Scientific Research Publishing

Advances in Pure Mathematics, 2026, 16(3), 226-248
https://www.scirp.org/journal/apm
ISSN Online: 2160-0384
ISSN Print: 2160-0368

Preconditioned Neural Network Solvers for the Frequency-Domain Wave Equation with Nearly Analytic Discretization

Shili Pang, Chao Lang*

School of Applied Science, Beijing Information Science and Technology University, Beijing, China
Email: *langchao@bistu.edu.cn

How to cite this paper: Pang, S.L. and Lang, C. (2026) Preconditioned Neural Network Solvers for the Frequency-Domain Wave Equation with Nearly Analytic Discretization. *Advances in Pure Mathematics*, **16**, 226-248.
https://doi.org/10.4236/apm.2026.163011

Received: February 9, 2026
Accepted: March 20, 2026
Published: March 23, 2026

Abstract

The nearly analytic discretization of the frequency-domain wave equation produces large-scale, sparse, and ill-conditioned linear system, which challenge conventional iterative solvers. To mitigate this problem, we employ deep neural networks to construct an approximate preconditioner, which is then embedded within BICGSTAB algorithm. This integration retains the convergence characteristics of classical methods while effectively accelerating the iterative process. To be specific, the proposed neural network-based preconditioned solver achieves faster convergence and improved stability compared with classical preconditioned solvers. Numerical tests on representative models show that the method reduces iterative numbers without sacrificing solution accuracy, suggesting its effectiveness for scalable and high-fidelity simulations of frequency-domain wave propagation.

Keywords

Deep Neural Networks, Preconditioned Iterative Methods, BICGSTAB, Frequency-Domain Wave Equation, Nearly Analytic Discretization

1. Introduction

The frequency-domain wave equation provides a fundamental mathematical model for describing wave propagation in heterogeneous media [1], whose numerical solution relies on accurate discretization of differential operators. Common discretization approaches include the finite difference method [2], finite element method [3], pseudo-spectral method [4], and spectral element method [5]. Owing to its simplicity, computational efficiency, and favorable parallel scalability, the finite difference method is widely adopted in practice [6]. However, stand-

ard finite difference schemes on coarse grids often suffer from significant numerical dispersion, which degrades solution accuracy, whereas the use of fine grids substantially increases computational cost [7]. To overcome this limitation, Yang and Lang proposed the nearly analytic discrete (NAD) scheme, a finite-difference formulation that incorporates both displacement and gradient information to approximate higher-order derivatives. By enhancing numerical accuracy and compactness while preserving the simplicity of classical finite difference methods, the NAD scheme effectively suppresses numerical dispersion on coarse grids and improves computational efficiency, particularly in the frequency-domain [8].

Independently of the specific discretization scheme, the frequency-domain wave equation gives rise to a large-scale sparse linear system, whose coefficient matrix (commonly referred to as the impedance matrix [9]) possesses unfavorable mathematical properties [10]. In particular, the impedance matrix is typically non-Hermitian, indefinite, and nearly singular [7], which poses substantial difficult for numerical solution and severely degrades the efficiency of wavefield simulation [11]. To address such linear system, Krylov subspace iterative method is widely adopted due to its favorable computational complexity and modest memory requirement [12]. For nonsymmetric and indefinite problem, BICGSTAB [13] and GMRES [14] are among the most commonly used algorithms. In practice, BICGSTAB is often preferred for large-scale simulation because it combines relatively fast convergence with low memory consumption, in the spirit of conjugate gradient methods [15]. Nevertheless, the severe ill-conditioning of the impedance matrix can significantly impair the convergence behavior of standard BICGSTAB iteration [16].

To alleviate the convergence degradation caused by the ill-conditioning of the impedance matrix, preconditioning is commonly employed to improve numerical stability and accelerate iterative solvers [17]. In the context of BICGSTAB, an appropriate preconditioner can effectively reduce the condition number of the system matrix, thereby enhancing convergence behavior. Classical preconditioners, such as incomplete LU (ILU) factorizations and complex-shift preconditioners (M_0), can provide moderate performance gains in certain cases [18]. However, the strong indefiniteness and near-singularity inherent in the impedance matric often limit the robustness and effectiveness of the traditional approach [16]. As a result, the development of more efficient and adaptive preconditioning strategies remains an active and important research direction.

Recent advances in deep neural networks (DNNs) have opened new possibilities for addressing challenging numerical problems arising from the solution of partial differential equations. In particular, DNNs have been explored as tools for learning operator-related structures and accelerating iterative solvers. For instance, Ali *et al.* [19] developed multigrid graph neural networks to learn optimized parameters within domain decomposition frameworks, while Xiang *et al.* [20] combined graph neural networks with radial basis function finite difference schemes to enhance approximation accuracy. Huang *et al.* [21] proposed learning

effective smoothers directly from operator stencils via DNNs, trained on small-scale problems and successfully generalized to larger systems of the same PDE class. These studies suggest that DNNs are capable of capturing nontrivial operator features and approximating inverse mappings associated with complex linear systems. Owing to their nonlinear representation capacity, DNN-based approaches offer a promising avenue for constructing adaptive preconditioners that alleviate ill-conditioning and improve convergence behavior [16].

In this work, we develop a deep neural network-based preconditioning strategy and incorporate it into the BICGSTAB solver for large-scale sparse linear systems arising from NAD discretization of the frequency-domain wave equation. The proposed approach couples the high-order accuracy of the NAD scheme with a data-driven preconditioner, enabling an adaptive mitigation of the adverse spectral properties of the impedance matrix. As a result, this method improves the convergence behavior of BICGSTAB compared with conventional preconditioning technique and provides an effective framework for the efficient solution of large-scale sparse linear system.

2. Preconditioned Iterative Methods for Frequency-Domain Wave Equations

2.1. NAD Scheme of the Frequency-Domain Wave Equation

Wave equations used in seismic modeling are typically classified into acoustic and elastic formulations. As a representative example, we consider the two-dimensional frequency-domain acoustic wave equation in a homogeneous medium.

$$\frac{\omega^2}{c^2}u(x,z)+\Delta u(x,z)=-\frac{1}{c^2}s,\quad (x,z)\in D, \tag{1}$$

where $\omega=2\pi f$ denotes the angular frequency with f being the frequency, $c=c(x,z)$ is the wave speed, $u(x,z)$ represents the wavefield, Δ denotes the Laplacian operator, s is the source term, and D denotes the computational domain.

The discretization of equation (1) is carried out with NAD scheme, where the partial derivatives along the x and z directions are computing respectively.

$$\begin{cases}\dfrac{\omega^2}{c^2}u+\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial z^2}=-\dfrac{1}{c^2}s\\ \dfrac{\omega^2}{c^2}\dfrac{\partial u}{\partial x}+\dfrac{\partial^3 u}{\partial x^3}+\dfrac{\partial^3 u}{\partial z^2\partial x}=-\dfrac{1}{c^2}\dfrac{\partial s}{\partial x}\\ \dfrac{\omega^2}{c^2}\dfrac{\partial u}{\partial z}+\dfrac{\partial^3 u}{\partial z^3}+\dfrac{\partial^3 u}{\partial x^2\partial z}=-\dfrac{1}{c^2}\dfrac{\partial s}{\partial z}\end{cases} \tag{2}$$

Due to computational limitations, the domain D must be truncated with artificial boundary. Such boundary, however, generate spurious reflections that compromise the accuracy of both forward and inverse computation. To mitigate these reflection, we employ the widely adopted Perfectly Matched Layer (PML) absorbing boundary condition [5], resulting in a modified formulation of the wave

equation incorporating PML [22]:

$$\begin{cases} \left(\dfrac{\iota\omega}{\iota\omega+d_x}\right)^2\dfrac{\partial^2 u}{\partial x^2}+\left(\dfrac{\iota\omega}{\iota\omega+d_z}\right)^2\dfrac{\partial^2 u}{\partial z^2}-\left(\dfrac{(\iota\omega)^2 d_x'}{(\iota\omega+d_x)^3}\right)^2\dfrac{\partial u}{\partial x} \\ -\left(\dfrac{(\iota\omega)^2 d_z'}{(\iota\omega+d_z)^3}\right)^2\dfrac{\partial u}{\partial z}+\dfrac{\omega^2}{c^2}u=-\dfrac{1}{c^2}s \\ \left(\dfrac{\iota\omega}{\iota\omega+d_x}\right)^2\dfrac{\partial^3 u}{\partial x^3}+\left(\dfrac{\iota\omega}{\iota\omega+d_z}\right)^2\dfrac{\partial^3 u}{\partial x\partial z^2}-\dfrac{3(\iota\omega)^2 d_x'}{(\iota\omega+d_x)^3}\dfrac{\partial^2 u}{\partial x^2} \\ -\dfrac{(\iota\omega)^2 d_z'}{(\iota\omega+d_z)^3}\dfrac{\partial^2 u}{\partial x\partial z}+\left(\dfrac{3(\iota\omega)^2 d_x'^2}{(\iota\omega+d_x)^4}-\dfrac{(\iota\omega)^2 d_x''}{(\iota\omega+d_x)^3}+\dfrac{\omega^2}{c^2}\right)\dfrac{\partial u}{\partial x}=-\dfrac{1}{c^2}\dfrac{\partial s}{\partial x} \\ \left(\dfrac{\iota\omega}{\iota\omega+d_z}\right)^2\dfrac{\partial^3 u}{\partial z^3}+\left(\dfrac{\iota\omega}{\iota\omega+d_x}\right)^2\dfrac{\partial^3 u}{\partial x^2\partial z}-\dfrac{3(\iota\omega)^2 d_z'}{(\iota\omega+d_z)^3}\dfrac{\partial^2 u}{\partial z^2} \\ -\dfrac{(\iota\omega)^2 d_x'}{(\iota\omega+d_x)^3}\dfrac{\partial^2 u}{\partial x\partial z}+\left(\dfrac{3(\iota\omega)^2 d_z'^2}{(\iota\omega+d_z)^4}-\dfrac{(\iota\omega)^2 d_z''}{(\iota\omega+d_z)^3}+\dfrac{\omega^2}{c^2}\right)\dfrac{\partial u}{\partial z}=-\dfrac{1}{c^2}\dfrac{\partial s}{\partial z} \end{cases} \tag{3}$$

