UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA
GRADUAÇÃO EM ENGENHARIA COMPUTACIONAL

ANTÔNIO JOSÉ DE MEDEIROS FILHO

Numerical Dispersion Suppression in the Two-Dimensional Acoustic Wave Equation Using PETSc and an Improved Pix2Pix Algorithm

Juiz de Fora

2025

ANTÔNIO JOSÉ DE MEDEIROS FILHO

Numerical Dispersion Suppression in the Two-Dimensional Acoustic Wave Equation Using PETSc and an Improved Pix2Pix Algorithm

Trabalho de conclusão de curso de Graduação em Engenharia Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de bacharel em Engenharia Computacional.

Orientador: Prof. Dr. José Jerônimo Camata

Juiz de Fora

2025

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

MEDEIROS, ANTÔNIO.
Numerical Dispersion Suppression in the Two-Dimensional Acoustic Wave Equation Using PETSc and an Improved Pix2Pix Algorithm / ANTÔNIO JOSÉ DE MEDEIROS FILHO. – 2025.
50 f. : il.

Orientador: José Jerônimo Camata
Trabalho de Conclusão de Curso (graduação) – Universidade Federal de Juiz de Fora, Faculdade de Engenharia. Graduação em Engenharia Computacional, 2025.

1. Equação de Ondas. 2. Método de Diferenças Finitas. 3. Supressão de Dispersão Numérica. I. Camata, José, orient. II. Título.

ANTÔNIO JOSÉ DE MEDEIROS FILHO

Numerical Dispersion Suppression in the Two-Dimensional Acoustic Wave Equation Using PETSc and an Improved Pix2Pix Algorithm

Trabalho de conclusão de curso de Graduação em Engenharia Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de bacharel em Engenharia Computacional.

Aprovada em 22 de Agosto de 2025

BANCA EXAMINADORA

Prof. Dr. José Jerônimo Camata - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Bernardo Martins Rocha
Universidade Federal de Juiz de Fora

Prof. Dr. Joventino de Olveira Marques
Universidade Federal de Juiz de Fora

Dedico este trabalho aos meus pais, que sempre me apoiaram, ao meu orientador, Prof. José Camata, pelo apoio no desenvolvimento do trabalho, à minha namorada pelo incentivo e suporte, e aos colegas de curso que, de alguma forma, contribuíram para este trabalho.

ABSTRACT

This work addresses the suppression of numerical dispersion in the solution of the two-dimensional acoustic wave equation using the Portable, Extensible Toolkit for Scientific Computation (PETSc) in combination with an improved Pix2Pix algorithm. The proposed approach begins with the numerical solution of the wave equation via a Finite Difference Method (FDM) solver implemented with PETSc, enabling efficient parallel computation. Coarse-grid simulations, which inherently exhibit numerical dispersion, are generated and compared against reference solutions obtained from refined-grid simulations. The coarse-grid data are used as inputs and the refined-grid solutions as target outputs for training a conditional Generative Adversarial Network (cGAN) based on an enhanced Pix2Pix architecture. Several parameter configurations are investigated to determine the optimal training setup, including learning rate, number of epochs, and certainty coefficients for both the L_1 and Sobel loss terms. The trained model is then applied to different velocity models to assess its generalization capabilities. Performance evaluation is conducted by computing the Normalized Root Mean Square Error (NRMSE) between the cGAN-predicted fields and the reference solutions. The results demonstrate that the improved Pix2Pix model can significantly reduce numerical dispersion effects, producing wavefields that closely match the accuracy of high-resolution simulations while retaining the lower computational cost of coarse-grid calculations.

Keywords: Wave Equation, Finite Difference Method, PETSc, Numerical Dispersion Supression, pix2pix

RESUMO

Este trabalho aborda a supressão da dispersão numérica na solução da equação da onda acústica bidimensional utilizando o Portable, Extensible Toolkit for Scientific Computation (PETSc) em combinação com um algoritmo Pix2Pix aprimorado. A abordagem proposta inicia-se com a solução numérica da equação de ondas por meio de um solver de Diferenças Finitas (FDM) implementado com o PETSc, permitindo computação paralela eficiente. Simulações em malhas grosseiras, que apresentam dispersão numérica intrínseca, são geradas e comparadas com soluções de referência obtidas em simulações com malhas refinadas. Os dados da malha grosseira são usados como entradas e as soluções de malha refinada como saídas-alvo para o treinamento de uma Rede Generativa Adversarial condicional (cGAN) baseada em uma arquitetura Pix2Pix aprimorada. Diversas configurações de parâmetros são investigadas para determinar o melhor cenário de treinamento, incluindo taxa de aprendizado (learning rate), número de épocas e coeficientes de ponderação para os termos de perda L_1 e Sobel. O modelo treinado é então aplicado a diferentes modelos de velocidade para avaliar sua capacidade de generalização. A avaliação de desempenho é realizada por meio do cálculo do Erro Quadrático Médio Normalizado (NRMSE) entre os campos previstos pela cGAN e as soluções de referência. Os resultados demonstram que o modelo Pix2Pix aprimorado pode reduzir significativamente os efeitos da dispersão numérica, produzindo campos de onda que se aproximam da precisão das simulações de alta resolução, ao mesmo tempo em que mantém o menor custo computacional das simulações em malhas grosseiras.

Palavras-chave: Equação de Ondas, Método de Diferenças Finitas, PETSc, Supressão de Dispersão Numérica, Pix2Pix.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

1 INTRODUCTION

The accurate simulation of seismic wave propagation plays a pivotal role in geophysical exploration, as it enables the virtual reconstruction of subsurface structures without the need for direct sampling. This capability is particularly valuable in the oil and gas industry, where seismic imaging guides both exploration and reservoir characterization [6].

Among the various numerical techniques available, the Finite Difference Method (FDM) [13] is one of the most widely adopted for acoustic wave propagation due to its conceptual simplicity and ease of implementation. Despite its popularity in both academia and industry, FDM is inherently susceptible to numerical dispersion, especially when coarse spatial grids are used to reduce computational costs. This dispersion can distort simulated wavefields and compromise the reliability of the results.

Recent developments have shown the potential of deep learning to address such limitations in scientific computing. For instance, Xu et al. [19] applied the *pix2pix* model, based on conditional GANs, to remove temporal dispersion in elastic wave modeling, enhancing reconstruction accuracy through the use of Sobel operators in the loss function. Yan [20] extended this concept to suppress spatial dispersion in finite difference modeling, incorporating architectural modifications to improve robustness. Han [7] proposed a semi-supervised framework combining convolutional neural networks (CNNs) and gated recurrent units (GRUs), leveraging transfer learning to generalize the trained model to different simulation setups.

Other works have explored hybrid approaches, integrating neural networks directly with traditional FDM solvers. Gadylshin [5] and Kaur [11] investigated the use of deep learning-based post-propagation filters to mitigate dispersion artifacts, showing that such techniques can complement numerical optimizations. In parallel, Koene [12] proposed mathematical transformations to add or remove temporal dispersion from synthetic seismograms, offering a purely numerical baseline for comparison. More recently, Ji [10] analyzed the detrimental effects of accumulated dispersion on wavefront fidelity and proposed machine learning-assisted finite difference schemes to improve accuracy.

In this context, image-to-image translation frameworks such as *pix2pix* [9] stand out as promising tools for enhancing numerical solutions without increasing the computational burden of the underlying solver. This convergence of numerical modeling and data-driven post-processing forms the foundation for the methodology explored in the present work.

1.1 Related Work

Several strategies have been proposed in the literature to reduce numerical dispersion and improve the accuracy of finite difference simulations for the wave equation. One common approach is to modify the discretization scheme itself. For instance, Zhou et al.

[23] introduced a high-order Padé approximation that effectively suppresses numerical dispersion in two-dimensional acoustic and elastic modeling, enabling more accurate wavefield representations without excessively refining the computational grid. Similarly, Yang et al. [21] developed the Nearly Analytic Discrete Method (NADM), which further reduces dispersion by exploiting an approximation that closely matches the continuous solution behavior.

Another line of research focuses on optimization techniques to fine-tune finite difference stencils. Vanga et al. [18], for example, employed a genetic algorithm to determine optimal coefficients for second-order spatial derivatives, achieving reduced dispersion while maintaining computational efficiency. These advances highlight a consistent trend: by refining the numerical formulation or its parameters, it is possible to improve the fidelity of wave propagation models without prohibitive increases in computational cost.

In parallel, emerging machine learning approaches—particularly those leveraging deep learning for post-processing—are beginning to offer an alternative path to dispersion suppression. Instead of altering the underlying solver, these methods aim to learn corrective mappings that transform dispersion-affected wavefields into higher-quality representations. Within this context, image-to-image translation frameworks, such as the *pix2pix* model [9], have demonstrated promising results in related scientific domains, suggesting that they could complement or even replace certain numerical optimization techniques. The works of Xu [19] and Yan [20] provide concrete evidence of the effectiveness of this approach in seismic modeling, and serve as a starting point for the methodology proposed here.

1.2 Objectives

Building on the recent convergence between numerical modeling and deep learning, this work investigates the application of the *pix2pix* algorithm to suppress numerical dispersion in finite difference simulations of seismic wave propagation. The central hypothesis is that a data-driven post-processing stage can enhance the accuracy of FDM solutions obtained on coarser grids, thus reducing computational demands without compromising solution quality.

To explore this hypothesis, an acoustic wave propagation solver is developed using PETSc, leveraging its parallel computing capabilities to efficiently generate large-scale datasets for training and validation. The study aims not only to evaluate the effectiveness of the *pix2pix* model in correcting dispersion artifacts, but also to quantify the trade-offs between numerical accuracy and computational cost.

The specific objectives are:

- Implement a parallelized acoustic wave propagation solver for anisotropic media using PETSc.

- Achieve fourth-order spatial and second-order temporal accuracy through Taylor series expansions.
- Integrate a deep learning post-processing stage, inspired by the *pix2pix* architecture, for numerical dispersion suppression.
- Evaluate the balance between computational savings from coarser grids and the accuracy improvements obtained through post-processing.

1.3 Organization of the Work

This document is organized as follows: In Chapter 2, the governing equations and boundary conditions of the acoustic wave problem are presented, along with the source function that initiates wave propagation. Chapter 3 describes the PETSc-based acoustic wave solver, including its finite difference formulation and parallelization strategy.

Chapter 4 focuses on the proposed dispersion suppression approach, detailing the *pix2pix* architecture, dataset preparation, and training procedure. Finally, Chapter 5 presents and discusses the results, highlighting the effectiveness of the *pix2pix* post-processing, the accuracy gains achieved, and the implications for reducing computational costs in seismic modeling.

2 WAVE EQUATION

In geophysical applications, particularly in seismic exploration and monitoring, the wave equation plays a central role in simulating the propagation of elastic or acoustic waves through the Earth's subsurface. By modeling how seismic waves travel and interact with geological structures, it is possible to infer information about the composition, layering, and discontinuities in the subsurface. This is essential for tasks such as identifying oil and gas reservoirs, assessing earthquake hazards, and monitoring CO_2 storage sites. The reflections and refractions produced at interfaces between different materials are key observables in seismic surveys and are predicted using numerical solutions to the wave equation [8].

2.1 Two-dimensional Equation

The wave equation is a second-order partial differential equation (PDE) and is one of the fundamental equations in physics that describes the propagation of waves through a medium. It is notably used to describe multiple natural and technological phenomena, such as sound and seismic waves, the latter being the simulation target for the solver that will be presented.

In this work, particular emphasis is given to seismic wave propagation, which is the focus of the numerical solver to be presented. To simplify the development and analysis of the numerical method, the two-dimensional form of the wave equation is adopted, although a three-dimensional extension of the solver is currently under development.

A two-dimensional Wave Equation for the pressure field u with spatial dimensions x (length) and z (height) and time dimension t takes the form:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial z^2} = \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} + f(x, z, t) \tag{2.1}$$

where c is the wave propagation velocity in the medium and f is the source function.

2.2 Source Function

The source term represents the origin of the wave and is responsible for initiating its propagation. Sources may be natural or artificial: in the context of seismic waves, natural sources include earthquakes and volcanic activity, whereas artificial sources are typically controlled explosions used in geophysical surveys.

The wave originates at a specific spatial location (x_s, z_s), referred to as the epicenter, where the source function is applied. To model the seismic pulse, the Ricker wavelet [15] — a commonly used compact and zero-mean wavelet in seismic modeling — is adopted. It corresponds to the second derivative of a Gaussian function and is expressed as:

$$f(t) = [1 - 2(\pi f_p t)^2]exp[-(\pi f_p t)^2] \tag{2.2}$$

where f_p is the peak frequency and t is the time relative to the pulse center.

2.3 Velocity Fields

A homogeneous medium is enough for most cases where the goal is to evaluate the accuracy and efficiency of a numerical solution for the wave equation PDE. However, in reality, the medium through which waves propagate is more complex, especially when dealing with the propagation of seismic waves in a medium that is the Earth's interior, which possesses diverse physical properties, making the wave propagation velocity c not a constant but a function of the spatial dimensions. A portion of the seismic wave is reflected at the interfaces between these velocity fields, enabling geophysicists to capture these reflections and analyze the underground.

2.4 Boundary Conditions

A critical aspect of simulating wave propagation in finite domains is the treatment of artificial reflections at the boundaries, which can significantly distort the solution. To mitigate such effects, *absorbing boundary conditions* are applied to simulate an open or infinite domain.

One of the widely used techniques is the *Reynolds non-reflective boundary condition* [14], which introduces first-order equations at the domain boundaries to absorb outgoing waves. The conditions for each boundary are as follows:

- Upper edge
$$-\frac{\partial u}{\partial z} + \frac{1}{c}\frac{\partial u}{\partial t} = 0 \tag{2.3}$$
- Bottom edge
$$\frac{\partial u}{\partial z} + \frac{1}{c}\frac{\partial u}{\partial t} = 0 \tag{2.4}$$
- Left side
$$-\frac{\partial u}{\partial x} + \frac{1}{c}\frac{\partial u}{\partial t} = 0 \tag{2.5}$$
- Right side
$$\frac{\partial u}{\partial x} + \frac{1}{c}\frac{\partial u}{\partial t} = 0 \tag{2.6}$$

Another way to reduce reflection at the borders is the damping boundary condition [3] where a damping function is applied to the wave field in a range close to the boundaries. The mitigation function is given by the following:

$$W(d) = exp[-(fat \cdot (N_a - d)^2)] \tag{2.7}$$

where d is the distance in points from the boundary, fat is the damping coefficient and N_a is the size of the layer where the damping function is applied. In this work both boundary conditions will be applied combined.

