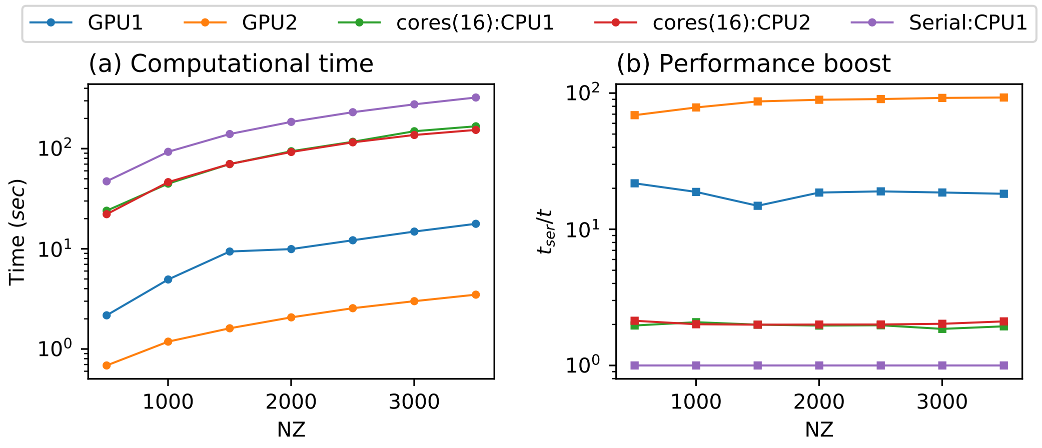

Figure 19. Comparison of computational time and performance for variation of spatial grid size along Z direction NZ keeping spatial grid size along X direction and temporal grid constant ($NX = 201$, $NT = 1000$) in forward simulations.

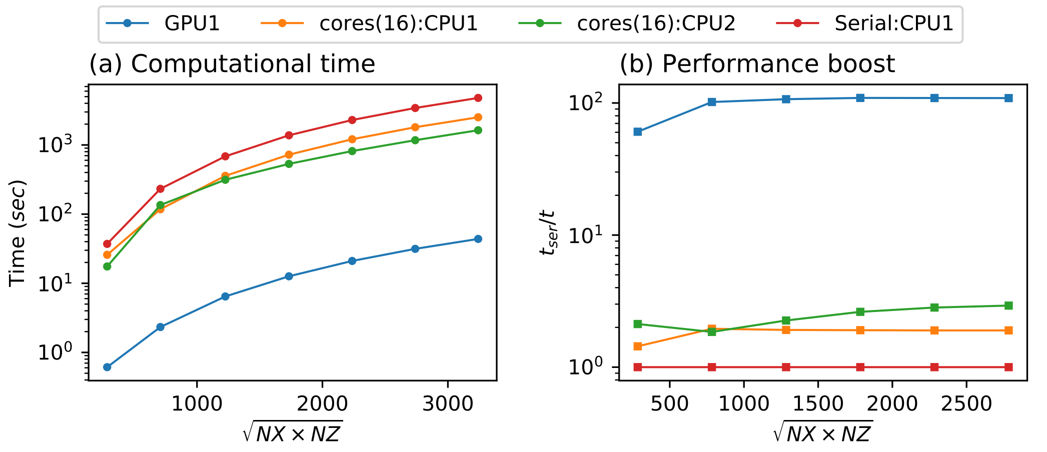

Figure 20. Comparison of computational time and performance for variation of spatial grid $NX \times NZ$ keeping temporal grid size constant ($NT = 1000$) in forward simulations.

It is observed from the FWI models that the higher wavelength band of the wave-field gives the general outline and the layers in the model guiding the convergence towards the global minima. In contrast, the lower wavelength band adds more detailing and sharper images. Thus, it can be interpreted that the low-frequency signals give less detailed or more blurred results near to the true model, whereas high-frequency signals give more detailed and accurate inversion results. However, computation in the high-frequency range has higher possibilities that the model may converge to the local minima. It is also observed for the spherical inclusion model that the higher the impedance ratio between the main body and the inclusion, the sharper inversion results and faster convergence are achieved. For the parallel computation of Full Waveform Inversion, the whole computation process is conducted in GPU, except the pre-processing and post-processing. It can be seen from Figure 7 that we have taken a strategy in that the whole iteration of full-waveform inversion is performed in GPU. The strategy is implemented to reduce the overload due to the memory copying from the host to the device and vice versa. The major limitation of this approach is that multiple GPUs can not be used and that the model's size depends upon the memory available in a single GPU. Similarly, Figures 21 and 22 show the computational time and performance boost in FWI simulations. The results show similar improvement to that of forward simulations. The performance boost in the most capable GPU we used is about 50 to 90 times that of a serial computation. In the case of FWI simulations, the parallel computation in CPU using OpenMP with 16 threads has given a performance boost up to 11 times compared to the serial computation.