Equation (3) is discretized using a fourth-order NAD finite-difference scheme, with detailed derivations provided in **Appendix A1**. Subsequently, the computational nodes are ordered sequentially in a row-wise manner, from left to right within each row, and the wavefield values together with their corresponding gradient terms at each node are stored contiguously. Under this ordering strategy, the discretized system can be written in the following linear system:

$$\mathrm{C}u=s, \tag{4}$$

where C denotes the impedance matrix, which is complex-valued owing to the incorporation of PML, while u and s represent the discrete wavefield vector and the discrete source vector, respectively. The sparse structure of C is [22].

$$\begin{bmatrix}
B_5^{(1)} & B_6^{(1)} & B_7^{(1)} & \overbrace{\cdots}^{n_x-3} & B_8^{(1)} & \overbrace{\cdots}^{n_x-3} & B_9^{(1)} & & & & & & \\
B_4^{(2)} & B_5^{(2)} & B_6^{(2)} & B_7^{(2)} & \cdots & B_8^{(2)} & \cdots & B_9^{(2)} & & & & & \\
 & \ddots & & \ddots & & \ddots & & & & & & & \\
 & B_1^{(j)} & & B_2^{(j)} & \cdots & B_3^{(j)} & B_4^{(j)} & B_5^{(j)} & B_6^{(j)} & B_7^{(j)} & \cdots & B_8^{(j)} & \cdots & B_9^{(j)} \\
 & & & & & & \ddots & & \ddots & & \ddots & & & \\
 & & & & & & & B_1^{(N-1)} & \cdots & B_2^{(N-1)} & \cdots & B_3^{(N-1)} & B_4^{(N-1)} & B_5^{(N-1)} & B_6^{(N-1)} \\
 & & & & & & & & B_1^{(N)} & \underbrace{\cdots}_{n_x-3} & B_2^{(N)} & \underbrace{\cdots}_{n_x-3} & B_3^{(N)} & B_4^{(N)} & B_5^{(N)}
\end{bmatrix}$$

$B_{i,j}\left(i=1,\cdots,9;j=1,\cdots,N\right)$ represents 3×3 nonzero sub-block matrix, $N=\left(n_x-2\right)\cdot\left(n_z-2\right)$ denotes the total number of grid nodes (*i.e.* problem scale) and n_x, n_z is the number grid nodes along horizontal and vertical direction re-

spectively. The blank positions are all zero elements, and '$\overbrace{\cdots}^{n_x-3}$' denotes there are n_x-3 zero sub-block matrices. It is obvious that the impedance matrix C has nine nonzero diagonals and the number of its non-zero elements is $47(n_x-4)(n_z-4)+62(n_x+n_z-8)+8=O(47N)$. For the specific expressions of the elements in $B_{i,j}$, see **Appendix A2**.

2.2. Adverse Mathematical Properties of Linear System

To assess the numerical difficulty of the resulting linear system, we investigate the spectral properties of the impedance matrix under a representative discretization with $n_x=n_z=66$. As shown in **Figure 1**, the eigenvalues are clustered near the origin along the real axis and exhibit both positive and negative values, indicating that the impedance matrix is indefinite and nearly singular. Moreover, the introduction of the perfectly matched layer (PML) absorbing boundary condition renders the matrix asymmetric [22].

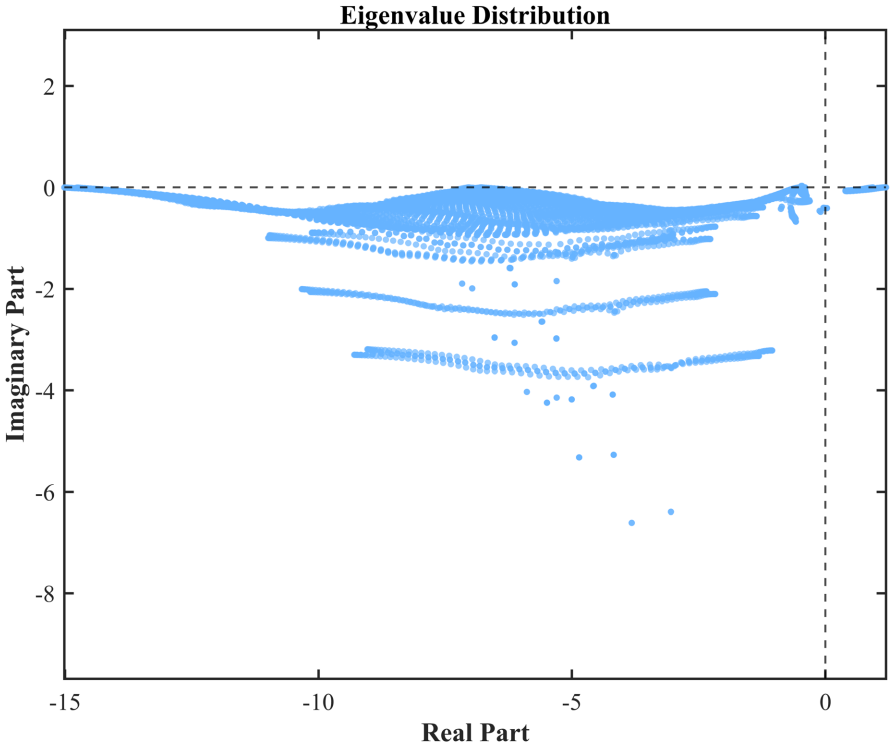

Figure 1. Typical eigenvalue distribution of the impedance matrix.

The coexistence of asymmetry, indefiniteness, and near-singularity poses substantial challenges for numerical solution. In particular, classical iterative methods such as the conjugate gradient (CG) algorithm require the coefficient matrix to be symmetric and positive definite [12] [23], and their direct application may result in poor convergence or even breakdown. Among Krylov subspace methods, BICGSTAB retains the low memory requirements characteristic of CG while enhancing robustness through biconjugate search directions and a local stabilization mechanism [13]. As a result, it is well suited for nonsymmetric and indefinite lin-

ear systems. Nevertheless, when the system is severely ill-conditioned, the convergence of standard BICGSTAB iterations can still be slow or numerically unstable [16].

2.3. Preconditioning Strategy for BICGSTAB Method

To further enhance the convergence behavior and numerical stability of the BICGSTAB algorithm, the introduction of preconditioning strategy is essential [12]. An appropriately designed preconditioner can alleviate the ill-conditioning of linear system while preserving solution accuracy, thereby reducing iterative numbers and improving overall efficiency. In particular, the original linear system (4) can be equivalently transformed into the following preconditioned form:

$$\begin{cases} \mathrm{C}\mathrm{P}^{-1}v = s \\ \mathrm{P}u = v \end{cases}, \tag{5}$$

where P denotes preconditioner [11], equation (5) has the same solutions as the original equation (4). **Algorithm 1** presents the preconditioned BICGSTAB implementation.

Algorithm 1. Preconditioned BICGSTAB method.

Require: Initial guess u_0 and tolerance $\varepsilon > 0$
Ensure: $\| \mathrm{C}u - s \|_2 / \| \mathrm{C}u_0 - s \|_2 < \varepsilon$
1: Set $p_1 = r_0$ and choose $\hat{r}_0$ such that $(r_0, \hat{r}_0) \neq 0$
2: **for** $k = 1, 2, \ldots$ **do**
3: $\rho_{k-1} = (r_{k-1}, \hat{r}_0)$
4: **if** $k > 1$ **then**
5: $\beta_{k-1} = (\alpha_{k-1} / \omega_{k-1}) \cdot (\rho_{k-1} / \rho_{k-2})$
6: $p_k = r_{k-1} + \beta_{k-1}(p_{k-1} - \omega_{k-1} v_{k-1})$
7: **end if**
8: $\hat{\boldsymbol{p}}_k = \boldsymbol{P}^{-1}\boldsymbol{p}_k$
9: $v_k = \mathrm{C}p_k$, $\alpha_k = \rho_{k-1} / (v_k, \hat{r}_0)$
10: $q_k = r_{k-1} - \alpha_k v_k$, $u_k = u_{k-1} + \alpha_k \hat{p}_k$
11: $\hat{\boldsymbol{q}}_k = \boldsymbol{P}^{-1}\boldsymbol{q}_k$
12: $w_k = \mathrm{C}\hat{q}_k$, $\omega_k = (q_k, w_k) / (w_k, w_k)$
13: $r_k = q_k - \omega_k w_k$, $u_{k+1} = u_k + \omega_k \hat{q}_k$
14: **if** $\| r_k \|_2 < \varepsilon \cdot \| r_0 \|_2$ **then**
15: **break**
16: **end if**
17: **enf for**

When P^{-1} provides a good approximation to C^{-1}, such that $\mathrm{C}\mathrm{P}^{-1}$ is close to the identity matrix, the condition number of the preconditioned system is significantly reduced, enabling convergence within a small number of iterations. Consequently, an effective preconditioner should satisfy $\mathrm{P}^{-1} \approx \mathrm{C}^{-1}$. At the same time, the construction of P must account for computational efficiency: the cost

of solving the auxiliary system $Pu = v$ should be kept minimal. These two requirements (accurate approximation of C^{-1} and efficient application of P^{-1}) are inherently competing. In practical implementations, an appropriate balance between approximation quality and computational cost is therefore required to achieve overall efficiency. A variety of classical preconditioning techniques, such as ILU factorization and the complex-shift M_0 preconditioner, have been proposed and successfully applied in many settings. However, for the impedance matrix characterized by strong indefiniteness and near-singularity, these traditional preconditioners often exhibit slow convergence, motivating the development of more effective and adaptive preconditioning strategy.

3. Deep Neural Networks-Based Preconditioner for Iterative Solvers

To alleviate the limitations of traditional preconditioning strategies for ill-conditioned linear systems, we employ deep neural networks to learn the inverse characteristics of the impedance matrix, leading to the construction of an efficient preconditioner that enhances the convergence rate of the BICGSTAB algorithm. From a mathematical perspective, the key lies in reformulating equation (5) into the equivalent form

$$\begin{cases} CNet(v,\theta) = s \\ Net(v,\theta) = u \end{cases}, \tag{6}$$

where v is an arbitrary complex-valued vector, and θ denote the network parameter. In the process of residual prediction, the networks need to predict the solution vector corresponding to each residual vector as accurately as possible. This operation requires that the network architecture satisfy the following key properties: 1) Consistency between input and output dimensions; 2) The capability to capture shared structural features across different vectors; and 3) Scalability to large-scale matrix–vector operations.