3 FINITE DIFFERENCE METHOD IMPLEMENTATION

The Acoustic Wave Equation, as is the case for most PDEs, can be solved using a wide range of numerical methods, each of them having its advantages and disadvantages. Although numerical methods only provide an approximate solution, they can be applied in various complex problems while having a easier implementation on computers.

An example of this is the Finite Difference Method (FDM for short), which addresses the solution of PDEs by discretizing the problem domain into a finite grid of points. The primary advantage of FDM lies in its simple implementation, allowing parallel computing without much effort, whereas its main limitations are its restriction to regular grids, which preclude the possibility of obtaining more precise results for specific portions of the domain, and the fact that the computational usage may increase substantially for more refined grids [4].

3.1 Finite Difference Method

3.1.1 Discretization of the domain

The Finite Difference Method, as with all of the numerical methods, requires a finite, discrete domain to find an approximate solution. For instance, consider a two-dimensional rectangular spatial domain with x and z spatial dimensions, with length X and height Z. These two dimensions may be discretized into a finite grid of points as illustrated in Figure 1.

The FDM solves the differential equation by replacing the derivative terms with discrete difference terms through Taylor series expansion. Therefore, coordinates (x, z) of a given point can be in discrete form represented by:

$$\begin{cases} x = i\Delta x, & i = 1, 2, 3, ..., N_x; \\ z = j\Delta z, & j = 1, 2, 3, ..., N_z; \end{cases} \tag{3.1}$$

where Δx is given by X/N_x and Δz is given by Z/N_z, with N_x and N_z being the number of points in x and z respectively. The quality of the solution and the computational usage are inversely proportional to the size of Δx and Δz, and, as will be discussed in the latter chapters, they are important to ensure numerical stability and avoid numerical dispersion.

3.1.2 Taylor Series Expansion

The Taylor series of an infinitely differentiable function $f(x)$ at a given point a is an infinite sum given by the function's derivatives:

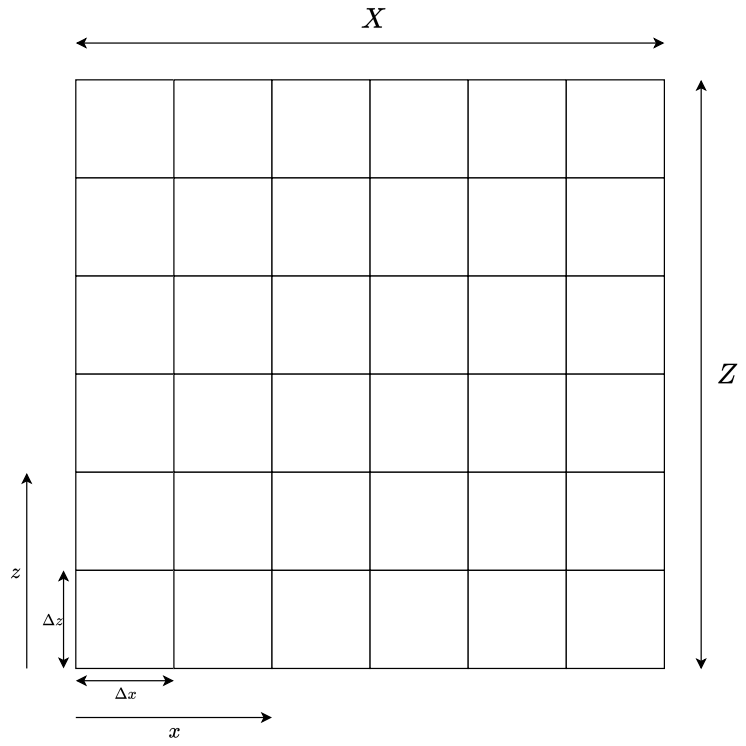


Figure 1 – Representation of the discrete two-dimensional grid.

$$f(a) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + \frac{f^{(1)}(a)}{1!}(x-a) + \frac{f^{(2)}(a)}{2!}(x-a)^2 + ... \tag{3.2}$$

where $f^{(n)}$ is the n-th derivative of f.

In the context of numerical methods, Taylor expansions are used to obtain an approximation for derivatives in nearby function values (*i.e* $f(x+\Delta x)$ where Δx is a small step size) by truncating higher-order terms. This introduces a truncation error, denoted by $\mathcal{O}(\Delta x)^n$, where n is the order of accuracy of the approximation, and is defined by the difference between the real value and the approximation.

For instance, the first derivative of a function $f(x)$ can be approximated by the 1st-order **forward difference**:

$$f(x+\Delta x) = f(x) + \Delta x f^1(x) + \frac{\Delta x^2}{2!} f^2(x) + \frac{\Delta x^3}{3!} f^3(x) + ... \tag{3.3}$$

truncating the Taylor series for $n > 1$

$$f(x+\Delta x) = f(x) + \Delta x f^1(x) + \mathcal{O}(\Delta x) \tag{3.4}$$

rearranging to isolate $f^{(1)}$:

$$f^1(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} + \mathcal{O}(\Delta x) \tag{3.5}$$

Then, the derivative approximations can be replaced in the original differential equation to obtain the numerical solution.

3.1.3 2nd-order approximation for $u^{(2)}(x)$

To find the 2nd-order approximation for $u^2(x)$, **central-differences** will be employed. Expading $u(x \pm \Delta x)$ trough Taylor series:

$$u(x+\Delta x) = u(x)+\Delta x\cdot u^{(1)}(x)+\frac{(\Delta x)^2}{2!}\cdot u^{(2)}(x)+\frac{(\Delta x)^3}{3!}\cdot u^{(3)}(x)+\frac{(\Delta x)^4}{4!}\cdot u^{(4)}(x)+... \tag{3.6}$$

and,

$$u(x-\Delta x) = u(x)-\Delta x\cdot u^{(1)}(x)+\frac{(\Delta x)^2}{2!}\cdot u^{(2)}(x)-\frac{(\Delta x)^3}{3!}\cdot u^{(3)}(x)+\frac{(\Delta x)^4}{4!}\cdot u^{(1)}(x)-... \tag{3.7}$$

Adding (3.6) and (3.7):

$$u(x+\Delta x) + u(x-\Delta x) = 2u(x) + (\Delta x)^2 \cdot u^{(2)}(x) + \frac{(\Delta x)^4}{12} \cdot u^{(4)}(x) + ... \tag{3.8}$$

Truncating the higher-order terms for $\mathcal{O}(\Delta x^2)$, the second derivative is isolated, resulting in:

$$u^{(2)}(x) \approx \frac{u(x+\Delta x) - 2u(x) + u(x-\Delta x)}{(\Delta x)^2} \tag{3.9}$$

For simplicity, $u(x)$ can be represented as u_i and $u(x+\Delta x)$ as u_{i+1}:

$$u^{(2)}(x) \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} \tag{3.10}$$

3.1.4 4th-order approximation for $u^{(2)}(x)$

To obtain the 4th-order approximation, more points can be included, considering a 5-point stencil, demonstrated by figure 2. The Taylor series expansion for $u(x-2\Delta x)$ and $u(x+2\Delta x)$ are given by:

$$u(x+2\Delta x) = u(x)+2\Delta x\cdot u^{(1)}(x)+\frac{(2\Delta x)^2}{2!}\cdot u^{(2)}(x)+\frac{(2\Delta x)^3}{3!}\cdot u^{(3)}(x)+\frac{(2\Delta x)^4}{4!}\cdot u^{(4)}(x)+\mathcal{O}(\Delta x^4) \tag{3.11}$$

and,

$$u(x-2\Delta x) = u(x)-2\Delta x\cdot u^{(1)}(x)+\frac{(2\Delta x)^2}{2!}\cdot u^{(2)}(x)-\frac{(2\Delta x)^3}{3!}\cdot u^{(3)}(x)+\frac{(2\Delta x)^4}{4!}\cdot u^{(4)}(x)+\mathcal{O}(\Delta x^4) \tag{3.12}$$

adding (3.11) and (3.12):

$$u(x+2\Delta x) + u(x-2\Delta x) = 2u(x) + 4(\Delta x)^2 \cdot u^{(2)}(x) + \frac{16(\Delta x)^4}{12} \cdot u^{(4)}(x) + \mathcal{O}(\Delta x^4) \tag{3.13}$$

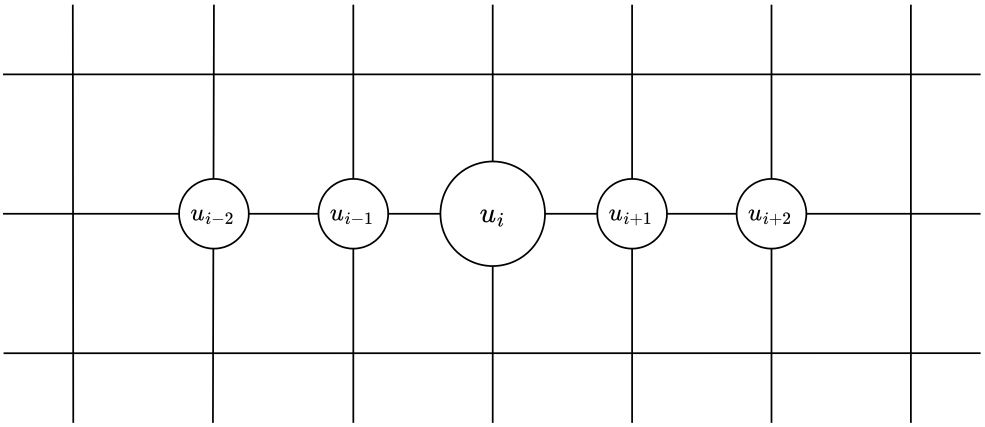


Figure 2 – Representation of the 5-point stencil.

To eliminate the $u^{(4)}(x)$ term, equation (3.8) can be multiplied by 16 and subtracted by (3.13), resulting in:

$$16(u(x+\Delta x)+u(x-\Delta x)-(u(x+2\Delta x)+u(x-2\Delta x)) = 30u(x)+12(\Delta x)^2u^{(2)}(x)+\mathcal{O}(\Delta x^4) \tag{3.14}$$

Isolating $u^{(2)}(x)$:

$$u^{(2)}(x) = \frac{1}{12\Delta x^2}[-u(x-2\Delta x)+16u(x-\Delta x)-30u(x)+16u(x+\Delta x)-u(x+2\Delta x)]+\mathcal{O}(\Delta x^4) \tag{3.15}$$

or simply:

$$u^{(2)}(x) = \frac{1}{12\Delta x^2}[-u_{i-2} + 16u_{i-1} - 30u_i + 16u_{i+1} - u_{i+2}] + \mathcal{O}(\Delta x^4) \tag{3.16}$$

3.2 FDM for the Wave Equation

3.2.1 Domain Definition and Discretization

To solve the wave equation through the FDM, the domain needs to be specified first. For that, a two-dimensional grid similar to the one presented in chapter **3.1.1** will be considered, but now with another dimension, which is time. The coordinates (x, z, t) will be discretely represented by:

$$\begin{cases} x = i\Delta x, & i = 1, 2, 3, ..., N_x; \\ z = j\Delta z, & j = 1, 2, 3, ..., N_z; \\ t = k\Delta t, & k = 1, 2, 3, ..., N_t; \end{cases} \tag{3.17}$$

where Δx, Δz and Δt are the step intervals, determined by:

$$\Delta x = \frac{X}{N_x},\ \Delta z = \frac{Z}{N_z},\ \Delta t = \frac{T}{N_t} \tag{3.18}$$

where X is the length of the domain, Z is the height, T is the total time duration and N_x, N_z and N_t are the number of points in dimensions x, z and t, respectively. For this work, a uniform grid was considered, meaning that $\Delta x = \Delta z = h$.

3.2.2 Approximating the two-dimensional Wave Equation

As mentioned in chapter 2.1, the two-dimensional Wave Equation takes the form of:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial z^2} = \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} + f(x,z,t) \tag{3.19}$$

where f is the source function, given by:

$$f(x,z,t) = \begin{cases} [1 - 2(\pi f_p t)^2] exp[-(\pi f_p t)^2], & \text{if } x = x_s \text{ and } z = z_s \\ 0, & \text{otherwise} \end{cases} \tag{3.20}$$

given the source origin point (x_s, z_s). A peak frequency $f_p = 40Hz$ will be considered.

To solve the wave equation through FDM, the 2nd-order (3.10) and 4th-order (3.16) approximations will be employed for the time and spatial dimensions, respectively. Therefore:

$$\left(\frac{\partial^2 u}{\partial x^2}\right)^k_{i,j} \approx \frac{1}{12\Delta x^2}[-u^k_{i-2,j} + 16u^k_{i-1,j} - 30u^k_{i,j} + 16u^k_{i+1,j} - u^k_{i+2,j}] \tag{3.21}$$

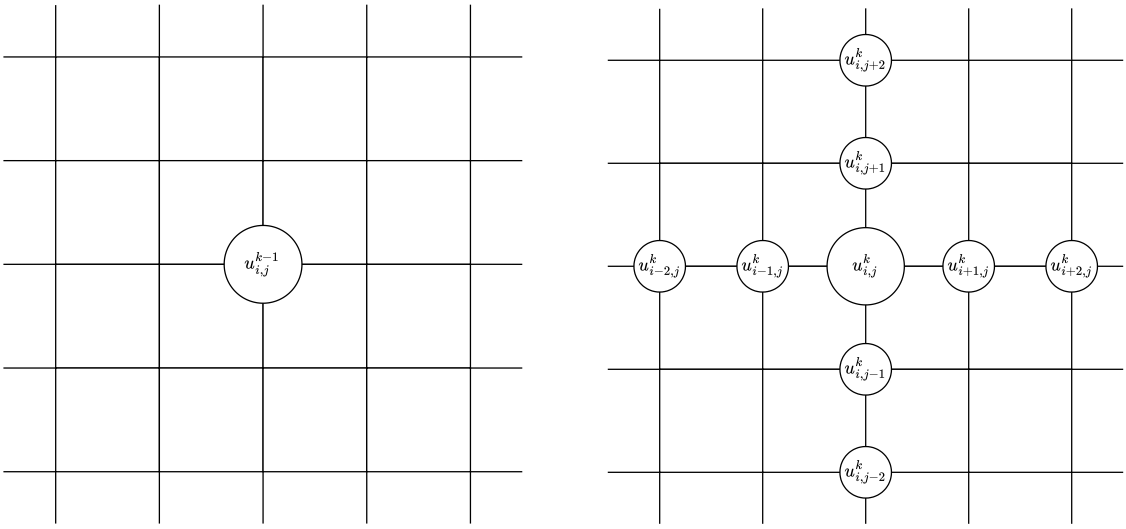
$$\left(\frac{\partial^2 u}{\partial z^2}\right)^k_{i,j} \approx \frac{1}{12\Delta z^2}[-u^k_{i,j-2} + 16u^k_{i,j-1} - 30u^k_{i,j} + 16u^k_{i,j+1} - u^k_{i,j+2}] \tag{3.22}$$

$$\left(\frac{\partial^2 u}{\partial t^2}\right)^k_{i,j} \approx \frac{u^{k-1}_{i,j} - 2u^k_{i,j} + u^{k+1}_{i,j}}{(\Delta t)^2} \tag{3.23}$$

Finally, replacing the second derivatives with the corresponding approximations leads to:

$$\begin{aligned} u^{k+1}_{i,j} = \frac{1}{12}\left(c \cdot \frac{\Delta t}{h}\right)^2 \Big[&- \Big(u^k_{i-2,j} + u^k_{i,j-2}\Big) + 16\Big(u^k_{i-1,j} + u^k_{i,j-1}\Big) \\ &- 60u^k_{i,j} + 16\Big(u^k_{i+1,j} + u^k_{i,j+1}\Big) - \Big(u^k_{i+2,j} + u^k_{i,j+2}\Big)\Big] \\ &+ 2u^k_{i,j} - u^{k-1}_{i,j} + f(i,j,k) \end{aligned} \tag{3.24}$$

Figure 3 demonstrates the **iterative stencil loop** that will be implemented in the solver.