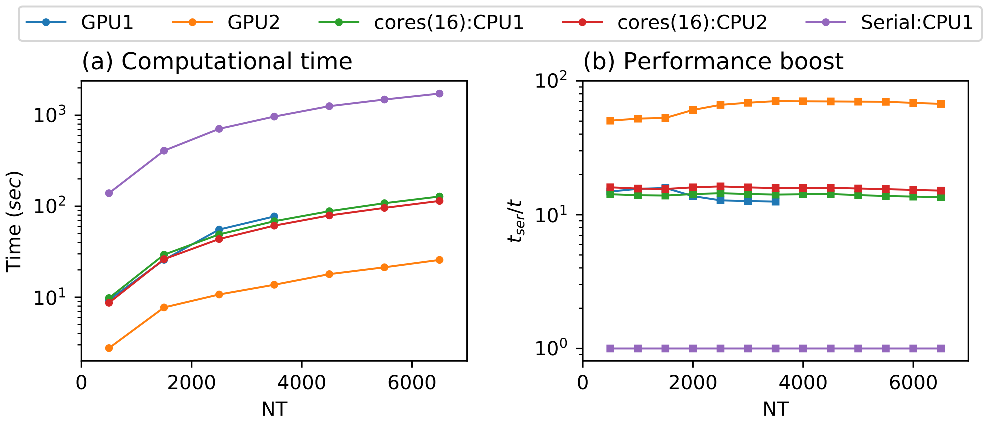

Figure 21. Comparison of computational time for 20 iterations and performance for variation of temporal grid size NT keeping spatial grid size constant ($NX = 201$, $NZ = 401$) in FWI simulations.

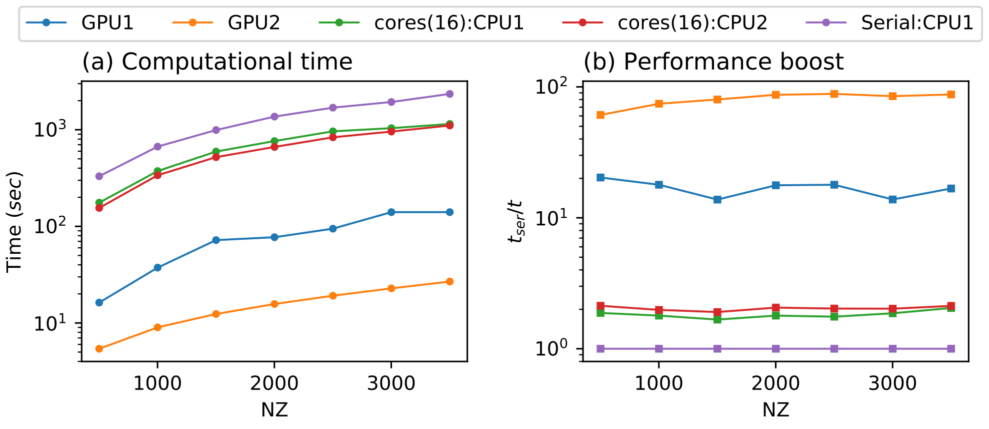

Figure 22. Comparison of computational for 20 iterations time and performance for variation of spatial grid size along Z direction NZ keeping spatial grid size along X direction and temporal grid constant ($NX = 201$, $NT = 1000$) in FWI simulations.

Overall, an average performance boost of about 80 times is observed in all the different modelling cases. It may not be significant for forward seismic models computed within minutes. However, for full-waveform inversion models, which may take hours and days for the whole iterations to compute, this may bring the computation of days to hours and hours to some minutes. This enhancement is based mainly on faster computation using parallelisation. A further enhancement can be made by using a more robust optimisation algorithm and machine learning methods for regression and optimisation studies [41,42].