3.1. Network Architecture

To meet the requirements of approximating the inverse impedance operator, this study adopts a modified U-Net as the core network architecture [24]. The encoder-decoder structure not only ensures matching input and output dimensions but also provides a natural multi-resolution representation, which is particularly suitable for capturing both local high-frequency interactions and global low-frequency components inherent in the inverse operator [25]. Skip connections preserve fine-scale information across layers, playing a role analogous to residual correction mechanisms commonly employed in multigrid solvers [26]. The depth and scalability of this architecture further enhance its approximation capability, enabling it to accommodate problems with varying spatial scales and structural complexity. The specific modified U-Net used in this work (**Figure 2**) comprises approximately 28 layers, including 4 downsampling and 4 upsampling blocks.

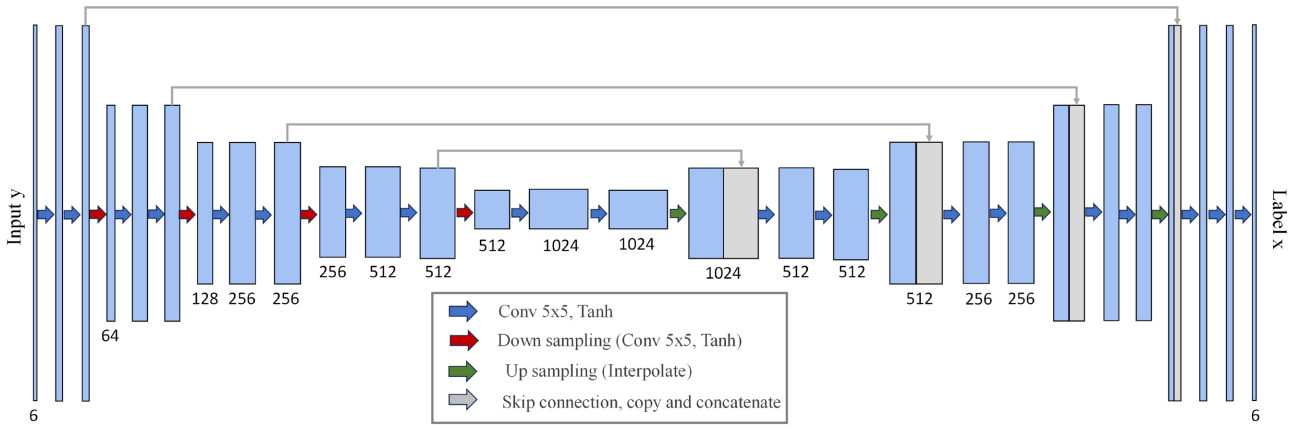

Figure 2. Network architecture diagram used for numerical experiments.

In the encoder, the feature extractor consists of repeated modules composed of convolutional layers with 5×5 kernel and Tanh activation function. The use of larger convolutional kernel effectively expands the receptive field, enabling the capture of broader contextual information [27]. The Tanh activation function introduces nonlinearity while allowing negative values to be activated, thereby enhancing the ability of networks to learn complicated features [28]. The ordered repetition of this module allows the feature extractor to progressively learn higher-level features. Additionally, the dimensionality adjustment module achieves feature compression by repeated convolutional operations, halving the spatial size and doubling the number of channels to enrich feature representation after each downsampling step [29].

In the decoder, a combination of nearest-neighbor interpolation and convolutional operation is employed to progressively reconstruct fine-grained spatial feature [30]. Nearest-neighbor interpolation is used to expand the spatial dimensions of the feature maps, while subsequent convolutional layers refine channel dimensions and extract features. To compensate for information loss during downsampling, skip connections are introduced after each upsampling stage to concatenate the corresponding encoder feature maps. This mechanism enhances the recovery of spatial details and ensures that networks can reconstruct complete and fine spatial feature distributions [31].

3.2. Dataset Construction

The NAD discretization expresses higher-order derivatives as linear combinations of the wavefield and its gradients. Accordingly, in equation (4), the solution vector consists of the wavefield and gradient components, which are defined independently and stored adjacently. To embed physical structure into the learning process, we adopt a multi-channel feature separation strategy. Specifically, the wavefield and its gradient components are assigned to separate channels, and the real and imaginary parts of each component are further processed independently to accommodate real-valued neural networks. As a result, the network input is represented as a six-channel real-valued tensor, which preserves physical infor-

mation while remaining compatible with real-valued network operations.

The overall dataset construction procedure is illustrated in Figure 3. To capture the dynamic behavior of the iterative solver, the residual vectors p_k is selected as the primary feature, as it reflect the evolving error structure during the iteration. To enhance sample diversity and mitigate overfitting to specific residual trajectories, additive noise vectors ξ_k are incorporated as a form of regularization. Each noise vector is generated with entries independently sampled from a uniform distribution, $\xi_k \sim \mathcal{U}(-\delta,\delta)$, where δ is a predefined noise level parameter. The signal-to-noise ratio (SNR) is uniformly sampled within the range of 10 dB to 20 dB and defined as $20\log_{10}\left(\frac{\max(|p_k|)}{|\xi_k|}\right)$. The noise magnitude is scaled accordingly to satisfy the sampled SNR value, and the enhanced vector $\boldsymbol{x}_k$ is constructed via element-wise addition: $\boldsymbol{x}_k = \boldsymbol{p}_k + \xi_k$. Subsequently, the matrix–vector product $\boldsymbol{y}_k = C\boldsymbol{x}_k$ is computed to obtain the corresponding output vector. During training, $\boldsymbol{y}_k$ is used as the input sample, while $\boldsymbol{x}_k$ serves as the reference target sample. This supervised pairing encourages the network to learn an implicit approximation of the inverse impedance operator, *i.e.*, to approximate the mapping C^{-1}.

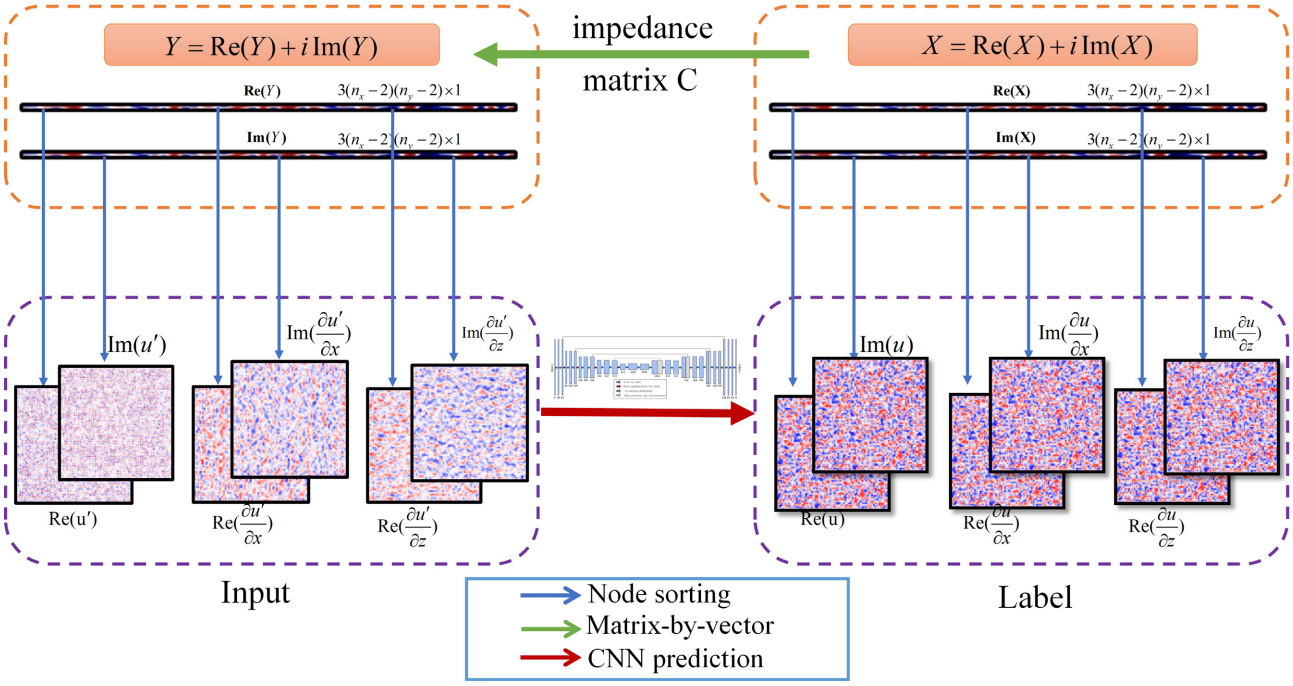

Figure 3. The construction process of training data (Re(.) and Im(.) represent the real part and imaginary part, respectively).

During preprocessing, both X and Y undergo structured transformations, including stride-3 sampling, separation of real and imaginary parts, and channel reorganization based on row-major ordering. The stride-3 operation is consistent with the underlying interleaved storage layout $(u, \partial u/\partial x, \partial u/\partial z)$, where the three quan-

tities are stored sequentially at each spatial node. Applying stride-3 along the storage dimension simply separates these physically coupled variables into distinct channels without discarding any information. The subsequent channel reorganization arranges them into contiguous feature maps suitable for convolutional processing.

3.3. Training Process

At the beginning of networks training, the layer parameters are initialized following the default Kaiming scheme [32]. Then, the parameter is gradually optimized by the backpropagation algorithm to construct a nonlinear mapping function $X = Net(Y, \theta)$ from the input data Y to the label data X. $\theta = \{K_w, b_w\}$ is the network parameter (including convolution weight coefficient k_w and bias variable b_w). Given that the gradient of the L1 loss function is not excessively amplified by large errors from outliers, this property effectively prevents the model parameters from being dominated by a few extreme samples, thereby enhancing the robustness of networks [33]. To be specific, the objective function is optimized by

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} \left\| x_i - Net(y_i; \theta) \right\|, \tag{7}$$

where m represents batch size, and the variables (x_i, y_i) can be selected from the training set. For the optimization of equation (7), the Adam optimizer (combining both momentum mechanisms and adaptive learning rate adjustment) is typically employed for parameter update. By integrating the first-order moment estimate of the gradients with a second-order moment correction term, its parameter update rule can be formulated [34]

$$\theta_t = \theta_{t-1} - \alpha \cdot \hat{m}_t \Big/ \left(\sqrt{\hat{v}_t} + \varepsilon \right), \tag{8}$$

with

$$\begin{cases} m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t, \hat{m}_t = m_t \Big/ \left(1-\beta_1^t\right) \\ v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2, \hat{v}_t = v_t \Big/ \left(1-\beta_2^t\right) \end{cases}, \tag{9}$$

where α denotes the line search step length, fuzz factor $\varepsilon = 10^{-8}$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. Although the Adam optimizer adapts the learning rate, applying an appropriate learning rate decay strategy can further enhance the performance of optimizer [35]. Thus, a scheduler (StepLR) is employed to decay the learning rate. During the training process, the networks continuously optimize its parameters by minimizing the L1 loss function, thereby progressively enhancing its representational capability.