(a) $t = (k-1)\Delta t$ (b) $t = k\Delta t$

Figure 3 – Visualization of the points used to calculate $u_{i,j}^{k+1}$

3.2.3 Boundary and Initial Conditions

Since the 2nd-order approximation was employed for the time dimension t, an initial condition for $u(x, z, t)$ must be set at $k = 0$ and $k = 1$ to obtain u at $k = 2$. The initial condition adopted was $u = 0$, for $i = 0, 1, ..., N_x$ and $j = 0, 1, ..., N_z$.

For Reynolds' boundary condition, a 1st-order forward difference was used to approximate each of the partial derivatives defined in chapter 2.4:

- Upper edge
$$u_{i,j}^{k+1} = u_{i,j}^{k} + C \cdot (u_{i,j+1}^{k} - u_{i,j}^{k}) \tag{3.25}$$
- Bottom edge
$$u_{i,j}^{k+1} = u_{i,j}^{k} - C \cdot (u_{i,j}^{k} - u_{i,j-1}^{k}) \tag{3.26}$$
- Left side
$$u_{i,j}^{k+1} = u_{i,j}^{k} + C \cdot (u_{i+1,j}^{k} - u_{i,j}^{k}) \tag{3.27}$$
- Right side
$$u_{i,j}^{k+1} = u_{i,j}^{k} - C \cdot (u_{i,j}^{k} - u_{i-1,j}^{k}) \tag{3.28}$$

3.2.4 Numerical Dispersion and Stability

Numerical dispersion refers to the artificial spreading of wavefronts in the numerical solution, causing different frequencies to propagate at different velocities. This is distinct from physical dispersion, which occurs due to material properties. In FDM, numerical dispersion arises because the discrete grid cannot perfectly represent continuous wave

propagation, especially for high-frequency components. This implies that the grid spacing is limited by:

$$h \leq \frac{c_{\min}}{k f_p} \tag{3.29}$$

where $c_{\min}$ is the minimum velocity from the velocity model, f_p is the source peak frequency and k represents the maximum number of samples by wave length.

Numerical stability ensures that errors do not grow unbounded over time. For explicit FDM schemes, stability is governed by the Courant-Friedrichs-Lewy (CFL) condition, which in uniform grids is determined by:

$$\Delta t \leq \frac{h}{\mu c_{\max}} \tag{3.30}$$

where $c_{\max}$ is the maximum velocity from the velocity model, and μ is a constant that depends on the physical properties of the medium.

In general, these conditions are satisfied by simply using a more refined grid, but this can increase computational usage significantly. In the next chapter, another alternative to suppress the numerical dispersion based on a neural network will be presented.

3.3 Computational Implementation

3.3.1 PETSc Library and Parallelization

PETSc stands for Portable, Extensible Toolkit for Scientific Computation and is a library designed to help implement high-performance scientific applications modeled by PDEs [2]. It is available for C, C++, Fortran, and Python programming languages, and supports parallelization through MPI and GPU (CUDA, OpenCL, etc), as well as hybrid MPI-GPU parallelism.

From the set of tools PETSc offers, one of the most important is the DM (Data Management) object, which is a manager for abstract grid-like objects. More specifically, there is the DMDA, which is intended for rectangular grids, working well with solvers that use a Finite Difference approach.

For example, the following piece of code demonstrates how to instantiate a DMDA object that creates vectors to store the current and next solutions in a given iteration:

```
DMDACreate2d(..., &da);
...
DMCreateGlobalVector(da, &u_current);
DMCreateLocalVector(da, &u_current_local);
```

```
DMCreateGlobalVector(da, &u_next);
DMCreateLocalVector(da, &u_next_local);
```

Note that, for each vector (`u_current` and `u_next`), a Global and a Local vector are created. The Global Vector represents the entire grid distributed across all MPI processes, while the Local Vector refers to the portion of the grid that each process will work with.

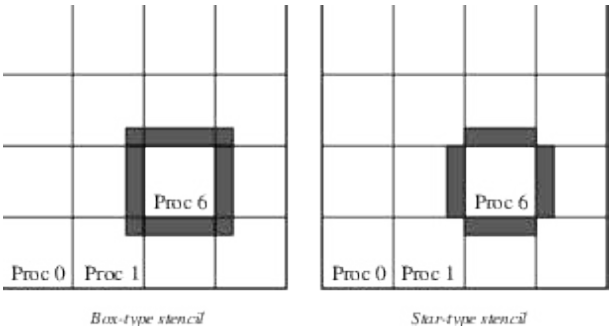


Figure 4 – Ghost points for process 6 with two Stencil type examples [2].

PETSc also provides an abstraction layer for MPI, handling most of the complexity when dealing with parallelization. For instance, the stencil computation requires cross-process data access at the local vector boundaries, as shown by figure 4. These are called ghost points and PETSc handles this automatically through the `DMDASetStencilWidth()` method and `DMDAStencilType` parameter. This ensures that the local vectors in each process are extended so that when calculating the stencil, these points are available locally in the process without the need to access the global vector. In the case of the 4th-order finite difference approximation, a stencil width of 2 is required, with the *Star-type* stencil (`DMDAStencilType::DMDA_STENCIL_STAR`).

To load the local vectors, the `DMGlobalToLocalBegin()` and `DMGlobalToLocalEnd()` methods can be used. Since the interval to iterate over the grid using the i and j indexes is different for each process, the `DMDAGetCorners()` needs to be used to obtain the start and end for the spatial loops. After updating the local vectors, the changes can be transferred to the global vector through the `DMLocalToGlobalBegin()` and `DMLocalToGlobalEnd()` methods.

3.3.2 VTK ImageData format

The default data format adopted for both input and output by the solver was VTK's ImageData format [17]. The ImageData format was designed to represent data that is regularly spaced in a topological and geometrical array, making it suitable for the FDM. VTK uses XML to describe the data structure.

```
<?xml version="1.0"?>
<VTKFile type="ImageData" version="0.1" byte_order="LittleEndian">
  <ImageData WholeExtent="0 299 0 299 0 0" Origin="0 0 0" Spacing="10 10 0">
    <Piece Extent="0 299 0 299 0 0">
      <PointData Scalars="velocity">
        <DataArray type="Float32" Name="velocity" format="ascii">
            ...
        </DataArray>
      </PointData>
    </Piece>
  </ImageData>
</VTKFile>
```

The code snippet demonstrates the structure of the *.vti* file. The `<ImageData>` section describes the dimensions of the dataset, in this case 300×300, as well as the origin coordinates and the spacing, which is the value for Δx and Δz. The `<Piece>` section specifies the portion from the data set that the file contains, in this case matching the `<ImageData>` extent, meaning that it's a single piece. The actual data is included inside the `<DataArray>` section, in this case in ASCII format (human readable).

3.3.3 Solver Overview

A solver for the two-dimensional wave equation using FDM was implemented with C++ and the PETSc library. The source code and documentation are available at the github repository *acwv-solver*[1]. The repository also contains a Python script to generate the example velocity models.

The solver is called through the command line and expects a set of parameters, those being:

- Input file (path for the input *.vti* file for the velocity matrix);
- Output directory (path for the directory to save the computed wave field *.vti* files for each timestep);
- Horizontal position of the source (x_s);
- Vertical position of the source (x_z);
- Number of time steps (N_t);
- Time step size (Δt);

[1] *acwv-solver* source code: https://github.com/antoniomedeiros1/acwv-solver

- Number of result frames to save (defaults to 40);

Figure 5 gives an overview of the *acwv-solver* flow. First, the `acwv-solver` binary is called through `mpirun`, specifying the number of processes, as well as the other parameters listed before. Then, the input *.vti* file is loaded and used to create the velocity field vector and the spatial domain parameters, those being the domain extent (X and Z) and spacing (Δx and Δz) defined in the `<ImageData>` section of the file.

Once the input data is successfully initialized, the time loop starts. For each step, the stencil is computed according to equation 3.24, and the boundary conditions are applied. If the value of k matches the frame rate, determined by N_t and the number of frames, the current wave field is saved to a *.vti* output file.

After the solver finishes computing the results for the specified time duration of the simulation, the time spent is displayed in standard output, and the program exits.

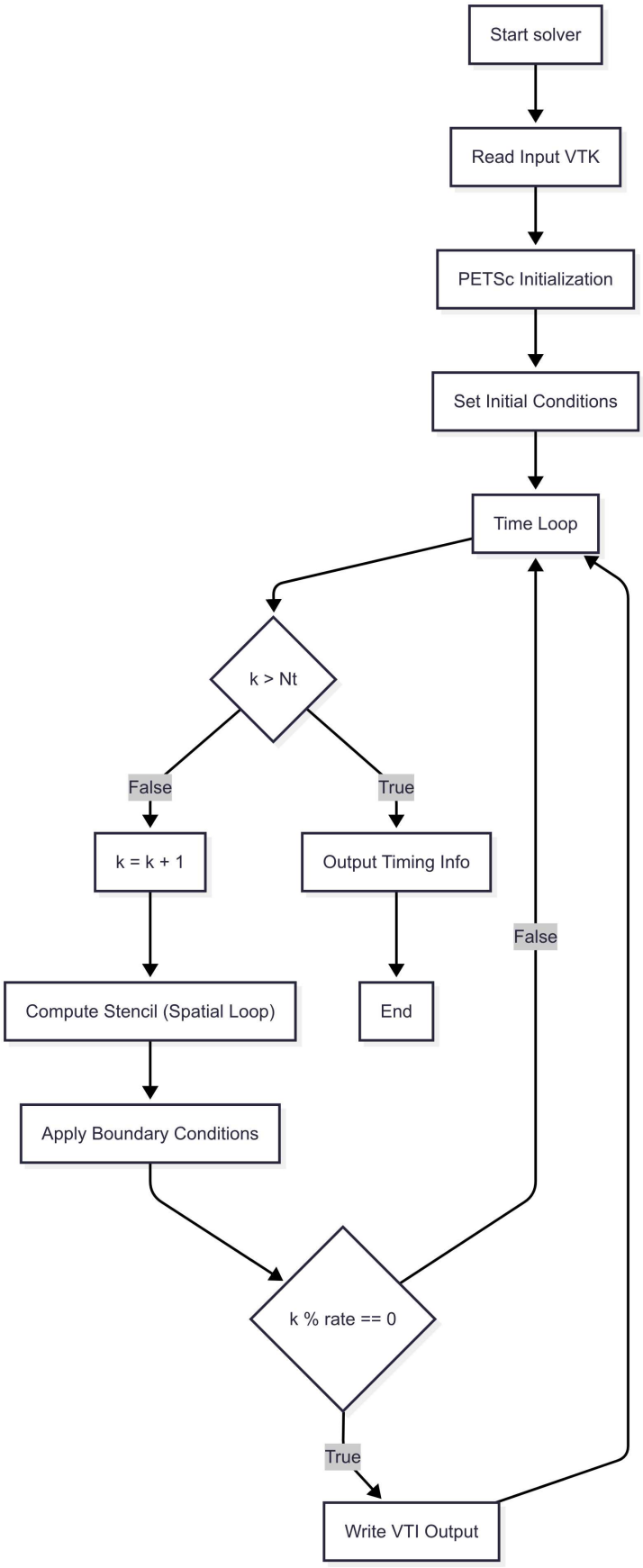


Figure 5 – Flowchart of the *acwv-solver* implementation.

4 NUMERICAL DISPERSION SUPRESSION

To further improve the accuracy of the results obtained with the *acwv-solver*, this work proposes a post-processing stage based on a convolutional neural network inspired by the approach of [20]. The method follows the *pix2pix* framework for image-to-image translation, adapted to scientific data.

In the context of acoustic wave modeling, the post-processing task can be understood as mapping a dispersion-affected wavefield, produced by a coarse-grid FDM simulation, to a high-fidelity reference wavefield computed on a finer grid. This is feasible because numerical dispersion in finite difference simulations follows systematic and deterministic patterns, depending on factors such as grid resolution, wave frequency, and propagation direction. A neural network trained on paired low/high-resolution data can learn this mapping and correct dispersion artifacts.

By leveraging this property, the proposed method enhances simulation accuracy without increasing grid density, thereby reducing computational costs while preserving the fidelity of the wavefield.

4.1 The pix2pix Algorithm

Convolutional Neural Networks (CNNs) have been extensively applied in image-to-image translation tasks [9]. While effective, traditional CNN-based approaches often require careful manual design of loss functions to ensure high-quality outputs. The pix2pix algorithm [9] addresses this limitation by employing Generative Adversarial Networks (GANs) for image-to-image translation, enabling the model to automatically learn a loss function that maps raw inputs to target outputs. Moreover, pix2pix makes use of a specific variant of GANs known as *conditional* GANs (cGANs).