6. Conclusions

Parallel programming GPU using CUDA C is implemented for in-plane seismic full waveform inversion, and the computational efficiency is compared to the serial and openMP parallel codes. The forward seismic wave propagation is studied for an earthen water reservoir dam with a crack and water table in the first step. It is important to study the forward model, as the FWI is based on an iterative computation of several forward models with a gradual improvement model and minimisation of the misfit. The influence of the crack and water table is visible in the results. In the next step, the full waveform inversion model is programmed in CUDA C, and a study is conducted for a spherical inclusion in full space. A balance of impedance ratio of the medium parameters and the frequency content of the input signal is essential for successful FWI computations. The high-frequency content in the time signal gives more detailed inversion results; however, there are more chances of the existence of local minima, and the model may converge elsewhere than the global minima. On the other hand, the low-frequency content in the time signal usually guides the model toward the global minima. However, the inverted results are blurred or less detailed, so the necessary details can be missed.

The computational efficiency of GPU parallelisation for the forward and full-waveform inversion models is studied compared to serial and CPU parallel codes. Comparing the computed results on different hardware and programming environments shows a significant reduction in computation time. The comparison is made for the change in the model size in time and space, which in the Finite Difference Method are grid sizes in Cartesian direction (NX, NZ) and the number of time steps (NT). For all the cases, the improvement in performance increases with an increase in the model dimension and remains constant after a certain threshold. The GPU parallel computation has shown up to the speed enhancement of about 60 to 100 times, whereas that CPU parallelisation in openMP (16 threads) is between 5 to 11 times that of the serial code. It can be concluded that GPU computation can significantly enhance the performance of in-plane seismic full waveform inversion. With the advancement of modern GPU, the FWI computation can be implemented into practical application by bringing the computation of the days into hours. It also provides further opportunities to implement robust optimisation and machine learning algorithms to enhance FWI computations.

Author Contributions: Conceptualization, M.B.B. and Z.H.R.; methodology, M.B.B.; software, M.B.B., M.A., A.H.A. and M.Z.; validation, M.B.B., M.A., A.H.A. and M.Z.; writing—original draft preparation, M.B.B., M.A. and Z.H.R.; writing—review and editing, Z.H.R., G.C. and F.W.; supervision, Z.H.R. and F.W.; project administration, M.B.B., Z.H.R. and F.W.; funding acquisition, F.W. All authors have read and agreed to the published version of the manuscript.

Funding: M.B.B., Z.H.R. and F.W. would like to thank project Betonbauwerken funded by Zentrales Innovationsprogramm Mittelstand, Germany.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data is available from corresponding author upon request.