3.4. DNNs-Preconditioned BICGSTAB Method

After training, the neural network with optimized parameters $\hat{\theta}$ defines a prediction operator $Net(\cdot, \hat{\theta})$, which serves as a data-driven approximation to the inverse impedance operator. This operator is embedded into the BICGSTAB framework by replacing the conventional preconditioning steps (Steps 8 and 11

in Algorithm 1) with network-based predictions, resulting in the proposed DNNs-preconditioned BICGSTAB method. During each iteration, the network directly maps the residual-related vectors to approximate solutions of the corresponding preconditioned systems, thereby avoiding the explicit solution of auxiliary linear systems. The complete iterative procedure of the proposed solver is summarized in Algorithm 2.

Algorithm 2. DNNs-preconditioned BICGSTAB method.

Require: Initial guess u_0 and tolerance $\varepsilon > 0$
Ensure: $\| Cu - s \|_2 / \| Cu_0 - s \|_2 < \varepsilon$
Set $p_1 = r_0$ and choose $\hat{r}_0$ such that $(r_0, \hat{r}_0) \neq 0$
for $k = 1, 2, \ldots$ **do**
  $\rho_{k-1} = (r_{k-1}, \hat{r}_0)$
  if $k > 1$ **then**
    Compute β_{k-1} and update the search direction p_k
  end if
  $\hat{\boldsymbol{p}}_k = \boldsymbol{P}^{-1}\boldsymbol{p}_k \cdots\cdots\cdots\cdots \hat{p}_k := \boldsymbol{Net}(p_k, \hat{\theta})$
  Compute v_k, α_k, q_k, and u_k
  $\hat{\boldsymbol{q}}_k = \boldsymbol{P}^{-1}\boldsymbol{q}_k \cdots\cdots\cdots\cdots \hat{q}_k := \boldsymbol{Net}(q_k, \hat{\theta})$
  Compute ω_k, update u_{k+1} and r_k
  if $\| r_k \|_2 < \varepsilon \cdot \| r_0 \|_2$ **then**
    break
  end if
end for

It is worth noting that directly approximating the exact inverse of the impedance matrix using a deep neural network (*i.e.*, enforcing $Net\left(\cdot, \hat{\theta}\right) \approx C^{-1}$) would, in principle, lead to near one-step convergence of the iterative solver. However, in high-dimensional settings, achieving such an approximation requires an exponentially growing number of training samples and optimization iterations, resulting in prohibitive computational cost. To balance approximation accuracy and efficiency, we adopt an optimized training strategy that enables the network to capture the dominant inverse characteristics of the impedance matrix without explicitly learning its full inverse. This design yields an effective data-driven preconditioner that significantly improves convergence while maintaining practical computational complexity.

4. Numerical Experiments

To assess the effectiveness of the proposed DNNs-BICGSTAB scheme for solving severely ill-conditioned linear system, we benchmark it against three reference solvers: BICGSTAB equipped with an ILU preconditioner (ILU-BICGSTAB), BICGSTAB using the M_0 preconditioner (M_0-BICGSTAB), and the standard BICGSTAB algorithm without preconditioning (None-BICGSTAB). Numerical tests are performed on three typical velocity models, namely a homogeneous

model, a two-layer model, and Marmousi model. The main parameter settings associated with each model are summarized in Table 1.

Table 1. Basic parameter sets for various media problems.

	Homogeneous	Two-layer	Marmousi
Computing area	$0 \le x \le 5.2$ km	$0 \le x \le 5.2$ km	$0 \le x \le 7.68$ km
	$0 \le z \le 5.2$ km	$0 \le z \le 5.2$ km	$0 \le z \le 2.56$ km
Spatial step (h)	0.04 km	0.04 km	0.04 km
PML layers	$\delta = 10h$	$\delta = 10h$	$\delta = 10h$
Dominant f	$f_0 = 10$ Hz	$f_0 = 10$ Hz	$f_0 = 10$ Hz
Grid scale	130×130	130×130	210×80
Matrix size ($3N \times 3N$)	49152×49152	49152×49152	48672×48672

In all numerical simulations, a Ricker wavelet is consistently adopted as the source term on the right-hand side of equation (4), and its corresponding frequency-domain expression is defined as

$$\mathrm{s}(f) = \frac{\sqrt{2}A}{\pi f_0}\left(\frac{f}{f_0}\right)^2 \mathrm{e}^{-\left(\frac{f}{f_0}\right)^2}, \tag{10}$$

where A is the source amplitude that is set to 1e−4, f denotes the frequency, and f_0 represents the dominant frequency of the source. In this section, the stopping criteria for all BICGSTAB-based solvers are specified as either reaching a maximum of 10,000 iterations or achieving a relative residual norm below the prescribed tolerance of 10^{-5}.

All experiments are conducted on the Kaggle platform (https://www.kaggle.com), which provides a cloud-based computing environment with GPU acceleration for neural network training and optimization. The computational resources consist of a 4-core Intel Xeon processor with 16 GB of RAM and an NVIDIA Tesla P100 GPU equipped with 16 GB of memory. The proposed network is trained independently on three representative velocity models. Specifically, model parameters are optimized using the Adam optimizer with an initial learning rate of 5e−5, while a stepwise learning rate scheduler is employed to reduce the learning rate by a factor of two every 50 epochs. The training dataset comprises approximately 9000 paired samples, which are randomly split into training and validation subsets, with 20% of the data allocated for validation. Training is performed using a batch size of 32 over roughly 300 epochs, and each training run takes about 3 hours on average. For consistency and reproducibility, all experimental configurations including data split, random seed, and hyperparameter setting are kept identical across different runs.

4.1. Homogeneous Medium

We firstly perform a numerical simulation using a homogeneous medium defined

over a 5.2 km × 5.2 km square domain, with a constant wave velocity of 2 km/s. A source with the frequency of 20 Hz is placed at coordinates (2.6 km, 0.45 km). The comparison results are presented in Figure 4(a) and Table 2, where Figure 4(a) shows the convergence curves of three solvers and the column 2 of Table 2 lists the number of iteration and computing time. The results indicate that both ILU-BICGSTAB, M_0-BICGSTAB, and DNNs-BICGSTAB accelerate convergence to some extent compared to the baseline None-BICGSTAB. It is worth noting that DNNs-BICGSTAB shows better convergence performance than other preconditioners, demonstrating its superior numerical efficiency. These findings suggest that deep learning techniques great potential for enhancing preconditioning strategies in numerical simulations. In addition, wavefield snapshot obtained using DNNs-BICGSTAB is presented (Figure 5(a)) to validate the accuracy and reliability of the proposed method. Since the same stopping criteria are used in the iterative solver, the use of a preconditioner does not affect the numerical consistency of the solution.

To further investigate the applicability and robustness of the proposed method under discretization schemes of different orders, equation (1) is additionally solved using a sixth-order NAD formulation. For consistency and fairness, all numerical parameters are kept identical to those adopted in the previously discussed fourth-order NAD scheme. After assembling the resulting linear system, the four solvers are applied, and the corresponding BICGSTAB convergence histories are presented in Figure 4(d). As illustrated in the figure, the proposed DNNs-BICGSTAB method continues to demonstrate clear performance advantages even when a higher-order discretization is employed. In particular, it converges significantly faster than ILU-BICGSTAB, M_0-BICGSTAB, and the unpreconditioned BICGSTAB. These results indicate that the effectiveness of the proposed approach is not restricted to a specific discretization order, but remains robust under higher-order NAD schemes, underscoring its suitability for more demanding and complex numerical simulations.

To evaluate how the training overhead influences the overall computational efficiency of the proposed approach in multi-source scenarios, a comparative study is performed. To be specific, simulation is conducted at a fixed frequency of 20 Hz, where seismic sources are uniformly distributed along the horizontal direction. The number of sources is gradually increased from 50 to 500, corresponding to source spacings ranging from 104 m to 10.4 m, thereby forming a sequence of test cases with increasing computational complexity. Figure 6 summarizes the total runtime of different solvers as a function of the number of sources. Several important observations can be drawn from the results. Although the DNN-based preconditioning solver introduces additional training costs for each individual linear system, its overall computational efficiency improves progressively as the number of sources increases. In particular, when the number of sources exceeds 150, the total runtime of DNNs-BICGSTAB becomes lower than that of the unpreconditioned BICGSTAB (None-BICGSTAB). As the source count further increases to 300, it also outperforms ILU-BICGSTAB, and at approximately 330 sources, it surpasses the M_0-BICGSTAB method. More importantly, among all

compared solvers, DNNs-BICGSTAB exhibits the slowest growth rate in computational time, clearly demonstrating its superior scalability and efficiency in large-scale multi-source simulations.

Table 2. The number of iterative steps (Iter) and computing time (Time: second) with respect to the listed preconditioned BICGSTAB methods for various media at a single source and $f = 20$ Hz.