4.1.1 Conditional Generative Adversarial Networks (cGANs)

A Generative Adversarial Network [22] is a deep learning framework in which a generator network (G) produces synthetic outputs from a random noise vector z, formally expressed as $G : z \rightarrow y$. The GAN consists of two components: the generator (G), which aims to produce outputs indistinguishable from real data, and the discriminator (D), which is trained to differentiate between real and generated data — hence the term “adversarial.”

GANs follow an encoding-decoding architecture, consisting of downsampling and upsampling layers. The generator first downsamples the input image through convolutional layers with strided convolutions (reducing spatial dimensions while increasing feature depth), followed by a bottleneck layer processing high-level features, then upsamples via transposed convolutions (recovering lost resolution), ultimately producing a realistic output image. Meanwhile, the discriminator, instead of evaluating the whole image, processes it

in small patches using a series of strided convolutions (downsampling progressively), and final convolutional layers that output a matrix of "real/fake" probabilities, allowing it to focus on local texture realism rather than global structure.

In Conditional GANs, the generator receives both an observed input x and a noise vector z, learning a mapping $G : \{x, z\} \rightarrow y$. This conditioning enforces a one-to-one correspondence between input and output data, allowing the model to penalize inconsistencies in the joint configuration of output pixels and thereby achieve more accurate and coherent results.

The loss function of the pix2pix algorithm, as defined in [9], is given by:

$$\mathcal{L}_{\text{pix2pix}}(G, D) = \text{argminmax}\{\mathcal{L}_{\text{cGAN}}(G, D) + \lambda_{L_1}\mathcal{L}_{L_1}(G)\} \tag{4.1}$$

where

$$\mathcal{L}_{\text{cGAN}}(G, D) = E_{y_r, y_t}[log(D(y_r, y_t))] + E_{y_r}[log(1 - D(y_r, G(y_r)))], \tag{4.2}$$

$$\mathcal{L}_{L_1}(G) = E_{y_r, y_t}[||y_t - G(y_r)||_1], \tag{4.3}$$

G is the generator, D is the discriminator, λ_{L_1} is the certainty coefficient of the L_1-norm loss term, E is the mean operator, y_r is the raw data, and y_t is the target data.

4.1.2 Edge Detection with Sobel Operator

The Sobel operator is a widely used method in image processing for edge detection (see Figure 6). As demonstrated by [20], the pix2pix network can be enhanced by incorporating a Sobel-based loss term into the training process.

Figure 6 – Example where the Sobel operator was used for edge detection of an image [1].

The Sobel operator computes the gradient magnitude of the image intensity:

$$S = \sqrt{(\boldsymbol{S_x} * \boldsymbol{A})^2 + (\boldsymbol{S_z} * \boldsymbol{A})^2}, \tag{4.4}$$

where S is the gradient magnitude, $\boldsymbol{A}$ is the input data, $*$ is the convolution operator, and $\boldsymbol{S_x}$ and $\boldsymbol{S_z}$ are the horizontal and vertical convolution kernels, respectively:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad S_z = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}. \tag{4.5}$$

To integrate the Sobel operator in the pix2pix network, the Sobel loss term $\mathcal{L}_S$ is introduced:

$$\mathcal{L}_{\text{pix2pix}}(G, D) = \text{argminmax}\{\mathcal{L}_{\text{cGAN}}(G, D) + \lambda_{L_1}\mathcal{L}_{L_1}(G) + \lambda_S\mathcal{L}_S(G)\}, \tag{4.6}$$

where

$$\mathcal{L}_S(G) = E_{y_r, y_t}[||S(y_t) - S(G(y_r))||_1], \tag{4.7}$$

and λ_S is the certainty coefficient of the Sobel operator loss term.

4.2 Implementation of the Numerical Dispersion Suppression Neural Network

The network was implemented following the official *pix2pix* tutorial, adapting the input format to VTK ImageData, and is available at the Github repository [1]. Scalar fields were reshaped into 2D tensors, normalized, and resized to 256×256 pixels. The dataset was split into training and testing subsets. An overview of the cGAN training process is shown in Figure 7.

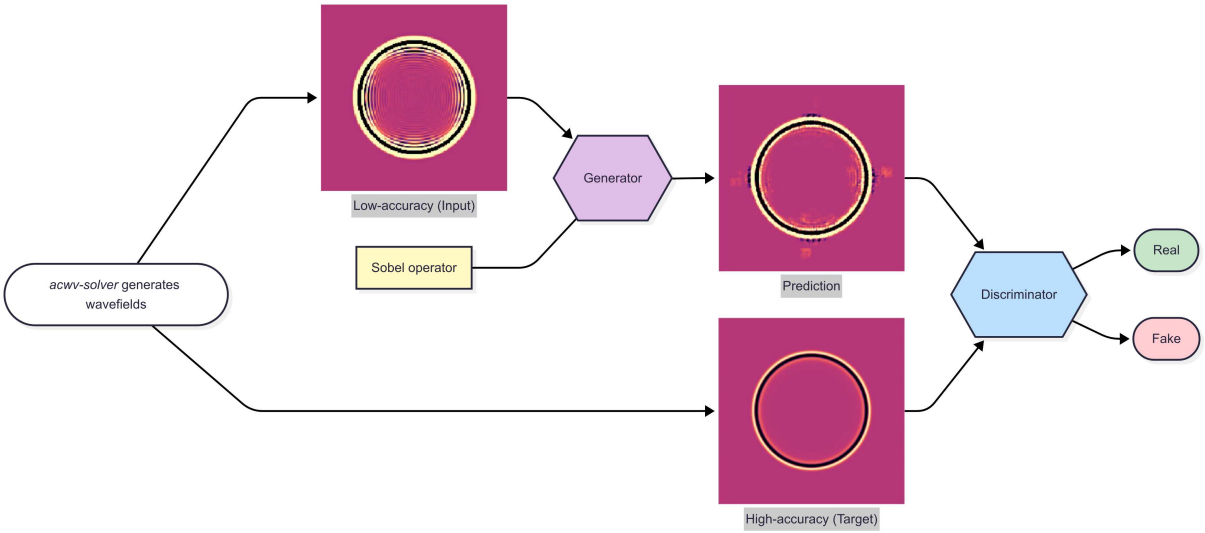


Figure 7 – Overview of the cGAN training process for numerical dispersion supression.

The generator used was a U-Net architecture [16], composed of 8 downsampling layers and 7 upsampling layers, all using 2x2 kernels. The downsampling path increases

[1] https://github.com/antoniomedeiros1/acwv-postprocessing

the number of filters from 64 up to 512, with batch normalization applied to all but the first layer, and LeakyReLU (preserving gradient flow for negative values) activations throughout. The upsampling path mirrors this structure, using Conv2DTranspose layers, batch normalization, and ReLU activations, with dropout applied to the first three upsampling layers. Skip connections are used between corresponding downsampling and upsampling layers, and the final output layer uses a tanh activation to produce a single-channel output.

The discriminator follows the PatchGAN [9] approach, concatenating the input and target images along the channel dimension and passing them through five downsampling layers with increasing filter sizes (64, 128, 256), followed by zero padding and a Conv2D layer with 512 filters. Batch normalization and LeakyReLU activations are used, and the final output is a patch-level map indicating real or fake regions.

The Sobel operator is applied to both the generated and target images to extract edge information, and an additional L1 loss is computed on these gradient images. The total generator loss is a weighted sum of the standard GAN loss, the L1 loss between the generated and target images, and the Sobel-based L1 loss, with tunable hyperparameters for each component. This encourages the generator to produce outputs that not only match the target globally but also preserve sharp features and edges, which are crucial in scientific data. For the training, the Adam optimizer was used for both networks.

5 RESULTS

The Finite Difference Method (FDM) solver was employed to compute the numerical solutions of the wave equation. To evaluate the effectiveness of the proposed numerical dispersion suppression strategy, two spatial discretizations were considered. First, a coarse grid with spatial spacing $h = 15$ was used to generate datasets containing numerical dispersion, as detailed in Chapter 3. Second, a refined grid with spacing $h = 5$ was adopted to produce the reference (target) dataset for training the cGAN. The dataset was divided into 437 (60%) samples for training and 146 (20%) samples for testing, from a total of 700 generated samples.

The evaluation procedure began with a comparative analysis of several parameter configurations for training the network, using the constant velocity model ($c = 1500$ m/s) as a benchmark. Once the optimal training parameters were identified, the trained network was applied to each of the velocity models. In all simulation scenarios, the total propagation time was set to $T = 2,\mathrm{s}$, with a time step $\Delta t = 0.00025,\mathrm{s}$ ($N_t = 8000$), yielding a total of 700 snapshots—corresponding to approximately one snapshot every 11 time iterations.

To assess prediction quality, the Normalized Root Mean Square Error (NRMSE) metric was employed to compare both the raw and the predicted data against the reference dataset. The NRMSE is defined as:

$$\mathrm{NRMSE}(y, y_{\mathrm{true}}) = \frac{\mathrm{RMSE}(y, y_{\mathrm{true}})}{\max(y_{\mathrm{true}}) - \min(y_{\mathrm{true}})}, \tag{5.1}$$

where the Root Mean Square Error (RMSE) is given by:

$$\mathrm{RMSE}(y, y_{\mathrm{true}}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_{\mathrm{true},i})^2}. \tag{5.2}$$

The simulations and neural network training were performed on workstation equipped with an *AMD Ryzen 7 5700X 8-Core Processor* CPU (8 cores, 16 threads), 32 GB of RAM. The software environment consisted of *Ubuntu 22.04 LTS*, Python 3.10, *Tensorflow 2.19.0*, and *NumPy* and *Matplotlib* libraries for numerical processing and visualization. All experiments were executed in double-precision floating-point format to ensure numerical stability.

5.1 Velocity Models

For the tests, artificially generated velocity fields were considered to evaluate the computational implementation and numerical dispersion suppression in a heterogeneous medium. These can be visualized in Figure 8, and their mathematical functions are given by:

- Parallel planes

$$c(i,j) = \begin{cases} 1500 & \text{if } 0 \le i < \frac{N_x}{3} \\ 2000 & \text{if } \frac{N_x}{3} \le i < \frac{2N_x}{3} \\ 3000 & \text{if } \frac{2N_x}{3} \le i < N_x \end{cases} \tag{5.3}$$

- Semicircle

$$c(i,j) = \begin{cases} 1500 & \text{if } j < \frac{N_z}{3} \\ 2000 & \text{if } \frac{N_z}{3} \le j < \sqrt{\left(\frac{N_x}{2}\right)^2 - (i - \frac{N_x}{2})^2} + N_z \\ 3000 & \text{otherwise} \end{cases} \tag{5.4}$$

- Bent layers

$$c(i,j) = \begin{cases} 2500 & \text{if } j < \sqrt{\left(\frac{N_z}{2}\right)^2 - i^2} + \frac{N_z}{2} \quad (\text{for } i < \frac{N_z}{2}) \\ 2500 & \text{if } j < -\sqrt{\left(\frac{N_z}{2}\right)^2 - (i - N_x)^2} + \frac{N_z}{2} \quad (\text{for } i \ge \frac{N_z}{2}) \\ 3500 & \text{otherwise} \end{cases} \tag{5.5}$$

- Reservoir

$$c(i,j) = \begin{cases} 1500 & \text{if } j \le \frac{8m_e^3}{i^2+4m_e^2} \\ 2000 & \text{if } \frac{8m_e^3}{i^2+4m_e^2} < j < m_e \\ 2500 & \text{if } m_e \le j < \min\left(\frac{8m_e^3}{i^2+4m_e^2} + 0.125N_z, 0.3675N_z\right) \\ 3000 & \text{if } 0.3675N_z \le j < \frac{8m_e^3}{i^2+4m_e^2} + 0.125N_z \\ 3500 & \text{if } \frac{8m_e^3}{i^2+4m_e^2} + 0.125N_z \le j < 0.7N_z \\ 4000 & \text{otherwise} \end{cases} \tag{5.6}$$

  where $m_e = 0.175N_z$.

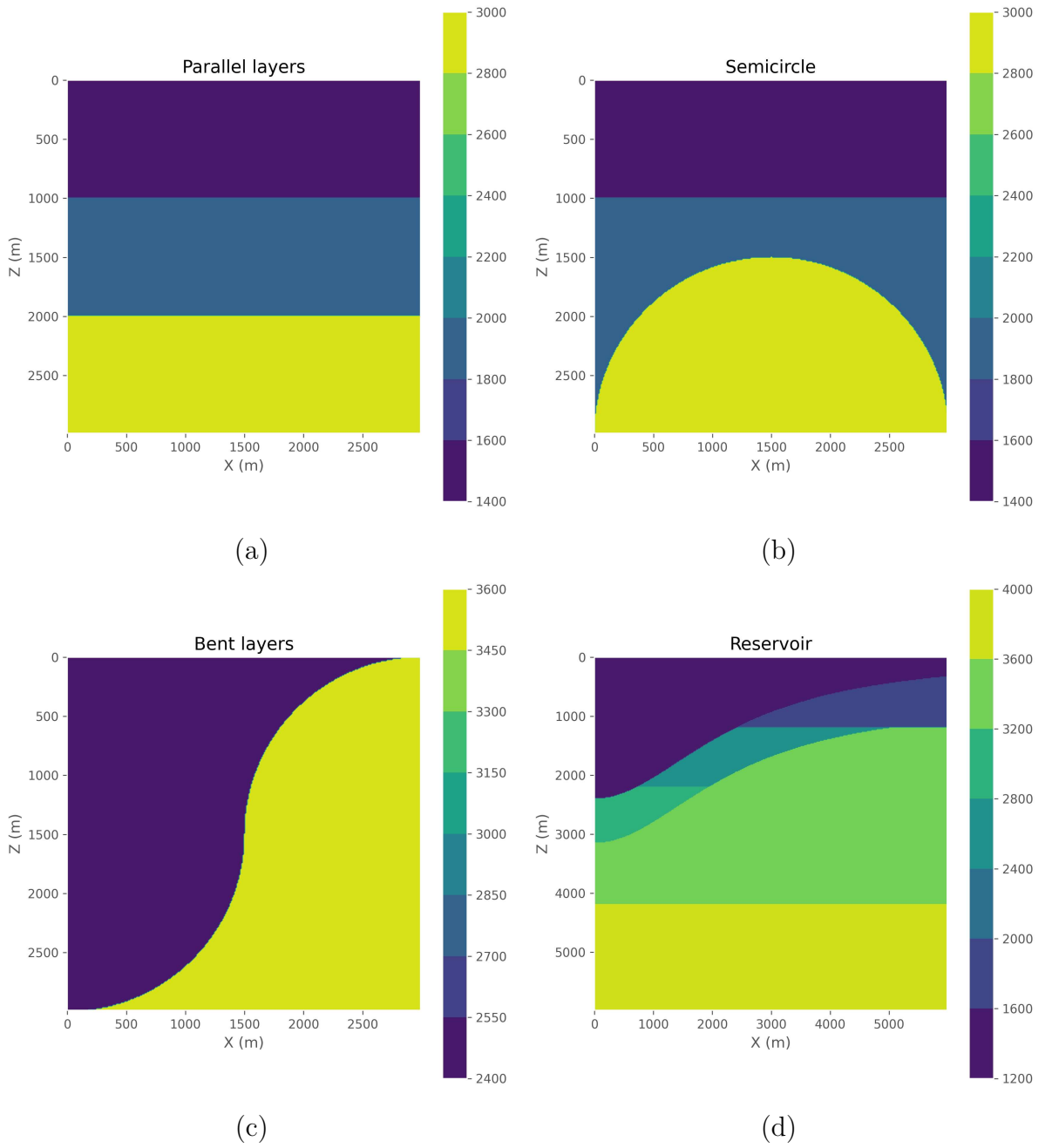


Figure 8 – Visualization of the artificial velocity fields used for the tests. (a) Has dimensions $3000m \times 3000m$ with velocities ranging from $1500m/s$ and $3000m/s$. (b) Has the same dimensions as and velocity variations as (a), with a semi-circle shape at the bottom. (c) Also has the same dimensions as (a) with velocities $2500m/s$ and $3500m/s$. (d) Has dimensions of $6000m \times 6000m$ and is supposed to simulate a reservoir region, with velocities ranging from $1500m/s$ to $4000m/s$.