Acknowledgments: We acknowledge financial support by Land Schleswig-Holstein within the funding programme Open Access Publikationsfonds.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Guasch, L.; Calderón Agudo, O.; Tang, M.X.; Nachev, P.; Warner, M. Full-waveform inversion imaging of the human brain. *NPJ Digit. Med.* **2019**, *3*, 1–12. [CrossRef]
2. Gauthier, O.; Virieux, J.; Tarantola, A. Two-dimensional nonlinear inversion of seismic waveforms: Numerical results. *Geophysics* **1986**, *51*, 1387–1403. [CrossRef]
3. Gauthier, O.; Virieux, J.; Tarantola, A. Nonlinear inversion of seismic reflection data. In *SEG Technical Program Expanded Abstracts 1985*; Society of Exploration Geophysicists: Houston, TX, USA, 2005; pp. 390–393. [CrossRef]
4. Mora, P. Nonlinear two-dimensional elastic inversion of multi offset seismic data. *Geophysics* **1987**, *52*, 1211–1228. [CrossRef]
5. Tarantola, A. Inversion of seismic reflection data in the acoustic approximation. *Geophysics* **1984**, *49*, 1259–1266. [CrossRef]
6. Tarantola, A. Linearized inversion of seismic reflection data. *Geophys. Prospect.* **1984**, *32*, 998–1015. [CrossRef]
7. Tarantola, A. Theoretical background for the inversion of seismic waveforms including elasticity and attenuation. *Pure Appl. Geophys.* **1988**, *128*, 365–399. [CrossRef]
8. Pratt, R.G.; Worthington, M.H. Inverse Theory Applied To Multi-Source Cross-Hole Tomography. *Geophys. Prospect.* **1990**, *38*, 287–310. [CrossRef]
9. Pratt, R.G. Inverse Theory Applied To Multi-Source Cross-Hole Tomography. *Geophys. Prospect.* **1990**, *38*, 311–329. [CrossRef]
10. Sourbier, F.; Operto, S.; Virieux, J.; Amestoy, P.; L'Excellent, J.Y. FWT2D: A massively parallel program for frequency-domain full-waveform tomography of wide-aperture seismic data-Part 2: Numerical examples and scalability analysis. *Comput. Geosci.* **2009**, *35*, 496–514. [CrossRef]
11. Sourbier, F.; Operto, S.; Virieux, J.; Amestoy, P.; L'Excellent, J.Y. FWT2D: A massively parallel program for frequency-domain full-waveform tomography of wide-aperture seismic data-Part 1: Algorithm. *Comput. Geosci.* **2009**, *35*, 487–495. [CrossRef]
12. Köhn, D. Time Domain 2D Elastic Full Waveform Tomography. Ph.D. Thesis, Kiel University, Kiel, Germany, 2011.
13. Yang, P.; Brossier, R.; Métivier, L.; Virieux, J.; Zhou, W. A Time-Domain Preconditioned Truncated Newton Approach to Visco-acoustic Multiparameter Full Waveform Inversion. *SIAM J. Sci. Comput.* **2018**, *40*, B1101–B1130. [CrossRef]
14. Wei, Z.F.; Gao, H.W.; Zhang, J.F. Time-domain full waveform inversion based on an irregular-grid acoustic modeling method. *Chin. J. Geophys.* **2014**, *57*, 586–594. (In Chinese) [CrossRef]
15. Charara, M.; Barnes, C.; Tarantola, A. Full waveform inversion of seismic data for a viscoelastic medium. *Methods Appl. Invers.* **2000**, *92*, 68–81. [CrossRef]
16. Fabien-Ouellet, G.; Gloaguen, E.; Giroux, B. Time domain viscoelastic full waveform inversion. *Geophys. J. Int.* **2017**, *209*, 1718–1734. [CrossRef]
17. Pan, W.; Innanen, K.A.; Wang, Y. SeisElastic2D: An open-source package for multiparameter full-waveform inversion in isotropic-, anisotropic- and visco-elastic media. *Comput. Geosci.* **2020**, *145*, 104586. [CrossRef]
18. Chen, J.B.; Cao, J. Modeling of frequency-domain elastic-wave equation with an average-derivative optimal method. *Geophysics* **2016**, *81*, T339–T356. [CrossRef]
19. Moczo, P. Introduction to Modeling Seismic Wave Propagation by the Finite-Difference Methods. In *Disaster Prevention Research Institute*; Kyoto University: Kyoto, Japan, 1998.
20. Moczo, P.; Robertsson, J.O.A.; Eisner, L. The finite-difference time-domain method for modeling of seismic wave propagation. *Adv. Geophys.* **2007**, *48*, 421–516.
21. Abubakar, A.; Pan, G.; Li, M.; Zhang, L.; Habashy, T.; van den Berg, P. Three-dimensional seismic full-waveform inversion using the finite-difference contrast source inversion method. *Geophys. Prospect.* **2011**, *59*, 874–888. [CrossRef]
22. Fang, J.; Chen, H.; Zhou, H.; Rao, Y.; Sun, P.; Zhang, J. Elastic Full-Waveform Inversion Based on GPU Accelerated Temporal Fourth-Order Finite-Difference Approximation. *Comput. Geosci.* **2020**, *135*. [CrossRef]
23. Wang, K.; Guo, M.; Xiao, Q.; Ma, C.; Zhang, L.; Xu, X.; Li, M.; Li, N. Frequency Domain Full Waveform Inversion Method of Acquiring Rock Wave Velocity in Front of Tunnels. *Appl. Sci.* **2021**, *11*, 6330. [CrossRef]
24. Komatitsch, D.; Martin, R. An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation. *Geophysics* **2007**, *72*, SM155–SM167. [CrossRef]
25. Martin, R.; Komatitsch, D.; Ezziani, A. An unsplit convolutional perfectly matched layer improved at grazing incidence for seismic wave propagation in poroelastic media. *Geophysics* **2008**, *73*, T51–T61. [CrossRef]
26. Martin, R.; Komatitsch, D. An unsplit convolutional perfectly matched layer technique improved at grazing incidence for the viscoelastic wave equation. *Geophys. J. Int.* **2009**, *179*, 333–344. [CrossRef]
27. Li, L.; Tan, J.; Schwarz, B.; Staněk, F.; Poiata, N.; Shi, P.; Diekmann, L.; Eisner, L.; Gajewski, D. Recent Advances and Challenges of Waveform-Based Seismic Location Methods at Multiple Scales. *Rev. Geophys.* **2020**, *58*, e2019RG000667. [CrossRef]
28. Jiang, J.; Zhu, P. Acceleration for 2D time-domain elastic full waveform inversion using a single GPU card. *J. Appl. Geophys.* **2018**, *152*, 173–187. [CrossRef]
29. Wang, B.; Gao, J.; Zhang, H.; Zhao, W. CUDA-based acceleration of full waveform inversion on GPU. In *SEG Technical Program Expanded Abstracts 2011*; SEG: Houston, TX, USA, 2012; pp. 2528–2533. [CrossRef]
30. Michéa, D.; Komatitsch, D. Accelerating a three-dimensional finite-difference wave propagation code using GPU graphics cards. *Geophys. J. Int.* **2010**, *182*, 389–402. [CrossRef]

