Solving Seismic Wave Equations on Variable Velocity Models With Fourier Neural Operator

Bian Li, Hanchen Wang, Shihang Feng, Xiu Yang, and Youzuo Lin

Abstract—In the study of subsurface seismic imaging, solving the acoustic wave equation is a pivotal component in existing models. The advancement of deep learning (DL) enables solving partial differential equations (PDEs), including wave equations, by applying neural networks to identify the mapping between the inputs and the solution. This approach can be faster than traditional numerical methods when numerous instances are to be solved. Previous works that concentrate on solving the wave equation by neural networks consider either a single velocity model or multiple simple velocity models, which is restricted in practice. Instead, inspired by the idea of operator learning, this work leverages the Fourier neural operator (FNO) to effectively learn the frequency domain seismic wavefields under the context of variable velocity models. We also propose a new framework paralleled FNO (PFNO) for efficiently training the FNO-based solver given multiple source locations and frequencies. Numerical experiments demonstrate the high accuracy of both FNO and PFNO with complicated velocity models in the OpenFWI datasets. Furthermore, the cross-dataset generalization test verifies that PFNO adapts to out-of-distribution velocity models. Finally, PFNO admits higher computational efficiency on large-scale testing datasets than the traditional finite-difference method. The aforementioned advantages endow the FNO-based solver with the potential to build powerful models for research on seismic waves.

Index Terms—Fourier neural operator (FNO), Helmholtz equation, operator learning, seismic wave.

I. INTRODUCTION

The last decade has witnessed the fast development of deep learning (DL), thanks to the enormous growth in computational power and increasingly ample data that are accessible in many fields. According to the universal approximation theorem [1], [2], [3], neural networks are capable of approximating any nonlinear function satisfying certain smoothness conditions. Therefore, their high expressiveness allows them to capture highly sophisticated latent patterns from data. Furthermore, neural architectures can be exploited to learn the mappings between functions in the infinite-dimensional space. As shown in the work [4], [5], neural networks can effectively approximate continuous nonlinear operators. This universal approximation theorem for operators leads to the development of DeepONet [6]. Under the dome of operator learning, another string of research defines the Fourier neural operator (FNO) by a series of integral operators and activation functions [7], [8], [9], [10].

The solution of a partial differential equation (PDE) is a type of complicated mapping that usually does not have an explicit form. In this work, we intend to solve the acoustic wave equation as wave equation forward modeling is the engine of many fields of study that are related to wave phenomena. We take the full waveform inversion (FWI) as an example, which is a powerful geophysical technique to reconstruct high-resolution subsurface seismic velocity models from seismic records [11]. It has been proven in the past decade that FWI is capable of meeting most of the industrial production demands [12]. However, the core of FWI involves numerically solving second-order PDEs (wave equations) twice in one iteration to calculate the gradient regarding the velocity update. The wavefield modeling step in FWI has a computational complexity of $O(N^3)$ for 2-D cases and $O(N^4)$ for 3-D cases. Usually, hundreds of such iterations would be needed before the algorithm converges to a satisfactory velocity model. Thus, a successful FWI implementation induces tremendous computational costs in real-data implementations, and such costs will even dramatically increase when dealing with large-scale datasets, such as the passive seismic inversion problems when a large number of sources are involved [13], [14]. As more and more seismic inversion frameworks [15], [16] perform forward modeling and the inversion concurrently, it is increasingly crucial to solving the forward problem accurately and efficiently.

It has been an emerging direction in which deep neural networks are applied to efficiently solve PDEs [17], [18], [19], [20]. This is motivated by numerous cases where the PDE admits no straightforward numerical methods...
or where traditional solvers are computationally expensive. Under such a context, the solution to the PDE, which is a function of time and location, is approximated by neural networks. PDE-constrained frameworks \[21\], \[22\], \[23\], \[24\] impose the governing equation(s) as a constraint in the training since purely data-driven models may yield solutions that violate the governing equation. These frameworks require that the form of the PDE is explicitly known. As mentioned above, the wave equation is of particular interest as it governs the behavior of seismic waves. As an instance of neural-network-based framework, the physics-informed neural network (PINN) was applied to solve the wave equation in the time domain \[25\], \[26\]. Most importantly, the frameworks above only solve the wave equation on a fixed velocity model. This is an obvious limitation since abundant velocity models are present in reality, and we expect a model to solve the equation given different velocity models. Another direction aims to embed the PDE information into the network architecture. For example, Sun et al. \[16\], \[27\] harness a recurrent neural network (RNN) structure based on the finite difference of the wave equation. Particularly, the sequence of interest is formed by the wavefield values that are consecutive in time so that the time evolution of the wave directly depends on preceding states. However, the PDE information is not strictly imposed during the training. Consequently, it remains an open question whether the model can generalize to new datasets. In addition, this framework relies on a mesh because it is built on a finite-difference scheme.