5.2 Parameters Comparison

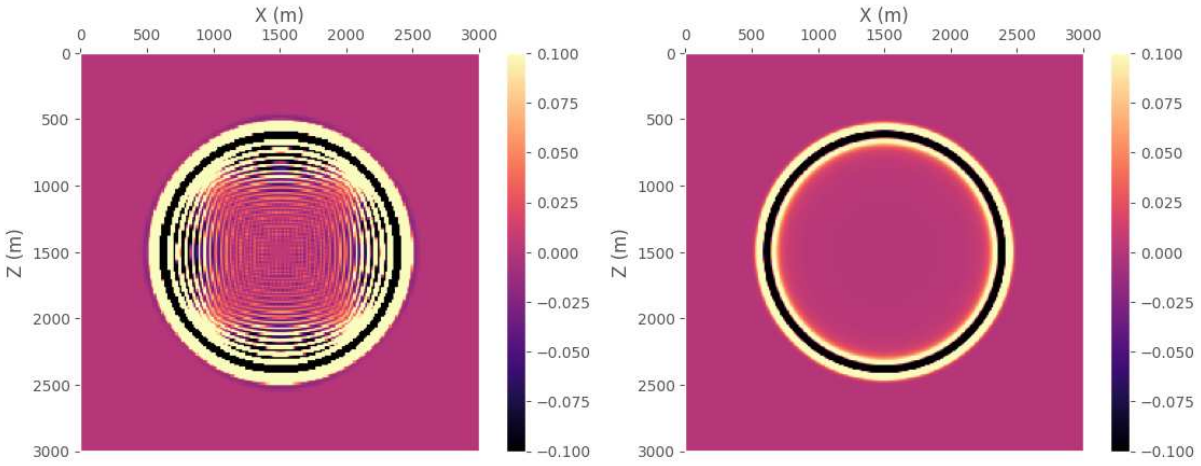
To identify the optimal set of parameters for training the numerical dispersion suppression model, several test cases were conducted. These tests considered different values for the learning rate, number of training epochs, and the certainty coefficients λ_{L_1}

and λ_S. The parameters used in each test case are summarized in Table 1.

Test Case	Learning rate	Epochs	λ_{L_1}	λ_S
1	0.001	400	30	0 (no Sobel)
2	0.001	400	30	10
3	0.001	600	30	10
4	0.001	600	30	15

Table 1 – Parameters adopted in each test case.

The computational domain employed for these experiments featured a constant velocity field of 1500 m/s, with dimensions of 3000 m × 3000 m. Figure 9 presents a snapshot of the results for the coarse grid (raw data) and for the refined grid (target data). The source was positioned at the domain center, located at $x_s = 1500$ m and $z_s = 1500$ m. The NRMSE for the coarse grid dataset ($h = 15$) relative to the refined grid ($h = 5$) was 85.84%.



(a) (b)

Figure 9 – Snapshots at $t = 0.6875s$ of the raw and target data: (a) $h = 15$, (b) $h = 5$. The NRMSE for calculated for the raw data was of 85.84%

The results for each test case are presented in Figure 10. For Test Case 1 (no Sobel operator), the NRMSE was 3.35%. For Test Case 2, which follows the parameters from [20], the NRMSE decreased to 2.76%. Test Case 3, with an increased number of epochs, achieved the lowest error of 1.26%. Finally, Test Case 4, with an increased λ_S, yielded an NRMSE of 1.72%.

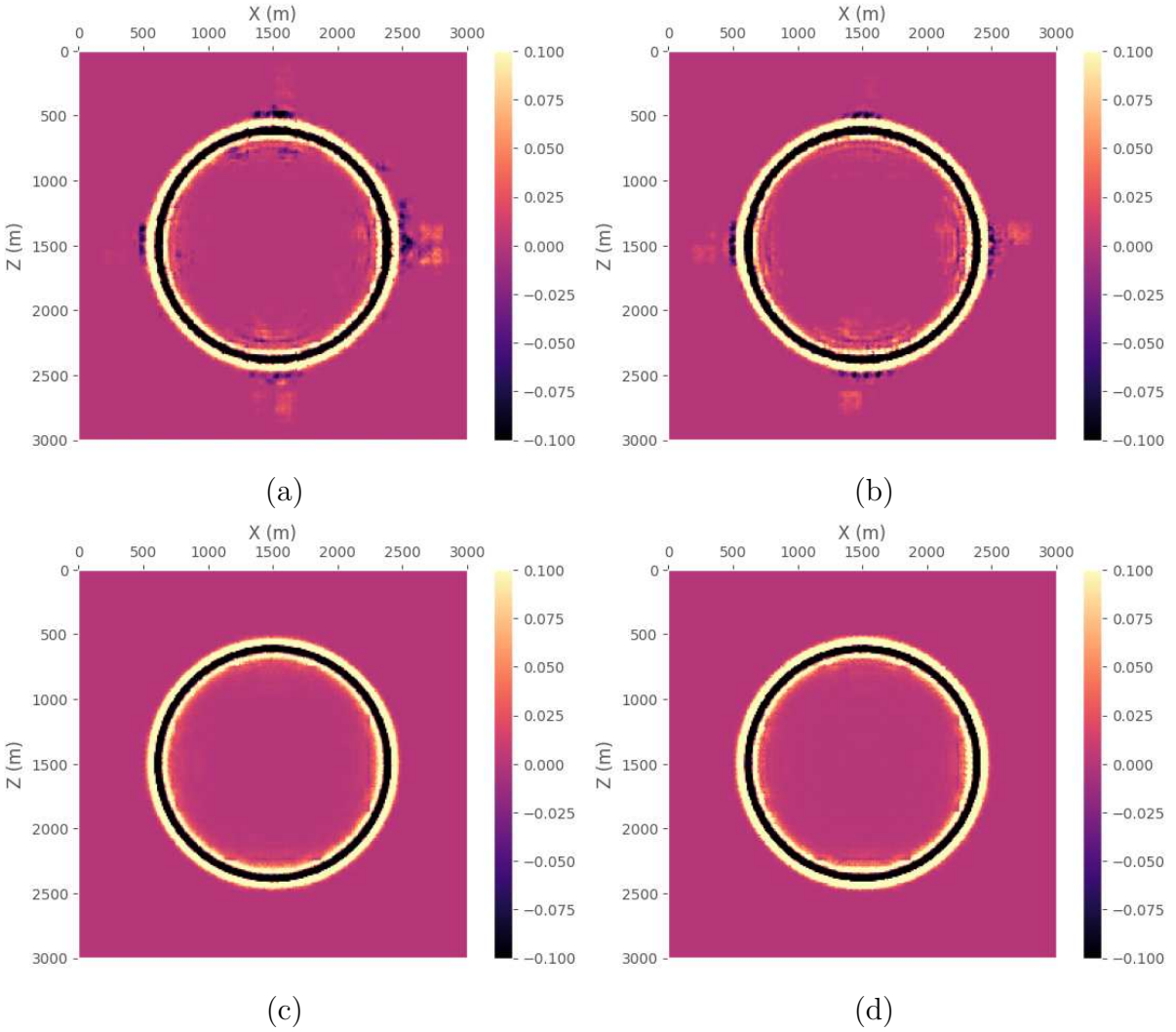
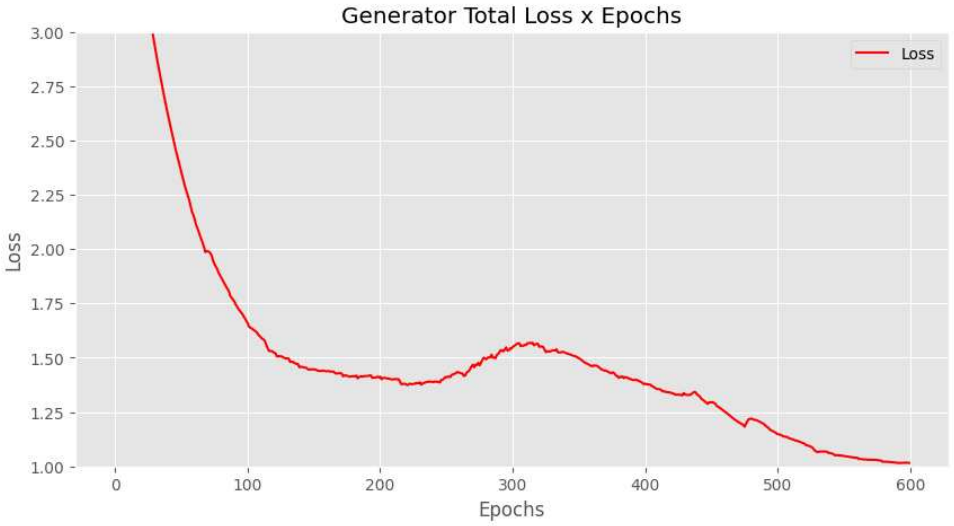


Figure 10 – Snapshots at $t = 0.6875s$ for each test case. (a) Corresponds to the first test case, where the Sobel operator is not applied, and presents an NRMSE of 3.35%. (b) The second test case, with the same parameters of [20], presents an NRMSE of 2.76%. (c) In the third test case, the number of epochs was increased, resulting in an NRMSE of 1.26%. Lastly, the Sobel loss term certainty coefficient was increased, and the NRMSE found was 1.72%.

The results indicate that the best performance for the cGAN implementation was achieved with a learning rate of 0.001, 600 training epochs, and certainty coefficients $\lambda_{L_1} = 30$ and $\lambda_S = 10$. The findings also highlight the importance of including the Sobel operator, which accelerates convergence, as well as the influence of the number of training epochs. Although the pix2pix algorithm alone provides mild numerical dispersion suppression, it remains insufficient. Figure 11 shows the evolution of the generator loss function for the best-performing case.



(a)

Figure 11 – Generator loss function.

.

5.3 Velocity Models Results

The FDM solver was used to generate the results for each of the velocity models described in chapter 5.1, evaluating the effectiveness of both the solver and the cGAN in each case. For all of the velocity field simulations, a learning rate of 0.001, with 600 epochs and certainty coefficients $\lambda_{L_1} = 30$ and $\lambda_S = 10$ was used, except for the reservoir model, where only 400 epochs were necessary, due to it having more points than the other models.

5.3.1 Parallel Layers

Figures 12 and 13 present the computed wavefields and the corresponding numerical dispersion suppression results for the parallel layers velocity model at timestamps $t = 0.55\,\text{s}$ and $t = 0.825\,\text{s}$, respectively. The source was positioned at the center of the model, with $x_s = 1500\,\text{m}$ and $z_s = 1500\,\text{m}$. For the case at $t = 0.55\,\text{s}$, the NRMSE of the coarse-grid data ($h = 15\,\text{m}$) relative to the reference solution ($h = 5\,\text{m}$) was 52.54%, whereas the prediction from the cGAN achieved a significantly lower NRMSE of 1.20%.

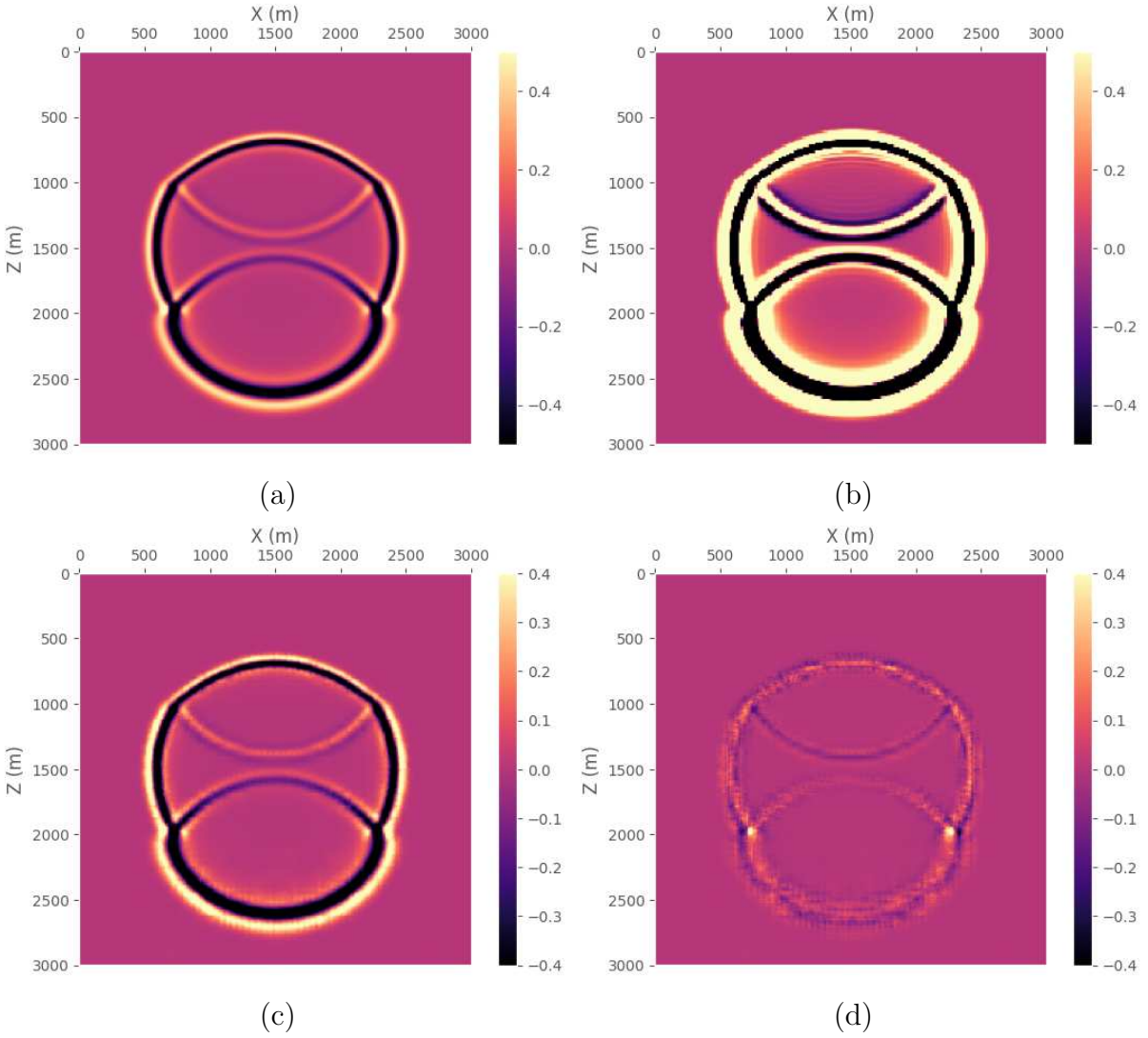


Figure 12 – Wave field for the parallel layers velocity model at $t = 0.55s$. (a) is the target data with $h = 5m$. (b) is the raw input data with $h = 15m$. (c) is the prediction from the cGAN. (d) is the difference between (c) and (a). NRMSE for the raw data was 52.54% while for the predicted wave field was 1.20%.