Methods	Homogeneous		Two-layer		Marmousi	
	Iter	Time(s)	Iter	Time(s)	Iter	Time(s)
None-BICGSTAB	4636	26.69	4940	27.45	4464	23.14
ILU-BICGSTAB	1385	14.75	1821	19.39	1568	16.50
M_0-BICGSTAB	1268	13.50	1695	18.05	1395	14.86
DNNs-BICGSTAB	56	2.39	63	2.69	80	3.42

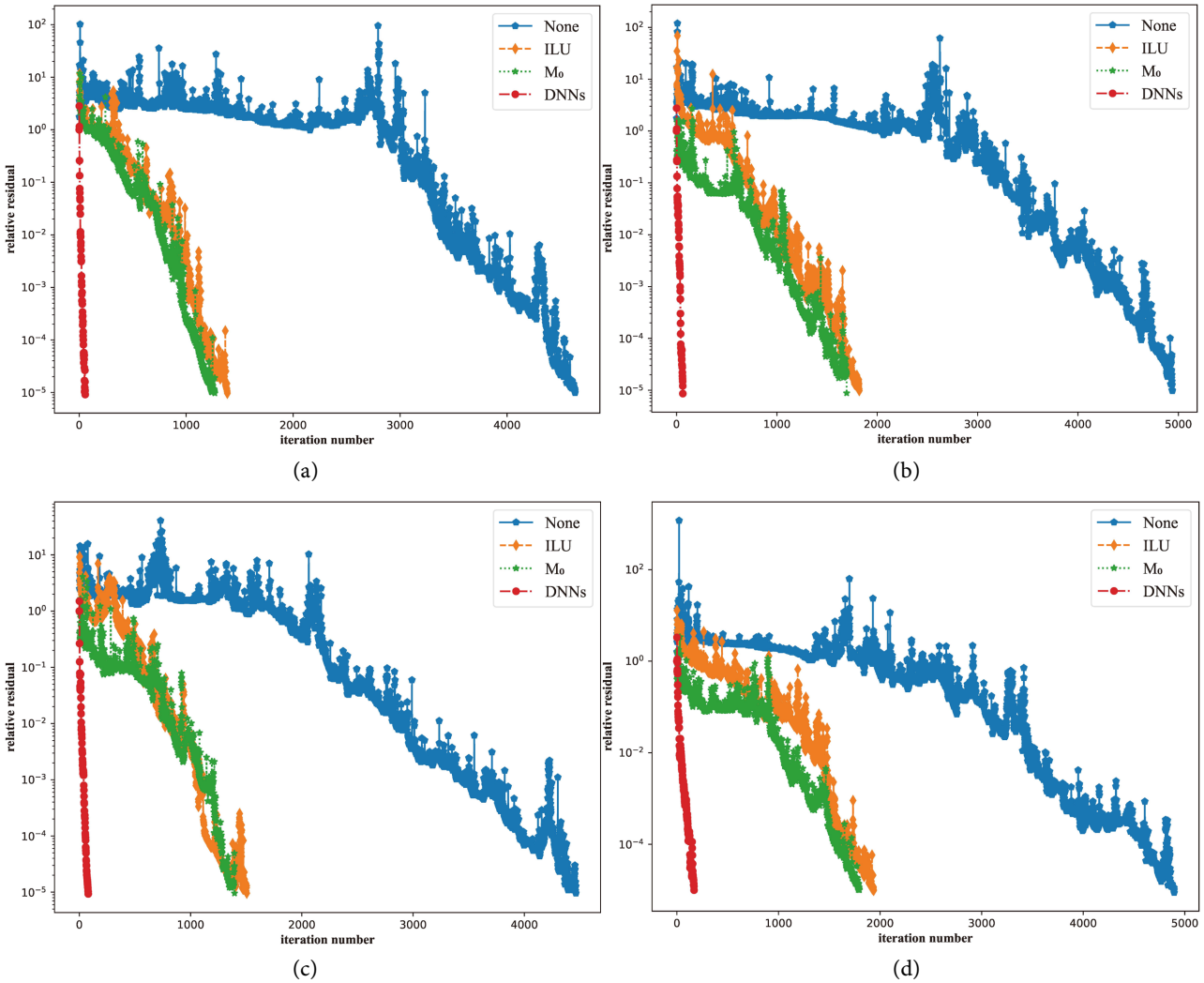

Figure 4. Convergence curves of different preconditioned BICGSTAB methods applied to various velocity media: (a) Homogeneous medium, (b) Two-layer medium, (c) Marmousi medium, and (d) Homogeneous medium discretized using the sixth-order NAD scheme.

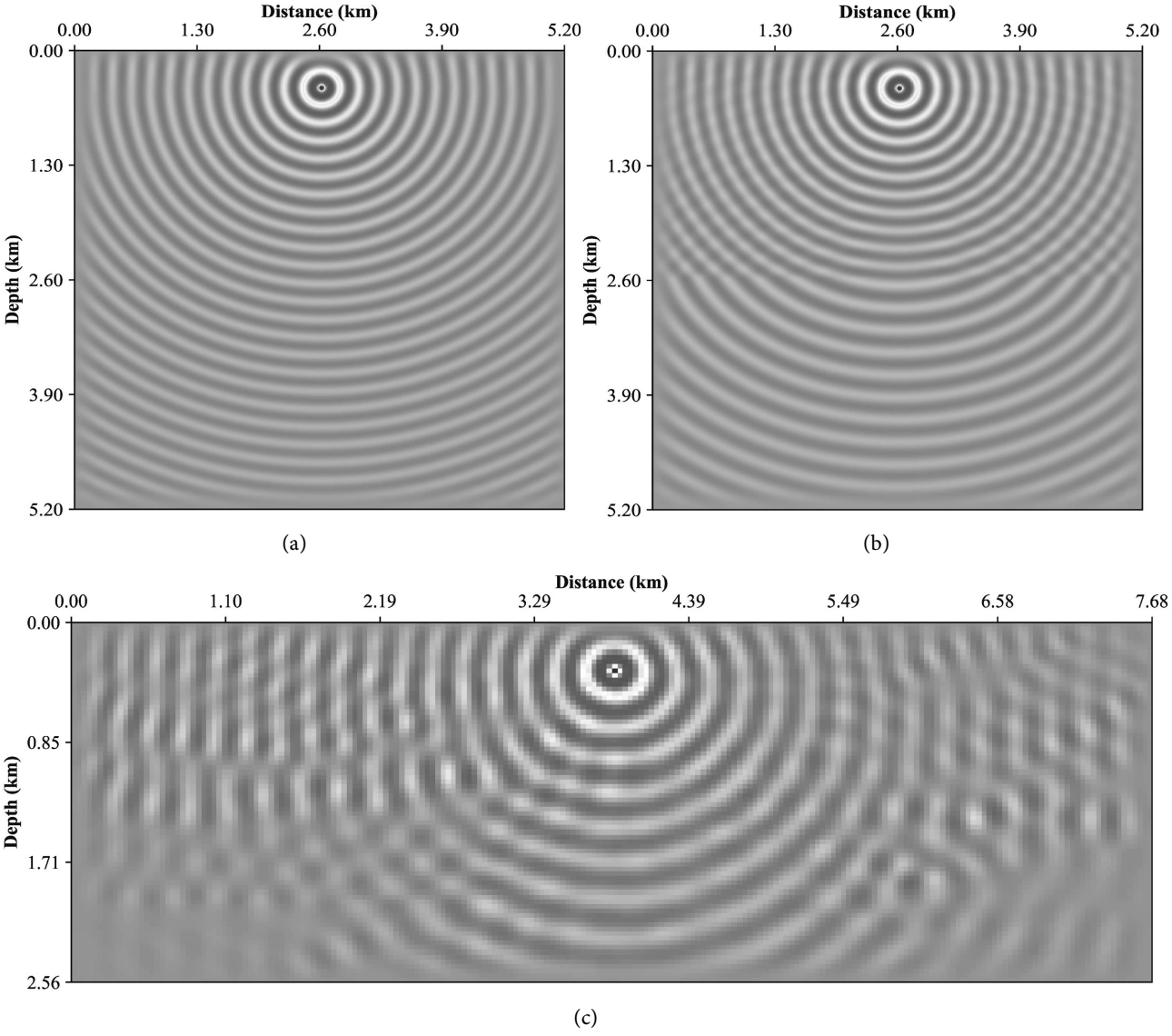

Figure 5. Mono-frequency wavefield snapshot at frequency $f = 20$ Hz in various media: (a) Homogeneous medium, (b) Two-layer medium, (c) Marmousi medium.

4.2. Two-Lay Medium

Next, for evaluating the performance of DNNs-BICGSTAB under varying medium configurations, a square two-layer velocity model of 5.2 km per side is constructed for numerical testing. The velocity in the upper layer is set to 2.0 km/s, and that in the lower layer to 2.5 km/s, simulating a typical geological interface with a clear velocity difference. The source is placed at the outer layer of the PML absorbing boundary with a frequency of 20 Hz. Figure 4(b) presents the residual convergence curves of four solvers, and the corresponding iterative numbers and computational times are listed in the third column of Table 2. These results show that None-BICGSTAB exhibits very low convergence efficiency, highlighting the necessity of introducing preconditioning techniques. While ILU and M_0 preconditioners improve convergence to some extent, the reduction in