Aiming at building a mesh-independent solver \[10\] incorporating variable velocity models, we hereby turn to operator-learning-based neural architectures for parametric PDEs. More specifically, the wave equation is considered a parametric PDE which takes the velocity as a variable parameter, and our goal is to map different velocity models to their corresponding wavefields. Notably, this work solves the wave equation in the frequency domain. Due to the scattering effects caused by complex subsurface structures (waves reflected by the complexity of the velocity model) and the causal relationship between time steps, the wavefields in the time domain can be extremely sophisticated. In comparison, the frequency-domain wavefields exhibit benign properties. Specifically, the wavefields induced by different velocity models share common structures: they are composed of layers of “circles” centered at the seismic source position. The deformed circles and other irregularities in a wavefield directly correspond to the features of the velocity model. In the frequency domain, the wave propagation is governed by the Helmholtz equation, which is the Fourier-transformed wave equation along the time axis. The Helmholtz equation has been solved with PINN in a series of papers \[28\], \[29\], \[30\]. In addition, the U-Net \[31\] is applied in \[32\] to perform fast wavefield interpolation of different frequencies. However, they all only solved the equation on a single velocity model.

Inspired by the seminal work \[7\], we apply the FNO to learn the wavefields based on variable velocity models. There are previous works that attempt to solve the Helmholtz equation using FNO \[33\], \[34\], but these frameworks focus on other variable parameters, e.g., source location and frequency, rather than velocity. The most related work to ours is \[35\]. This work applies FNO to perform time evolution of the wave equation based on multiple velocity models generated by the Gaussian random field, which may not yield geologically realistic features such as layered structures and fault zones. In addition, the relatively low-frequency source wavelet has little sensitivity to the high-wavenumber random scatters of the velocity models in their experiments. Hence, scattered events do not appear in the corresponding time-domain wavefields under such a configuration, making the wavefield smooth and easier to predict than reality. Our work not only adapts FNO to approaching multiple realistic velocity models but also extends FNO so that it is capable of solving the Helmholtz equation in a multisource-location and multifrequency setting. To this end, we propose the paralleled FNO (PFNO), where a group of FNOs process data of different frequencies in parallel. This structure simplifies the mapping by avoiding taking the frequency as another variable parameter. The resulting wavefields predicted by PFNO capture irregularities in high-frequency data. In other words, PFNO preserves prediction accuracy while incorporating multiple sources and frequencies.

To validate our models, we implemented comprehensive numerical tests over six datasets from three families in the OpenFWI datasets \[36\]. The first objective is to verify whether the FNO-based solvers make high-accuracy predictions on the wavefields. In the single source location and single frequency setting, an FNO provides satisfactory results from low-frequency to high-frequency data. However, it is not as impressive when multiple sources and frequencies are involved. In addition, a ForwardNet, which is based on an encoder–decoder architecture, is included in the comparison. In this case, PFNO outperforms FNO and ForwardNet in terms of both the mean-squared error (mse) and the wavefield visualization. Generalization is another important factor for model evaluation. To test the generalization of PFNO, we train a PFNO for each of the six datasets and test its performance on the other five ones. This cross-dataset generalization test shows that PFNO is able to generalize to unseen velocity models that share common features with the velocity models in the training set. In addition, we quantify the relationship between PFNO performance and FNO structure as well as velocity model complexity. The last part of the numerical experiments addresses the computational efficiency of PFNO. We show that PFNO has an evident speed advantage over the “optimal” 9-point finite difference method \[37\], a standard traditional numerical method for the Helmholtz equation, in the case of large-scale inferences.