31. Köhn, D.; Kurzmann, A. *DENISE User Manual*; Kiel University: Kiel, Germany, 2014.
32. Levander, A. Fourth-order finite-difference P-S. *Geophysics* **1988**, *53*, 1425–1436. [CrossRef]
33. Virieux, J. P-SV wave propagation in heterogeneous media: Velocity-stress finite-difference method. *Geophysics* **1986**, *51*, 889–901. [CrossRef]
34. Ross, P.E. Why CPU Frequency Stalled. *IEEE Spectrum* **2008**, *45*, 72. [CrossRef]
35. Ramanathan, R. White Paper Intel® Multi-Core Processors: Making the Move to Quad-Core and Beyond. In *White Paper From Intel 424 Corporation*; Intel: Santa Clara, CA, USA, 2006.
36. Robey, R.; Zamora, Y. *Parallel and High Performance Computing*; Simon and Schuster: New York, NY, USA, 2021.
37. Memeti, S.; Li, L.; Pllana, S.; Kołodziej, J.; Kessler, C. Benchmarking OpenCL, OpenACC, OpenMP, and CUDA: Programming productivity, performance, and energy consumption. In Proceedings of the 2017 Workshop on Adaptive Resource Management and Scheduling for Cloud Computing, Washington, DC, USA, 28 July 2017; pp. 1–6.
38. Chandra, R.; Dagum, L.; Kohr, D.; Maydan, D.; McDonald, J.; Menon, R. *Parallel Programming in OpenMP*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2001.
39. Gimenes, T.L.; Pisani, F.; Borin, E. Evaluating the Performance and Cost of Accelerating Seismic Processing with CUDA, OpenCL, OpenACC, and OpenMP. In Proceedings of the 2018 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2018, Vancouver, BC, Canada, 21–25 May 2018; pp. 399–408. [CrossRef]
40. Santamarina, J.C.; Rinaldi, V.A.; Fratta, D.; Klein, K.; Wang, Y.H.; Cho, G.C.; Cascante, G. *A Survey of Elastic and Electromagnetic Properties of Near-Surface Soils*; SEG: Houston, TX, USA, 2009.
41. Rizvi, Z.H.; Akhtar, S.J.; Haider, H.; Follmann, J.; Wuttke, F. Estimation of seismic wave velocities of metamorphic rocks using artificial neural network. *Mater. Today Proc.* **2020**, *26*, 324–330. [CrossRef]
42. Wuttke, F.; Lyu, H.; Sattari, A.S.; Rizvi, Z.H. Wave based damage detection in solid structures using spatially asymmetric encoder–decoder network. *Sci. Rep.* **2021**, *11*, 20968. [CrossRef] [PubMed]