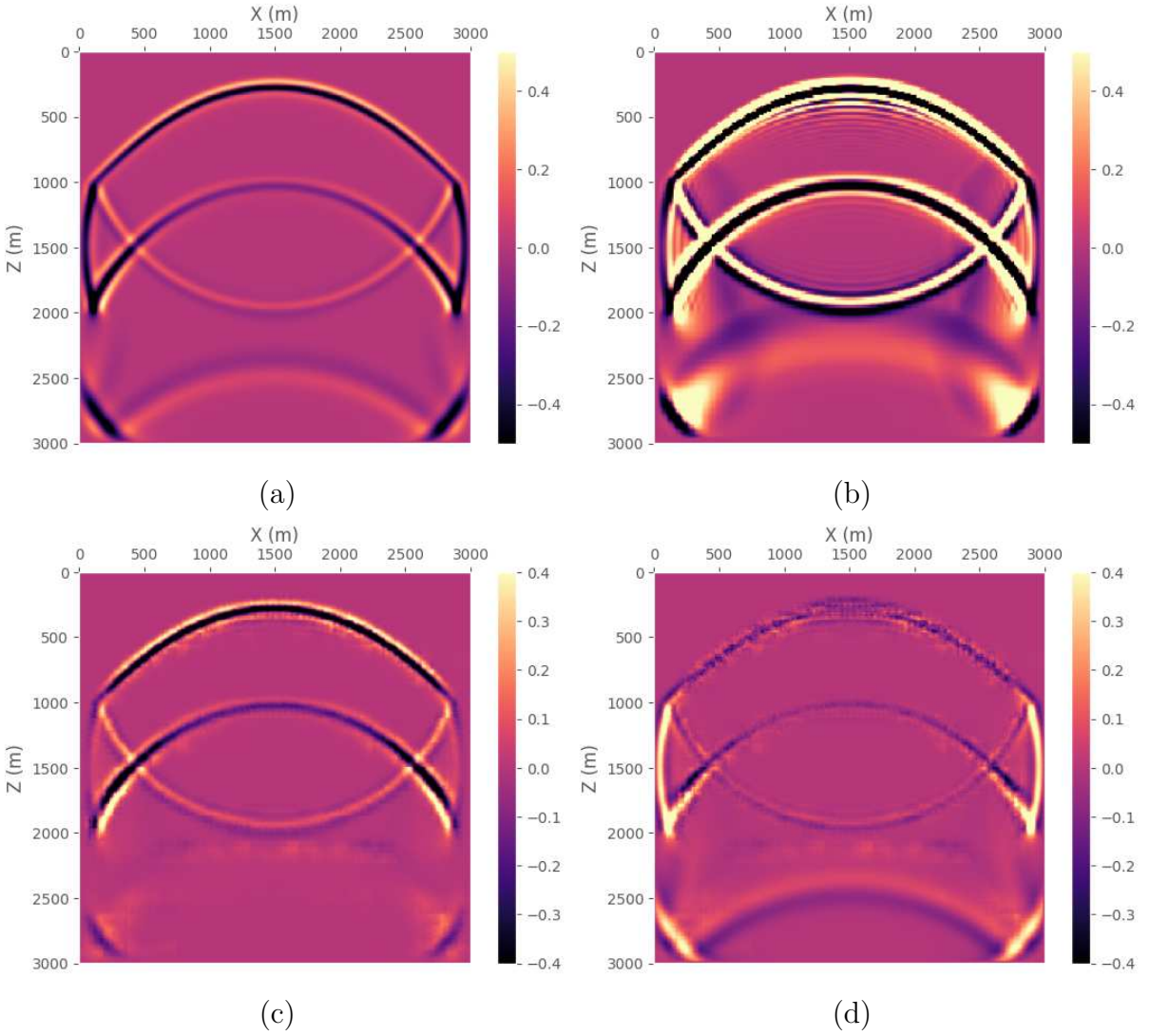


Figure 13 – Wave field for the parallel layers velocity model at $t = 0.825s$. (a) is the target data with $h = 5m$. (b) is the raw input data with $h = 15m$. (c) is the prediction from the cGAN. (d) is the difference between (c) and (a). NRMSE for the raw data was 35.84% while for the predicted wave field was 4.12%.

5.3.2 Semicircle

Figures 14 and 15 present the computed wavefields and the corresponding numerical dispersion suppression results for the semicircle velocity model at timestamps $t = 0.275\,\mathrm{s}$ and $t = 0.4125\,\mathrm{s}$, respectively. The source was positioned at the center of the model, with $x_s = 1500\,\mathrm{m}$ and $z_s = 1500\,\mathrm{m}$. At $t = 0.275\,\mathrm{s}$, the NRMSE of the coarse-grid input ($h = 15\,\mathrm{m}$) relative to the refined-grid reference ($h = 5\,\mathrm{m}$) was 43.23%. The cGAN prediction reduced this error to 1.58%, demonstrating substantial suppression of numerical dispersion.

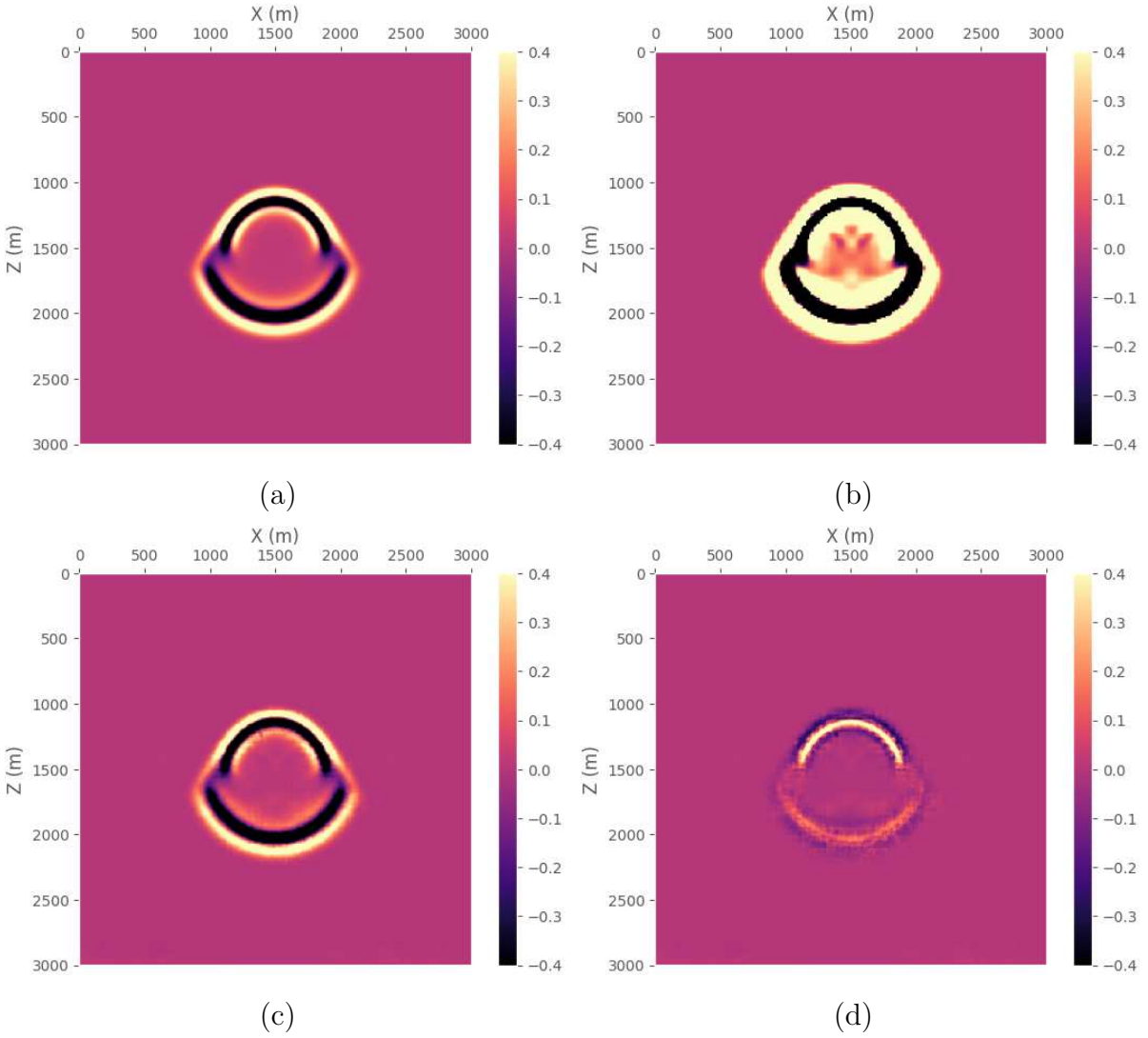


Figure 14 – Wave field for the semicircle velocity model at $t = 0.275s$. (a) is the target data with $h = 5m$. (b) is the raw input data with $h = 15m$. (c) is the prediction from the cGAN. (d) is the difference between (c) and (a). NRMSE for the raw data was 43.23% while for the predicted wave field was 1.58%.

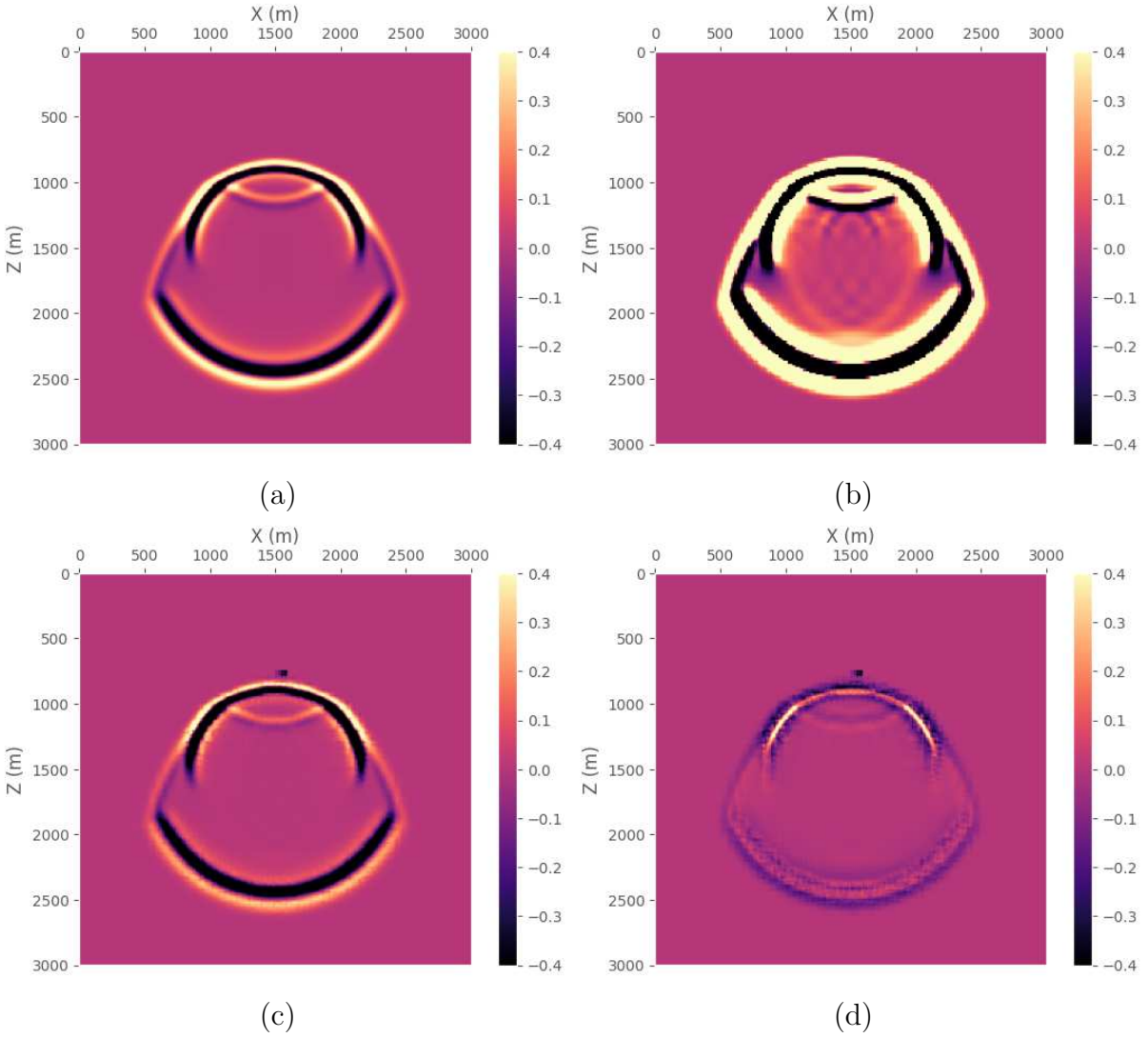


Figure 15 – Wave field for the semicircle velocity model at $t = 0.4125s$. (a) is the target data with $h = 5m$. (b) is the raw input data with $h = 15m$. (c) is the prediction from the cGAN. (d) is the difference between (c) and (a). NRMSE for the raw data was 50.50% while for the predicted wave field was 1.82%.