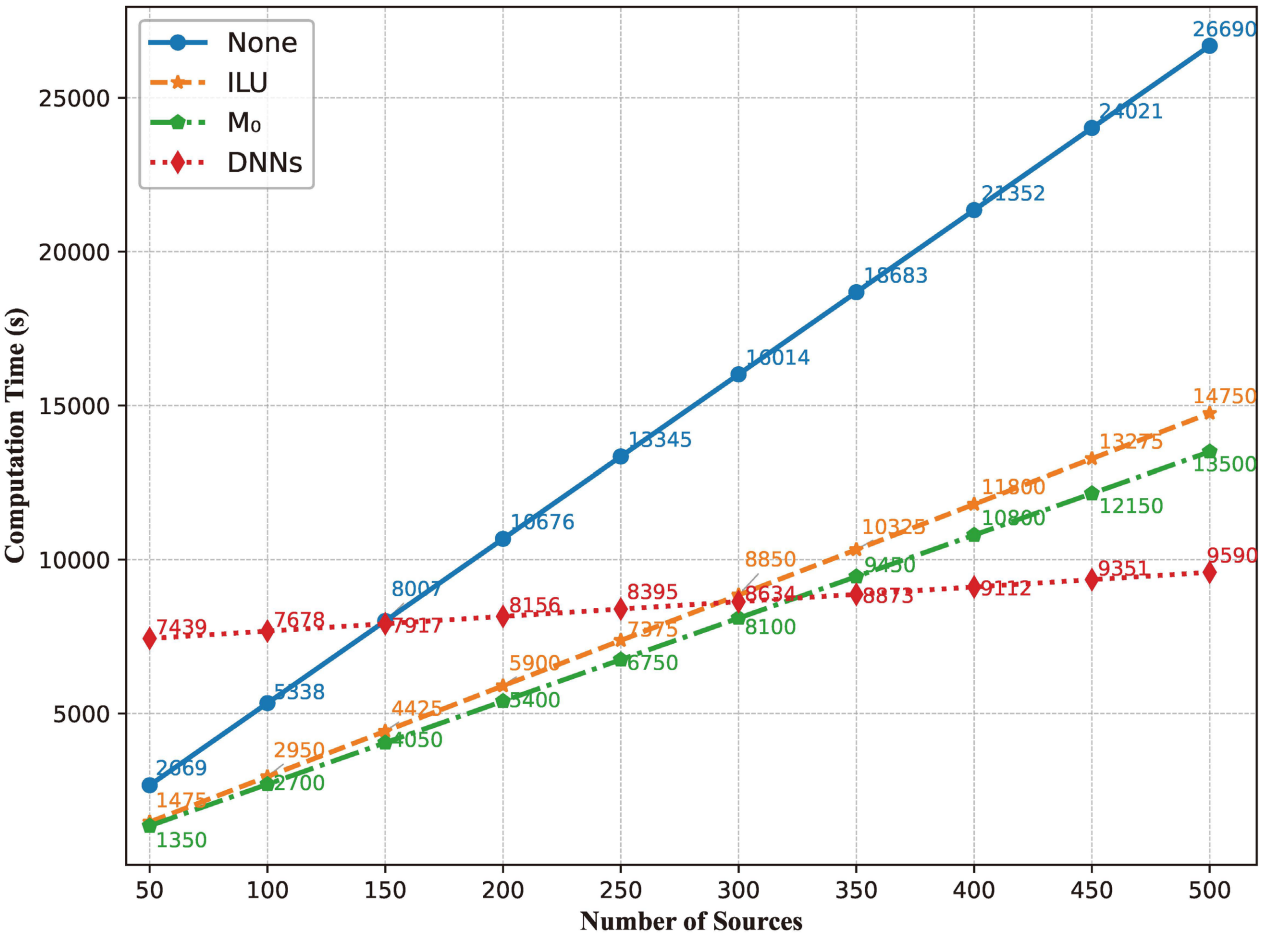

Figure 6. Computing time (involving training time) versus the number of sources for various preconditioned methods applied to the wavefield simulation of Homogeneous medium.

the number of iterations remains limited. In contrast, DNNs-BICGSTAB method demonstrates stable and efficient convergence throughout the entire iterative process, outperforming the other three solvers. These findings confirm that DNNs-based preconditioning strategy maintains robust numerical performance even in the presence of strong velocity contrasts at media interfaces. Figure 5(b) displays the wavefield snapshot under this medium, further validating the accuracy and reliability of the proposed method in two-lay settings.

To further evaluate the general applicability of the proposed method over a range of frequencies, several representative frequency values commonly adopted in seismic exploration are selected for comparative analysis. For each frequency, the network is retrained independently to capture the corresponding characteristics of the impedance matrix. Table 3 summarizes the iterative counts and computational costs of different solvers at these frequencies. The results indicate that DNNs-BICGSTAB method consistently achieves robust performance at all tested frequencies, reducing the number of iterations compared with ILU-BICGSTAB, M_0-BICGSTAB, and the unpreconditioned BICGSTAB. These observations demonstrate that the proposed neural network-based preconditioning approach

is effective and stable when applied across different frequency conditions.

Table 3. The iterative number (Iter) and running time (Time: second) of preconditioned BICGSTAB methods for solving single-source problem at various frequencies in Two-layer medium.

f (Hz)	None-BICGSTAB		ILU-BICGSTAB		M_0-BICGSTAB		DNNs-BICGSTAB	
	Iter	Time(s)	Iter	Time(s)	Iter	Time(s)	Iter	Time(s)
10	6596	36.65	2590	27.58	2263	24.10	126	5.38
15	5076	28.21	1994	21.23	1838	19.57	80	3.42
20	4940	27.45	1821	19.39	1695	18.05	63	2.69
25	4893	27.19	1920	20.44	1726	18.38	53	2.26
30	6015	33.42	2362	25.15	2113	22.50	113	4.82

4.3. Marmousi Medium

Finally, to assess the applicability of the proposed approach in geologically complex environments, the classical Marmousi velocity model (shown in Figure 7) is adopted as a challenging test case. The seismic source is positioned at (3.84 km, 0.36 km) and operates at a frequency of 20 Hz. The convergence histories of the different solvers are illustrated in Figure 4(c), while the corresponding iteration counts and computational times are reported in the fourth column of Table 2. Representative wavefield snapshots for this model are further presented in Figure 5(c). The results indicate that, even in the presence of strong lateral velocity variations and complex geological structures, the DNN-based preconditioning strategy maintains stable and efficient convergence, clearly demonstrating its robustness and effectiveness for large-scale seismic wavefield simulations in realistic geological settings.

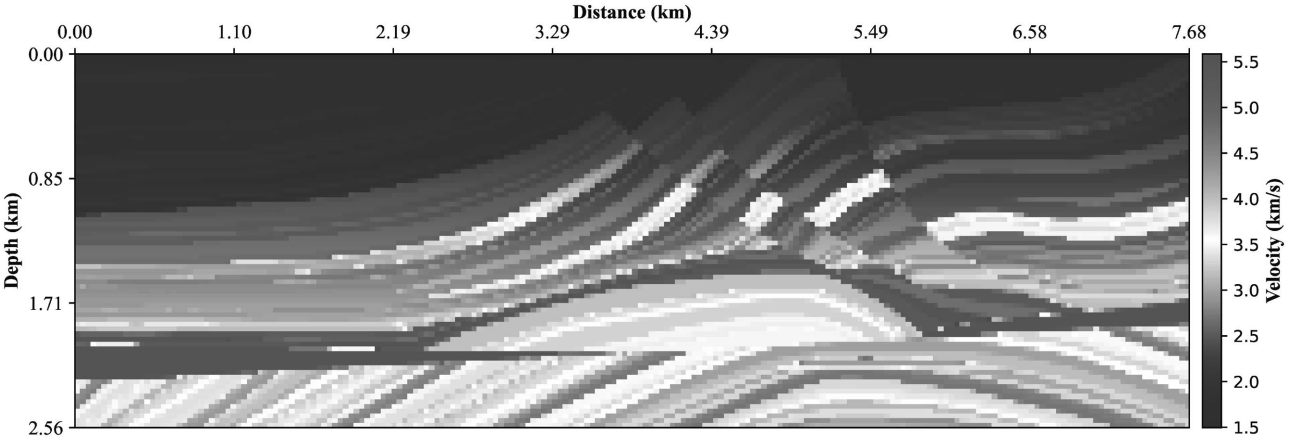

Figure 7. Marmousi velocity structure of experimental medium.

5. Conclusion

This paper presents a deep neural network-based preconditioning strategy embedded in the BICGSTAB framework for solving large-scale, highly ill-condi-

tioned linear systems arising from the NAD discretization of the frequency-domain wave equation. By learning inverse-related features of the impedance matrix, the proposed data-driven preconditioner improves the convergence behavior of Krylov subspace iteration without explicitly forming the inverse operator. Numerical results obtained under different discretization orders, frequency values, source configurations, and velocity models demonstrate that the proposed DNNs-BICGSTAB method consistently outperforms classical preconditioned and unpreconditioned BICGSTAB solvers. In particular, our method shows clear advantages over commonly used ILU and M_0 preconditioners. From a numerical linear algebra viewpoint, the learned preconditioner effectively enhances the residual reduction mechanism and exhibits favorable scalability, particularly when multiple right-hand sides are involved. The framework is general and can be naturally extended to three-dimensional problems. Future work will focus on theoretical analysis, large-scale 3D benchmarking, and systematic evaluation against more advanced classical preconditioners.

Acknowledgement

This study was supported by Beijing Natural Science Foundation (Grant No. 8232023) and Beijing Science and Technology Planning Project (Grant No. KM202111232009).

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Li, Y., Jia, X., Wu, X. and Geng, Z. (2022) Deep Learning for Enhancing Multisource Reverse Time Migration. *IEEE Transactions on Geoscience and Remote Sensing*, **60**, 1-13. https://doi.org/10.1109/tgrs.2022.3206283

[2] Moczo, P., Robertsson, J.O.A. and Eisner, L. (2007) The Finite-Difference Time-Domain Method for Modeling of Seismic Wave Propagation. In: *Advances in Geophysics*, Elsevier, 421-516. https://doi.org/10.1016/s0065-2687(06)48008-0

[3] Liu, S., Li, X., Wang, W. and Liu, Y. (2014) A Mixed-Grid Finite Element Method with PML Absorbing Boundary Conditions for Seismic Wave Modelling. *Journal of Geophysics and Engineering*, **11**, Article 055009. https://doi.org/10.1088/1742-2132/11/5/055009

[4] Kosloff, D.D. and Baysal, E. (1982) Forward Modeling by a Fourier Method. *Geophysics*, **47**, 1402-1412. https://doi.org/10.1190/1.1441288

[5] Komatitsch, D., Tsuboi, S., Tromp, J., Levander, A. and Nolet, G. (2005) The Spectral-Element Method in Seismology. In: Levander, A. and Nolet, G., Eds., *Geophysical Monograph Series*, American Geophysical Union, 205-227. https://doi.org/10.1029/157gm13

[6] Yang, D., Lu, M., Wu, R. and Peng, J. (2004) An Optimal Nearly Analytic Discrete Method for 2D Acoustic and Elastic Wave Equations. *Bulletin of the Seismological Society of America*, **94**, 1982-1992. https://doi.org/10.1785/012003155

[7] Lang, C., Li, Q., Zhou, Y., He, X. and Han, R. (2020) A High-Precision Low-Dispersive Nearly Analytic Difference Method with Its Application in Frequency-Domain

Seismic Waveform Inversion. *Exploration Geophysics*, **51**, 355-377. https://doi.org/10.1080/08123985.2019.1699786

[8] Liu, S., Lang, C., Yang, H. and Wang, W. (2018) A Developed Nearly Analytic Discrete Method for Forward Modeling in the Frequency Domain. *Journal of Applied Geophysics*, **149**, 25-34. https://doi.org/10.1016/j.jappgeo.2017.12.007

[9] Pratt, R.G. (1999) Seismic Waveform Inversion in the Frequency Domain; Part 1, Theory and Verification in a Physical Scale Model. *Geophysics*, **64**, 888-901. https://doi.org/10.1190/1.1444597

[10] Plessix, R.-E. (2006) A Review of the Adjoint-State Method for Computing the Gradient of a Functional with Geophysical Applications. *Geophysical Journal International*, **167**, 495-503. https://doi.org/10.1111/j.1365-246x.2006.02978.x

[11] Erlangga, Y.A., Vuik, C. and Oosterlee, C.W. (2004) On a Class of Preconditioners for Solving the Helmholtz Equation. *Applied Numerical Mathematics*, **50**, 409-425. https://doi.org/10.1016/j.apnum.2004.01.009

[12] Saad, Y. (2003) Iterative Methods for Sparse Linear Systems. SIAM.