5.3.3 Bent layers

Figures 16 and 17 present the computed wavefields and the corresponding numerical dispersion suppression results for the bent layers velocity model at timestamps $t = 0.275\,\mathrm{s}$ and $t = 0.55\,\mathrm{s}$, respectively. The source was positioned at the center of the model, with $x_s = 1500\,\mathrm{m}$ and $z_s = 1500\,\mathrm{m}$.

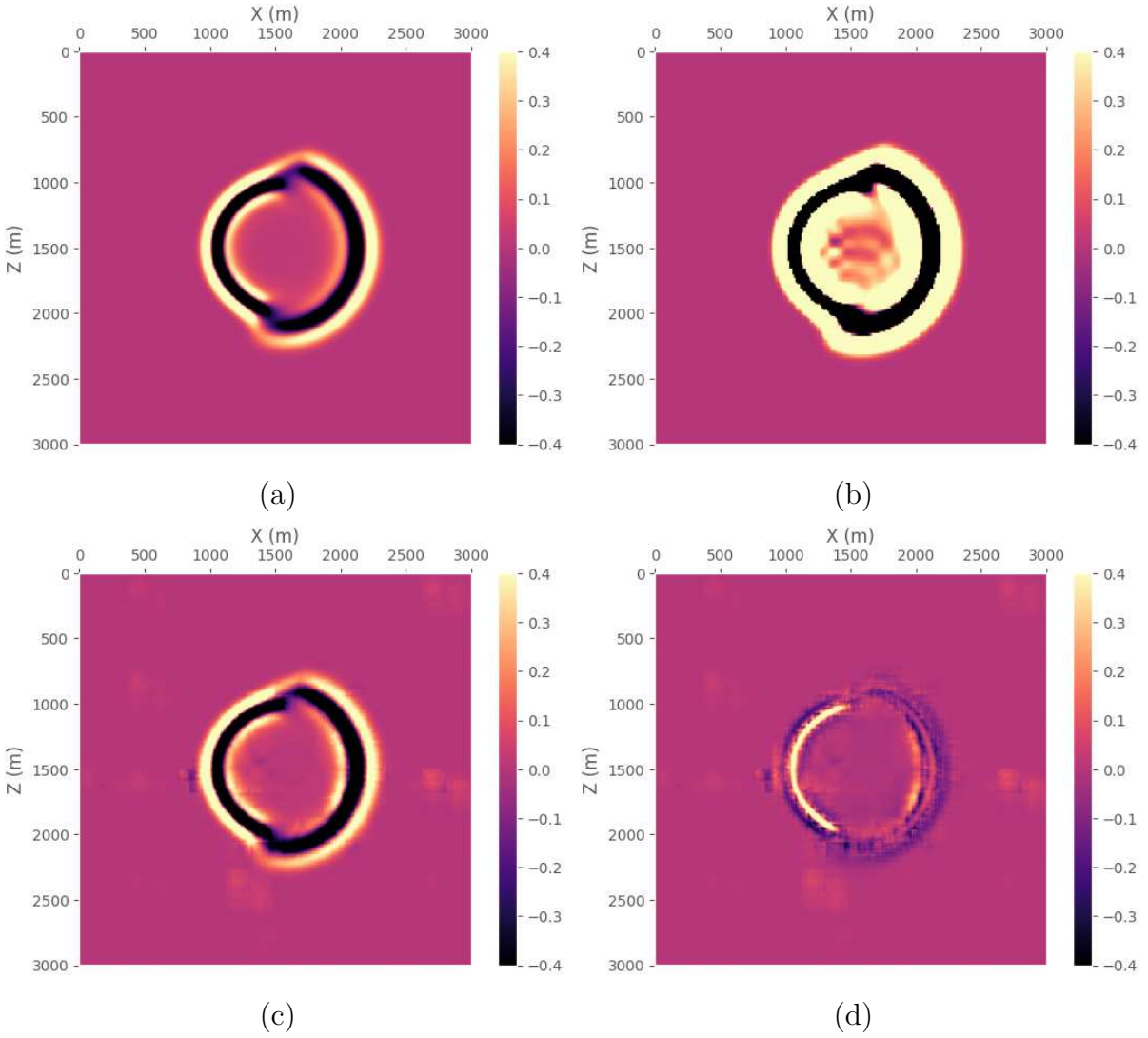


Figure 16 – Wave field for the bent layers velocity model at $t = 0.275s$. (a) is the target data with $h = 5m$. (b) is the raw input data with $h = 15m$. (c) is the prediction from the cGAN. (d) is the difference between (c) and (a). NRMSE for the raw data was 53.27% while for the predicted wave field was 1.78%.

At $t = 0.275\,\mathrm{s}$, the coarse-grid input ($h = 15\,\mathrm{m}$) exhibited an NRMSE of 53.27% relative to the refined-grid reference ($h = 5\,\mathrm{m}$). The cGAN prediction reduced this error to 1.78%. At $t = 0.55\,\mathrm{s}$, the NRMSE of the coarse-grid data was 58.24%, while the cGAN-predicted wavefield achieved an NRMSE of 2.28%.

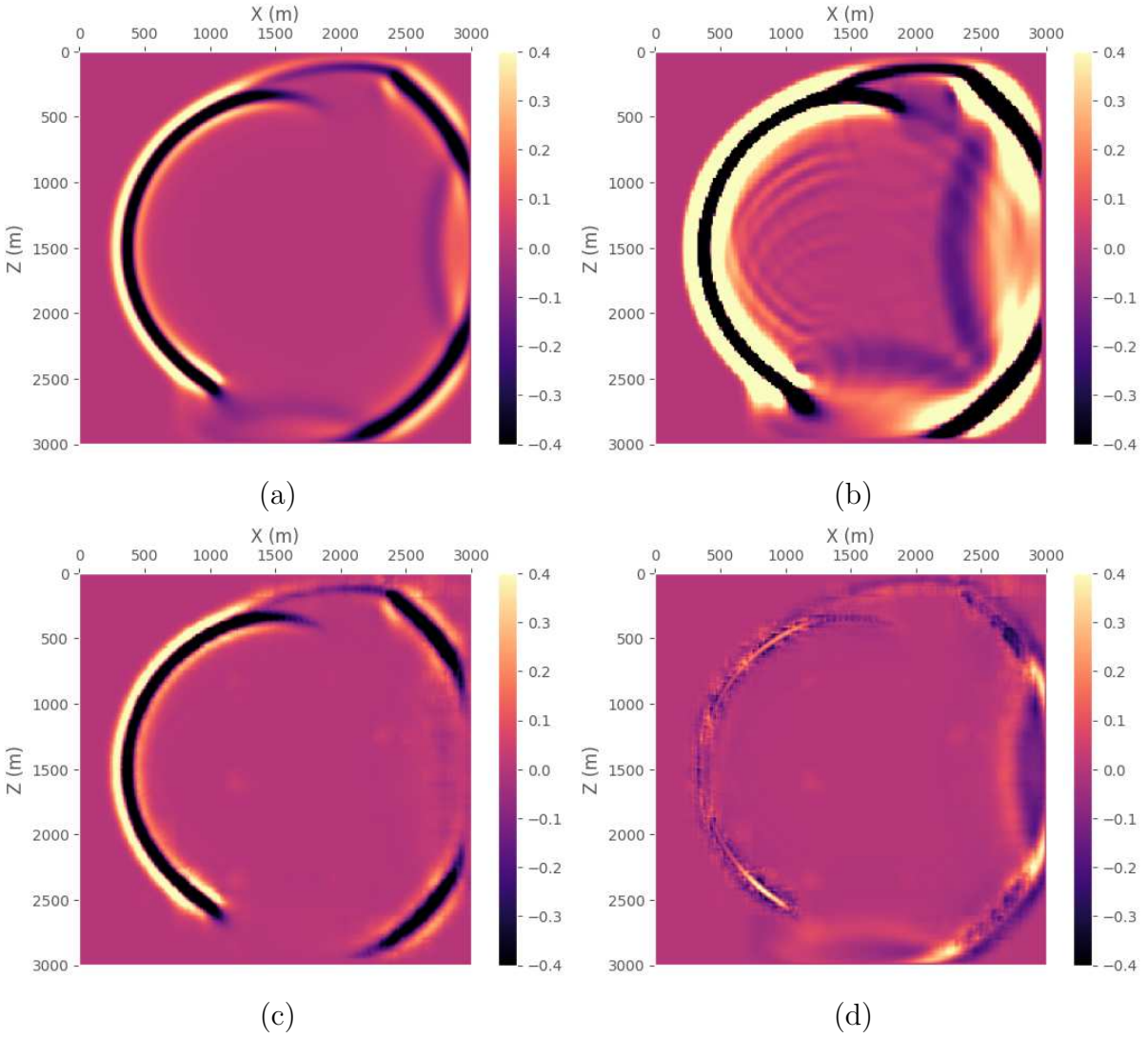


Figure 17 – Wave field for the bent layers velocity model at $t = 0.55s$. (a) is the target data with $h = 5m$. (b) is the raw input data with $h = 15m$. (c) is the prediction from the cGAN. (d) is the difference between (c) and (a). NRMSE for the raw data was 58.24% while for the predicted wave field was 2.28%.

5.3.4 Reservoir

Figures 18 and 19 present the computed wavefields and the corresponding numerical dispersion suppression results for the reservoir velocity model at timestamps $t = 1.375\,\mathrm{s}$ and $t = 1.7875\,\mathrm{s}$, respectively. The source was positioned at the top of the model, with $x_s = 3000\,\mathrm{m}$ and $z_s = 300\,\mathrm{m}$.

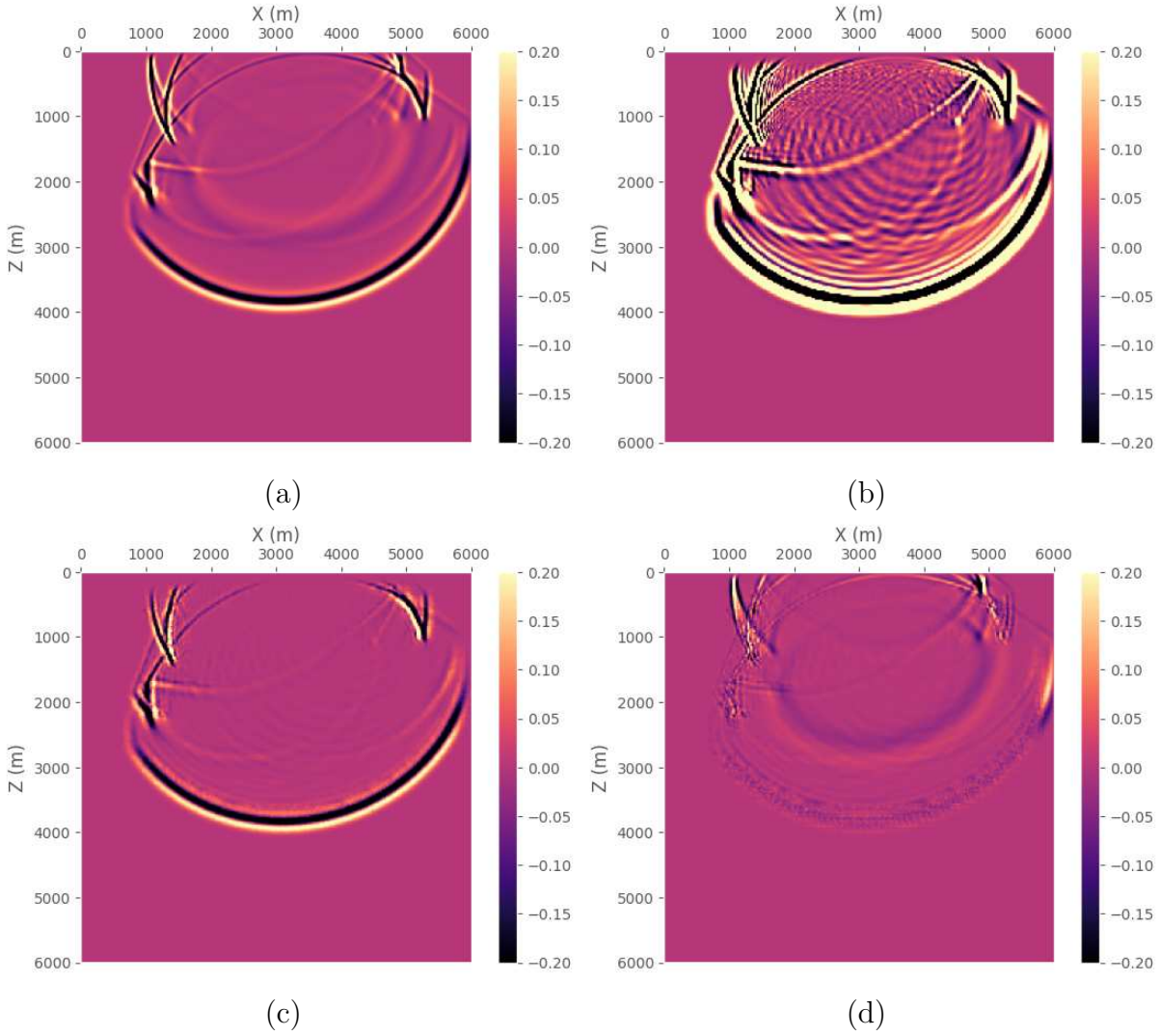


Figure 18 – Wave field for the reservoir velocity model at $t = 1.375s$. (a) is the target data with $h = 5m$. (b) is the raw input data with $h = 15m$. (c) is the prediction from the cGAN. (d) is the difference between (c) and (a). NRMSE for the raw data was 25.89% while for the predicted wave field was 1.62%.

At $t = 1.375\,\mathrm{s}$, the NRMSE of the coarse-grid input ($h = 15\,\mathrm{m}$) relative to the refined-grid reference ($h = 5\,\mathrm{m}$) was 25.89%, while the cGAN prediction achieved an NRMSE of 1.62%. At $t = 1.7875\,\mathrm{s}$, the NRMSE values were 23.35% for the coarse-grid data and 1.69% for the prediction.