[13] van der Vorst, H.A. (1992) Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-Cg for the Solution of Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, **13**, 631-644. https://doi.org/10.1137/0913035

[14] Saad, Y. and Schultz, M.H. (1986) GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, **7**, 856-869. https://doi.org/10.1137/0907058

[15] Du, Z., Liu, J., Liu, W. and Li, C. (2016) Frequency-Space Domain Acoustic Wave Simulation with the BiCGstab (ℓ) Iterative Method. *Journal of Geophysics and Engineering*, **13**, 70-77. https://doi.org/10.1088/1742-2132/13/1/70

[16] Wang, N. and Lang, C. (2025) Deep Transfer Learning-Based Preconditioned GMRES Method for Acoustic Reverse Time Migration in the Frequency Domain. *IEEE Transactions on Geoscience and Remote Sensing*, **63**, 1-12. https://doi.org/10.1109/tgrs.2025.3556842

[17] Lang, C. and Ren, Z. (2013) Inexact Rotated Block Triangular Preconditioners for a Class of Block Two-by-Two Matrices. *Journal of Engineering Mathematics*, **93**, 87-98. https://doi.org/10.1007/s10665-013-9674-1

[18] Varilsuha, D. and Candansayar, M.E. (2018) 3D Magnetotelluric Modeling by Using Finite-Difference Method: Comparison Study of Different Forward Modeling Approaches. *Geophysics*, **83**, WB51-WB60. https://doi.org/10.1190/geo2017-0406.1

[19] Taghibakhshi, A., Nytko, N., Zaman, T.U., MacLachlan, S., Olson, L. and West, M. (2023) MG-GNN: Multigrid Graph Neural Networks for Learning Multilevel Domain Decomposition Methods. *Proceedings of the International Conference on Machine Learning*, Honolulu, 23-29 July 2023, 33381-33395.

[20] Jiang, L., Wang, L., Chu, X., Xiao, Y. and Zhang, H. (2023) PhyGNNet: Solving Spatiotemporal PDEs with Physics-Informed Graph Neural Network. *Proceedings of the* 2023 2*nd Asia Conference on Algorithms*, *Computing and Machine Learning*, Shanghai, 17-19 March 2023, 143-147. https://doi.org/10.1145/3590003.3590029

[21] Huang, R., Li, R. and Xi, Y. (2022) Learning Optimal Multigrid Smoothers via Neural Networks. *SIAM Journal on Scientific Computing*, **45**, S199-S225. https://doi.org/10.1137/21m1430030

[22] Lang, C. and Yang, D. (2016) A Nearly Analytic Discrete Method for Solving the Acoustic-Wave Equations in the Frequency Domain. *Geophysics*, **82**, T43-T57. https://doi.org/10.1190/geo2016-0248.1

[23] Greenbaum, A. (1997) Iterative Methods for Solving Linear Systems. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611970937

[24] Ronneberger, O., Fischer, P. and Brox, T. (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W. and Frangi, A., Eds., *Lecture Notes in Computer Science*, Springer International Publishing, 234-241. https://doi.org/10.1007/978-3-319-24574-4_28

[25] Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F. and Courville, A. (2019) On the Spectral Bias of Neural Networks. *Proceedings of the International Conference on Machine Learning*, Long Beach, 9-15 June 2019, 5301-5310.

[26] Azulay, Y. and Treister, E. (2022) Multigrid-Augmented Deep Learning Preconditioners for the Helmholtz Equation. *SIAM Journal on Scientific Computing*, **45**, S127-S151. https://doi.org/10.1137/21m1433514

[27] Simonyan, K. and Zisserman, A. (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition. https://arxiv.org/abs/1409.1556

[28] Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P. (2002) Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, **86**, 2278-2324. https://doi.org/10.1109/5.726791

[29] Dumoulin, V. and Visin, F. (2016) A Guide to Convolution Arithmetic for Deep Learning. https://arxiv.org/abs/1603.07285

[30] Zhang, H., Cisse, M., Dauphin, Y.N. and Lopez-Paz, D. (2017) Mixup: Beyond Empirical Risk Minimization. https://arxiv.org/abs/1710.09412

[31] Badrinarayanan, V., Kendall, A. and Cipolla, R. (2017) SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **39**, 2481-2495. https://doi.org/10.1109/tpami.2016.2644615

[32] He, K., Zhang, X., Ren, S. and Sun, J. (2015) Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification. 2015 *IEEE International Conference on Computer Vision* (*ICCV*), Washington, 7-13 December 2015, 1026-1034. https://doi.org/10.1109/iccv.2015.123

[33] Barron, J.T. (2019) A General and Adaptive Robust Loss Function. 2019 *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*), Long Beach, 15-20 June 2019, 4331-4339. https://doi.org/10.1109/cvpr.2019.00446

[34] Kingma, D.P. (2014) Adam: A Method for Stochastic Optimization. https://arxiv.org/abs/1412.6980

[35] Loshchilov, I. and Hutter, F. (2017) Decoupled Weight Decay Regularization. https://arxiv.org/abs/1711.05101

[36] Lang, C., Liu, S.L., Yang, X.T. and Xu, X.W. (2022) Frequencydomain Acoustic reVerse Time Migration Based on Improved NAD Method. *Chinese Journal of Geophysics*, **65**, 1071-1085.

Appendix A

A1. Derivation Process of Sparse Impedance Matrix

In this paper, the spatial step sizes in the horizontal direction x and the vertical direction z are set to be equal and uniformly denoted as h. The classical fourth-order NAD finite difference scheme is employed, and the difference template for the high-order partial derivatives of equation (3) in x is

$$\begin{aligned}\frac{\partial^2 u_{i,j}}{\partial x^2} &\approx \frac{1}{h^2}\left(a_{-1}u_{i-1,j}+a_0u_{i,j}+a_1u_{i+1,j}\right)+\frac{1}{h}\left(b_{-1}\frac{\partial u_{i-1,j}}{\partial x}+b_0\frac{\partial u_{i,j}}{\partial x}+b_1\frac{\partial u_{i+1,j}}{\partial x}\right)\\ \frac{\partial^3 u_{i,j}}{\partial x^3} &\approx \frac{1}{h^3}\left(c_{-1}u_{i-1,j}+c_0u_{i,j}+c_1u_{i+1,j}\right)+\frac{1}{h^2}\left(d_{-1}\frac{\partial u_{i-1,j}}{\partial x}+d_0\frac{\partial u_{i,j}}{\partial x}+d_1\frac{\partial u_{i+1,j}}{\partial x}\right)\end{aligned} \tag{11}$$

to derive the coefficients of the above equation (11), it is necessary to use the Taylor formula to expand, and the specific process is

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0\\ -1 & 0 & 1 & 1 & 1 & 1\\ \frac{1}{2} & 0 & \frac{1}{2} & -1 & 0 & 1\\ -\frac{1}{6} & 0 & \frac{1}{6} & \frac{1}{2} & 0 & \frac{1}{2}\\ \frac{1}{24} & 0 & \frac{1}{24} & -\frac{1}{6} & 0 & \frac{1}{6}\\ -\frac{1}{120} & 0 & \frac{1}{120} & \frac{1}{24} & 0 & \frac{1}{24} \end{pmatrix}\begin{pmatrix} a_{-1}\\ a_0\\ a_1\\ b_{-1}\\ b_0\\ b_1 \end{pmatrix}=\begin{pmatrix} 0\\ 0\\ 1\\ 0\\ 0\\ 0 \end{pmatrix}, \tag{12}$$

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0\\ -1 & 0 & 1 & 1 & 1 & 1\\ \frac{1}{2} & 0 & \frac{1}{2} & -1 & 0 & 1\\ -\frac{1}{6} & 0 & \frac{1}{6} & \frac{1}{2} & 0 & \frac{1}{2}\\ \frac{1}{24} & 0 & \frac{1}{24} & -\frac{1}{6} & 0 & \frac{1}{6}\\ -\frac{1}{120} & 0 & \frac{1}{120} & \frac{1}{24} & 0 & \frac{1}{24} \end{pmatrix}\begin{pmatrix} c_{-1}\\ c_0\\ c_1\\ d_{-1}\\ d_0\\ d_1 \end{pmatrix}=\begin{pmatrix} 0\\ 0\\ 0\\ 1\\ 0\\ 0 \end{pmatrix}, \tag{13}$$

solving the linear equations of (12) and (13) yields the coefficients of NAD method:

$$\begin{cases} a_{-1}=2\\ a_0=-4\\ a_{-1}=2 \end{cases}\quad \begin{cases} b_{-1}=\frac{1}{2}\\ b_0=0\\ b_1=-\frac{1}{2} \end{cases}\quad \begin{cases} c_{-1}=-\frac{15}{2}\\ c_0=0\\ c_1=\frac{15}{2} \end{cases}\quad \begin{cases} d_{-1}=\frac{3}{2}\\ d_0=-12\\ d_1=\frac{3}{2} \end{cases} \tag{14}$$