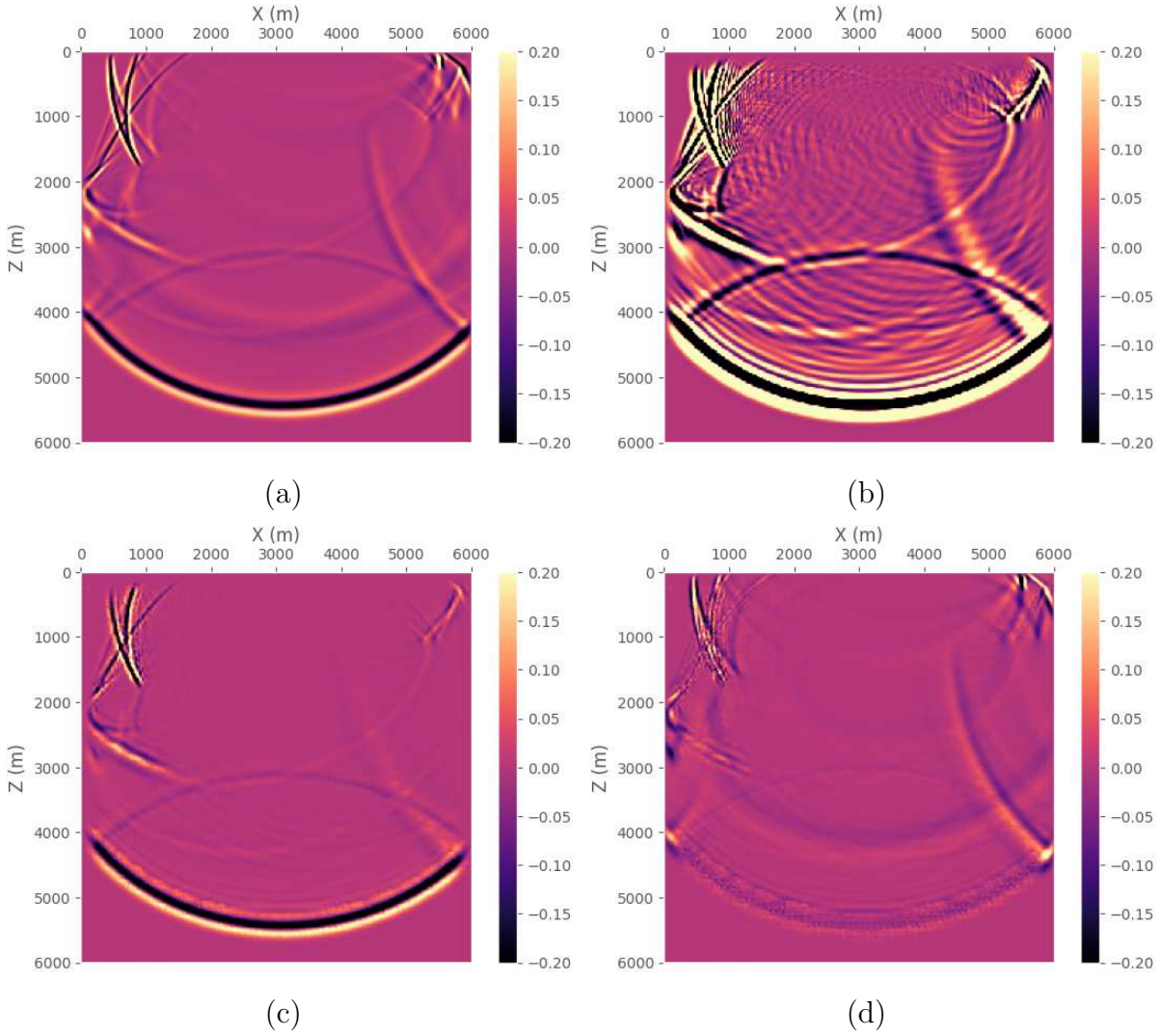


Figure 19 – Wave field for the reservoir velocity model at $t = 1.7875s$. (a) is the target data with $h = 5m$. (b) is the raw input data with $h = 15m$. (c) is the prediction from the cGAN. (d) is the difference between (c) and (a). NRMSE for the raw data was 23.35% while for the predicted wave field was 1.69%.

Due to its larger dimensions compared to the other velocity models, the reservoir model required only 400 training epochs, with a total training time of 117.7 s. The computation of the wavefields took 128.7 s for the coarse grid and 741.9 s for the refined grid. Considering that 182 s were necessary to suppress the numerical dispersion in all of the 700 samples generated for the coarse grid, the total time was reduced by 42.3% when comparing the refined grid with the coarse grid combined with the numerical dispersion suppression network.

A histogram of NRMSE for all 700 samples is shown in Figure 20. The mean NRMSE for the raw coarse-grid input was 19.60%, whereas the mean NRMSE for the cGAN prediction was 1.86%, indicating a substantial improvement in accuracy after applying the model.

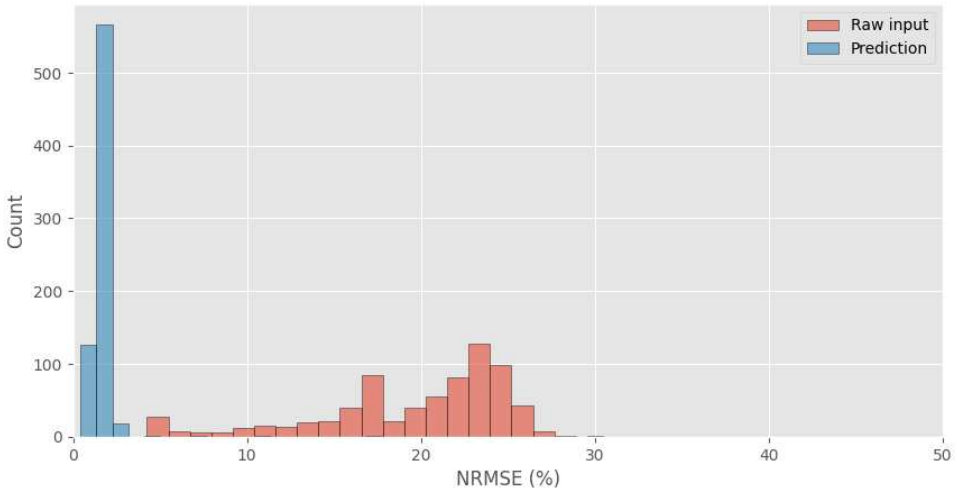


Figure 20 – NRMSE percentage histogram for the reservoir velocity model.

5.4 Overall Analysis

Table 2 summarizes the NRMSE before and after cGAN correction for all velocity models and time instants evaluated.

Model	**Time (s)**	**Raw NRMSE (%)**	**cGAN NRMSE (%)**
Parallel Layers	0.55	52.54	1.20
Parallel Layers	0.825	35.84	4.12
Semicircle	0.275	43.23	1.58
Semicircle	0.4125	50.50	1.82
Bent Layers	0.275	53.27	1.78
Bent Layers	0.55	58.24	2.28
Reservoir	1.375	25.89	1.62
Reservoir	1.7875	23.35	1.69

Table 2 – NRMSE before and after dispersion suppression.

The results show a consistent and significant reduction in NRMSE across all tested models and times. In some cases, such as the Parallel Layers model at $t = 0.55\,\mathrm{s}$, the error dropped by more than 50 percentage points. The inclusion of the Sobel operator was key for improved edge preservation and convergence stability. These findings confirm the effectiveness of the proposed cGAN-based approach for mitigating numerical dispersion in FDM simulations, even in complex heterogeneous velocity fields.

6 CONCLUSION AND FUTURE WORKS

The Finite Difference Method (FDM) solver developed for the acoustic wave equation demonstrated strong agreement with reference results from the literature, confirming the validity of the numerical formulation and implementation. The adoption of Reynolds and damping boundary conditions was effective in minimizing spurious reflections at domain boundaries, which are known to degrade solution accuracy in wave propagation problems. This, combined with the integration of the PETSc library, enabled scalable parallel execution through MPI, significantly improving computational performance. The choice of the VTK `ImageData` format for output not only ensured compatibility with widely used visualization tools, such as ParaView, but also allowed embedding of essential simulation metadata—including grid spacing, domain size, and temporal resolution—thereby enhancing the reproducibility and interoperability of the results.

In addition to the solver implementation, an improved Pix2Pix conditional Generative Adversarial Network (cGAN) was successfully applied to suppress numerical dispersion in coarse-grid simulations. Across multiple velocity models, the proposed approach reduced the Normalized Root Mean Square Error (NRMSE) from values exceeding 50% in the raw coarse-grid data to below 3% in the cGAN predictions, with the best cases achieving errors near 1%. Although a refined-grid solution is still required for training, the approach achieved competitive performance using only 437 samples for the training set, which is significantly smaller than typical datasets in deep learning-based wavefield reconstruction.

The integration of the *acwv-solver* with the cGAN presents a promising pathway for reducing computational costs in large-scale simulations. In scenarios involving velocity models with large spatial extents, the method enables the initial computation of a limited number of refined-grid time steps for network training, after which the remaining simulation can be performed on a coarser grid without significant loss of accuracy due to numerical dispersion. This hybrid approach balances computational efficiency with predictive accuracy, offering a practical solution for scenarios where high-resolution modeling would otherwise be prohibitively expensive.

Looking ahead, the next stage of development involves extending the *acwv-solver* to three-dimensional domains. This enhancement will enable the study of wave propagation in more realistic and complex geological models, further increasing the solver's applicability. Moreover, it will open the possibility of adapting the numerical dispersion suppression network to fully three-dimensional spatial domains—a direction for which no related studies were identified in the current literature. In addition, future work should include a detailed performance evaluation of the solver in parallel computing environments, assessing scalability, load balancing, and communication overhead for different problem sizes and computing architectures. Such tests will provide deeper insights into the computational

efficiency of the proposed method and guide optimizations for high-performance computing platforms.

These advancements could position this methodology at the forefront of high-performance numerical modeling combined with data-driven dispersion correction, potentially impacting a broad range of applications in computational geophysics and beyond.

REFERENCES

1 Ashish. Understanding edge detection (sobel operator). `https://medium.datadriveninvestor.com/understanding-edge-detection-sobel-operator-2aada303b900`, 2018. Acesso em agosto de 2025.

2 Balay, S., Abhyankar, S., Adams, M. F., Benson, S., Brown, J., Brune, P., Buschelman, K., Constantinescu, E. M., Dalcin, L., Dener, A., Eijkhout, V., Faibussowitsch, J., Gropp, W. D., Hapla, V., Isaac, T., Jolivet, P., Karpeev, D., Kaushik, D., Knepley, M. G., Kong, F., Kruger, S., May, D. A., McInnes, L. C., Mills, R. T., Mitchell, L., Munson, T., Roman, J. E., Rupp, K., Sanan, P., Sarich, J., Smith, B. F., Zampini, S., Zhang, H., Zhang, H., and Zhang, J. PETSc Web page. `https://petsc.org/`, 2025.

3 Cerjan, C., Kosloff, D., Kosloff, R., and Reshef, M. A nonreflecting boundary condition for discrete acoustic and elastic wave equations. *Geophysics 50*, 4 (1985), 705–708.

4 De Souza Silva, B., Silva, J., and Landau, L. Optimized finite differences scheme applied to the acoustic wave equation.

5 Gadylshin, D., Ivanov, P., and Petrova, T. Machine learning-based numerical dispersion mitigation in finite difference modeling. In *High Performance Computing*. Springer, 2021, pp. 35–49.

6 Golubev, V., Shevchenko, A., and Petrov, I. Simulation of seismic wave propagation in a multicomponent oil deposit model. *International Journal of Applied Mechanics 12*, 08 (2020), 2050084.

7 Han, B., Li, X., and Chen, Y. Eliminate time dispersion of seismic wavefield simulation by semi-supervised learning with cnn and gru. *Energies 15*, 20 (2022), 7701.

8 Huang, G., and Symes, W. W. Analytic and numerical solutions to the seismic wave equation in continuous media. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 476*, 2240 (November 2020), 20200636.

9 Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks.

10 Ji, H., Wang, J., and Li, X. Wave propagation modeling using a machine learning-based finite difference scheme. *Journal of Computational Physics 510* (2025), 113133.

11 Kaur, H., Smith, J., and Wang, Y. Post-propagation filters with deep learning for numerical dispersion suppression. In *SEG Technical Program Expanded Abstracts 2019* (2019), pp. 3207–3211.

12 Koene, E. F. M. On eliminating time dispersion from synthetic seismograms. *Geophysical Journal International 213*, 1 (2018), 169–181.

13 Malkoti, A., Vedanti, N., and Tiwari, R. K. A highly efficient implicit finite difference scheme for acoustic wave propagation. *Journal of Applied Geophysics 161* (2019), 204–215.

14 Reynolds, A. C. Boundary conditions for the numerical solution of wave propagation problems. *Geophysics 43*, 6 (1978), 1099–1110.

15 Ricker, N. Wavelet functions and their polynomials. *Geophysics 18*, 1 (1953), 10–40.

16 Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation, 2015.

17 Schroeder, W., Martin, K., and Lorensen, B. *The Visualization Toolkit (4th ed.)*. Kitware, 2006.

18 Vanga, M., Barman, D., and Ojha, M. An optimized finite-difference method to minimize numerical dispersion of acoustic wave propagation using a genetic algorithm. *Geophysics 87*, 3 (04 2022), T265–T279.

19 Xu, Y., Li, X., Chen, Y., and Zhang, X. Removing time dispersion from elastic wave modeling with the pix2pix algorithm based on cgan. *Remote Sensing 15*, 12 (2023), 3120.

20 Yan, H.-Y. Seismic modeling by combining the finite-difference scheme with the numerical dispersion suppression neural network. *Petroleum Science 21*, 5 (2024), 3157–3165.

21 Yang, D., Song, G., Hua, B., and Calandra, H. Simulation of acoustic wavefields in heterogeneous media: A robust method for automatic suppression of numerical dispersion. *Geophysics 75*, 3 (06 2010), T99–T110.

22 Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. Dive into deep learning, 2023.

23 Zhou, Y., Yang, D., Ma, X., and Li, J. An effective method to suppress numerical dispersion in 2d acoustic and elastic modelling using a high-order padé approximation. *Journal of Geophysics and Engineering 12*, 1 (01 2015), 114–129.