According to Lang and Yang [22] and Liu *et al.* [8], the discrete formulas for other higher-order partial differential terms in equation (3) are as follows:

$$\begin{aligned}\frac{\partial^2 u_{i,j}}{\partial x\partial z} \approx & \frac{1}{4h^2}\left(a_1 u_{i+1,j+1}+a_{-1}u_{i-1,j-1}-a_1 u_{i-1,j+1}-a_{-1}u_{i+1,j-1}\right)\\ & +\frac{1}{4h}\left(b_1\frac{\partial u_{i+1,j+1}}{\partial z}+b_{-1}\frac{\partial u_{i-1,j-1}}{\partial z}-b_1\frac{\partial u_{i-1,j+1}}{\partial z}-b_{-1}\frac{\partial u_{i+1,j-1}}{\partial z}\right)\\ & +\frac{1}{4h}\left(b_1\frac{\partial u_{i+1,j+1}}{\partial x}+b_{-1}\frac{\partial u_{i-1,j-1}}{\partial x}+b_1\frac{\partial u_{i-1,j+1}}{\partial x}+b_{-1}\frac{\partial u_{i+1,j-1}}{\partial x}\right.\\ & \left.+2b_0\frac{\partial u_{i,j}}{\partial x}\right)\end{aligned} \tag{15}$$

$$\begin{aligned}\frac{\partial^3 u_{i,j}}{\partial x^2\partial z} \approx & \frac{1}{6h^3}\left(c_1 u_{i+1,j+1}+c_1 u_{i-1,j+1}-2c_1 u_{i,j+1}+c_{-1}u_{i+1,j-1}+c_{-1}u_{i-1,j-1}\right.\\ & \left.-2c_{-1}u_{i,j-1}\right)+\frac{1}{6h^2}\left(d_1\frac{\partial u_{i+1,j+1}}{\partial x}+d_{-1}\frac{\partial u_{i-1,j-1}}{\partial x}-d_1\frac{\partial u_{i-1,j+1}}{\partial x}\right.\\ & \left.-d_{-1}\frac{\partial u_{i+1,j-1}}{\partial x}\right)+\frac{1}{6h^2}\left(d_1\frac{\partial u_{i+1,j+1}}{\partial z}+d_{-1}\frac{\partial u_{i-1,j-1}}{\partial z}+d_1\frac{\partial u_{i-1,j+1}}{\partial z}\right.\\ & \left.+d_{-1}\frac{\partial u_{i+1,j-1}}{\partial z}-2d_1\frac{\partial u_{i,j+1}}{\partial z}-2d_{-1}\frac{\partial u_{i,j-1}}{\partial z}\right)\end{aligned} \tag{16}$$

$$\begin{aligned}\frac{\partial^3 u_{i,j}}{\partial x\partial z^2} \approx & \frac{1}{6h^3}\left(c_1 u_{i+1,j+1}-c_{-1}u_{i+1,j-1}-c_1 u_{i-1,j+1}+c_{-1}u_{i-1,j-1}-2c_{-1}u_{i-1,j}\right.\\ & \left.-2c_1 u_{i+1,j}-2c_0 u_{i,j}\right)+\frac{1}{6h^2}\left(d_1\frac{\partial u_{i+1,j+1}}{\partial z}+d_{-1}\frac{\partial u_{i-1,j-1}}{\partial z}\right.\\ & \left.-d_1\frac{\partial u_{i-1,j+1}}{\partial z}-d_{-1}\frac{\partial u_{i+1,j-1}}{\partial z}\right)+\frac{1}{6h^2}\left(d_1\frac{\partial u_{i+1,j+1}}{\partial x}+d_{-1}\frac{\partial u_{i-1,j-1}}{\partial x}\right.\\ & \left.+d_1\frac{\partial u_{i-1,j+1}}{\partial x}+d_{-1}\frac{\partial u_{i+1,j-1}}{\partial x}-2d_1\frac{\partial u_{i+1,j}}{\partial x}-2d_{-1}\frac{\partial u_{i-1,j}}{\partial x}\right)\end{aligned} \tag{17}$$

when the above NAD grid difference template is used to discretize the frequency domain acoustic wave equation, the equation (3) can be discretized. The grid nodes are sorted row by row, and the corresponding node numbers correspond to: $k=n_x\cdot(j-1)+i$. Then, the wavefield and its gradient term at the same node are placed as $\left(\cdots,u_k,\frac{\partial u_k}{\partial x},\frac{\partial u_k}{\partial z},\cdots\right)^{\mathrm{T}}$, and the sparse block structure of the impedance matrix can be formed [7] [22] [36].

A2. Expressions for Non-Zero Sub-Block Elements of the Impedance Matrix

The specific expression of the non-zero sub-block matrix in equation (4) is

$$B_1^{(j)}=\begin{bmatrix}0 & 0 & 0\\ \frac{c_{-1}}{6t_z^2}+\frac{\imath d_z' h a_1}{4\omega t_z^3} & \frac{hd_{-1}}{6t_z^2}+\frac{\imath d_z' h^2 b_{-1}}{4\omega t_z^3} & \frac{hd_{-1}}{6t_z^2}+\frac{\imath d_z' h^2 b_{-1}}{4\omega t_z^3}\\ \frac{c_{-1}}{6t_x^2}+\frac{\imath d_x' h a_1}{4\omega t_x^3} & \frac{hd_{-1}}{6t_x^2}+\frac{\imath d_x' h^2 b_{-1}}{4\omega t_x^3} & \frac{hd_{-1}}{6t_x^2}+\frac{\imath d_x' h^2 b_{-1}}{4\omega t_x^3}\end{bmatrix}$$

$$B_2^{(j)}=\begin{bmatrix} \dfrac{a_{-1}}{t_z^2} & 0 & \dfrac{hb_{-1}}{t_z^2} \\ 0 & 0 & 0 \\ \dfrac{c_{-1}}{t_z^2}-\dfrac{c_{-1}}{3t_x^2}+\dfrac{3\imath d_z' h a_{-1}}{\omega t_z^3} & 0 & \dfrac{hd_{-1}}{t_z^2}-\dfrac{hd_{-1}}{3t_x^2}+\dfrac{3\imath d_z' h^2 b_{-1}}{\omega t_z^3} \end{bmatrix}$$

$$B_3^{(j)}=\begin{bmatrix} 0 & 0 & 0 \\ -\dfrac{c_{-1}}{6t_z^2}-\dfrac{\imath d_z' h a_{-1}}{4\omega t_z^3} & \dfrac{hd_{-1}}{6t_z^2}+\dfrac{\imath d_z' h^2 b_{-1}}{4\omega t_z^3} & -\dfrac{hd_{-1}}{6t_z^2}-\dfrac{\imath d_z' h^2 b_{-1}}{4\omega t_z^3} \\ \dfrac{c_{-1}}{6t_x^2}-\dfrac{\imath d_x' h a_{-1}}{4\omega t_x^3} & -\dfrac{hd_{-1}}{6t_x^2}+\dfrac{\imath d_x' h^2 b_{-1}}{4\omega t_x^3} & \dfrac{hd_{-1}}{6t_x^2}-\dfrac{\imath d_x' h^2 b_{-1}}{4\omega t_x^3} \end{bmatrix}$$

$$B_4^{(j)}=\begin{bmatrix} \dfrac{a_{-1}}{t_x^2} & \dfrac{hb_{-1}}{t_x^2} & 0 \\ \dfrac{c_{-1}}{t_x^2}-\dfrac{c_{-1}}{3t_z^2}+\dfrac{3\imath d_x' h a_{-1}}{\omega t_x^3} & \dfrac{hd_{-1}}{t_x^2}-\dfrac{hd_{-1}}{3t_z^2}+\dfrac{3\imath d_x' h^2 b_{-1}}{\omega t_x^3} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$B_5^{(j)}=\begin{bmatrix} \dfrac{\omega^2 h^2}{c^2}-\left(\dfrac{a_0}{t_x^2}+\dfrac{a_0}{t_z^2}\right) & \dfrac{b_0 h}{t_x^2}+\dfrac{\imath d_x' h^2}{\omega t_x^3} & \dfrac{b_0 h}{t_z^2}+\dfrac{\imath d_z' h^2}{\omega t_z^3} \\ \dfrac{c_0}{t_x^2}-\dfrac{c_0}{3t_z^2}+\dfrac{3\imath d_x' h a_0}{\omega t_x^3} & \dfrac{hd_0}{t_x^2}-\dfrac{3d_x'^2 h^3}{\omega^2 t_x^4}+\dfrac{\imath d_x'' h^3}{\omega t_x^3}-\dfrac{\omega^2 h^3}{c^2} & 0 \\ \dfrac{c_0}{t_z^2}+\dfrac{3\imath d_z' h a_0}{\omega t_z^3} & 0 & \dfrac{hd_0}{t_z^2}-\dfrac{3d_z'^2 h^3}{\omega^2 t_z^4}+\dfrac{\imath d_z'' h^3}{\omega t_z^3}+\dfrac{\omega^2 h^3}{c^2} \end{bmatrix}$$

$$B_6^{(j)}=\begin{bmatrix} \dfrac{a_1}{t_x^2} & \dfrac{hb_1}{t_x^2} & 0 \\ \dfrac{c_1}{t_x^2}-\dfrac{c_1}{3t_z^2}+\dfrac{3\imath d_x' h a_1}{\omega t_x^3} & \dfrac{hd_1}{t_x^2}-\dfrac{hd_1}{3t_z^2}+\dfrac{3\imath d_x' h^2 b_1}{\omega t_x^3} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$B_7^{(j)}=\begin{bmatrix} 0 & 0 & 0 \\ -\dfrac{c_1}{6t_z^2}-\dfrac{\imath d_z' h a_1}{4\omega t_z^3} & \dfrac{hd_1}{6t_z^2}+\dfrac{\imath d_z' h^2 b_1}{4\omega t_z^3} & -\dfrac{hd_1}{6t_z^2}-\dfrac{\imath d_z' h^2 b_1}{4\omega t_z^3} \\ \dfrac{c_1}{6t_x^2}-\dfrac{\imath d_x' h a_1}{4\omega t_x^3} & -\dfrac{hd_1}{6t_x^2}+\dfrac{\imath d_x' h^2 b_1}{4\omega t_x^3} & \dfrac{hd_1}{6t_x^2}-\dfrac{\imath d_x' h^2 b_1}{4\omega t_x^3} \end{bmatrix}$$

$$B_8^{(j)}=\begin{bmatrix} \dfrac{a_1}{t_z^2} & 0 & \dfrac{hb_1}{t_z^2} \\ 0 & 0 & 0 \\ \dfrac{c_1}{t_z^2}-\dfrac{c_1}{3t_x^2}+\dfrac{3\imath d_z' h a_1}{\omega t_z^3} & 0 & \dfrac{hd_1}{t_z^2}-\dfrac{hd_1}{3t_x^2}+\dfrac{3\imath d_z' h^2 b_1}{\omega t_z^3} \end{bmatrix}$$

$$B_9^{(j)}=\begin{bmatrix} 0 & 0 & 0 \\ \dfrac{c_1}{6t_z^2}-\dfrac{\imath d_z' h a_1}{4\omega t_z^3} & \dfrac{hd_1}{6t_z^2}+\dfrac{\imath d_z' h^2 b_1}{4\omega t_z^3} & \dfrac{hd_1}{6t_z^2}-\dfrac{\imath d_z' h^2 b_1}{4\omega t_z^3} \\ \dfrac{c_1}{6t_x^2}+\dfrac{\imath d_x' h a_1}{4\omega t_x^3} & \dfrac{hd_1}{6t_x^2}+\dfrac{\imath d_x' h^2 b_1}{4\omega t_x^3} & \dfrac{hd_1}{6t_x^2}-\dfrac{\imath d_x' h^2 b_1}{4\omega t_x^3} \end{bmatrix}$$