This article is organized as follows. We first give some background on the Helmholtz equation and FNO in Section II. Section III follows to describe the FNO-based solver of the Helmholtz equation with variable velocity. Then, the performance of the models is discussed in detail in Section IV. Finally, we arrive at the discussion and the conclusion in Sections V and VI, respectively.
II. BACKGROUND

A. Helmholtz Equation

Prior to an in-depth discussion of the proposed method, we first briefly introduce the basics of the acoustic wave equation. Let \((x, z) \in \mathbb{R} \times \mathbb{R}\) denote a point in a 2-D spatial domain. Specifically, \(x\) is the horizontal distance, and \(z\) is the depth. Then, the 2-D acoustic wave equation with a constant density is

\[
\nabla^2 p(t, x, z) - \frac{1}{v(x, z)^2} \frac{\partial^2 p(t, x, z)}{\partial t^2} = s(t, x, z) \quad (1)
\]

where \(p\) denotes the acoustic pressure wavefield, \(v(x, z)\) is the velocity value at location \((x, z)\), and \(s\) is the point source at location \((x_s, z_s)\). The symbol \(\nabla^2 := (\partial^2/\partial x^2) + (\partial^2/\partial z^2)\) is the Laplacian. Equation (1) describes the spatiotemporal pattern of how seismic energy spreads out from the source. However, it is challenging to learn the time-domain wavefields, given complicated velocity models. Particularly, the subsurface structures reflected by the velocity models cause highly nonlinear scattering effects, making the wavefields exceedingly complicated. Moreover, the temporal derivative induces the dependence between snapshots of the wavefield at different time steps, which hinders the training of a neural network. Therefore, we focus on solving the Helmholtz equation

\[
\nabla^2 u(x, z; \omega) + k^2 u(x, z; \omega) = f(x, z; \omega) \quad (2)
\]

where \(\omega\) is the angular frequency, and \(k = (\omega/v)\) is the wavenumber. The quantity of interest is the frequency-domain pressure wavefield \(u\). This equation can be viewed as the result of applying the Fourier transform to the time component while preserving the spatial information. Given a frequency \(\omega\), the wavefield does not interact with wavefields of other frequencies as there is no “frequency derivative” in the equation. Therefore, wavefields of different frequencies are considered independent of each other, which allows us to train a neural network on these wavefields in parallel.

B. Fourier Neural Operator

The idea of the FNO stems from Green’s function of the linear differential operator \(L\). Consider a linear PDE \(Lu = f\), where \(u\) is the solution to the PDE, and \(f\) is the external force. The corresponding Green’s function \(G(\cdot, \cdot)\) then gives the solution as an integral

\[
u(x) = \int_{\mathcal{D}} G(x, y)f(y)dy \quad (3)
\]

where \(x, y \in \mathcal{D} \subset \mathbb{R}^d\). If there exists a function \(g\) such that \(G(x, y) = g(x - y)\), (3) reduces to a convolution operation, and \(G\) serves as the convolution kernel. For a parametric PDE, Green’s function also depends on the parameter. For example, the velocity \(v\) is a parameter in the wave equation or the Helmholtz equation. In this framework, the parameter is assumed to be a function of \(x\). According to [7], the kernel integral operator is defined by

\[
(K_\theta(h_i))(x) = \int_{\mathcal{D}} \kappa_\theta(x, y, v(x), v(y)) h_i(y)dy \quad (4)
\]

where \(v\) is the PDE parameter, and the kernel integral operator \(K_\theta\) is parameterized by a neural architecture with learnable parameters \(\theta\). On the right-hand side, \(\kappa\) approximates the (convolution) kernel \(G\). In particular, \(h_i\) denotes the latent representation generated by a Fourier layer in a sequence of Fourier layers indexed by \(i\).

To better understand the structure of a Fourier layer, we visualize it in Fig. 1. The flow of \(h_i(x)\) bifurcates into two paths. The upper path approaches the integral in (4) by defining the kernel integral operator. Unlike [8] in which message passing is used as in a graph neural network, FNO leverages the convolution theorem which states convolution is equivalent to pointwise multiplication in the Fourier space. Hereby, FNO initializes complex-valued learnable parameters directly in the Fourier space. Following the notations in [7], we use \(R_\theta\) to represent the Fourier transform of \(\kappa\), i.e., \(R_\theta = \mathcal{F}(\kappa_\theta)\). Then, the Fourier convolution operator is defined by

\[
K_\theta(h_i) = \mathcal{F}^{-1}(\mathcal{F}(K_\theta)\mathcal{F}(h_i)) = \mathcal{F}^{-1}(R_\theta \mathcal{F}(h_i)) \quad (5)
\]

whose output is real-valued in the latent space. In the implementation, the fast Fourier transform (FFT) and its inverse are used to preserve computational efficiency. In the lower path, we apply a linear transform \(W\) to \(h_i(x)\), wherein the discretized \(W\) is a matrix of learnable parameters. The results of the two paths are then added together, followed by a nonlinear activation function \(\sigma\).

With the Fourier layer defined, Fig. 2 illustrates the architecture of FNO. FNO incorporates a lifting layer \(P\) which lifts the input tensor into high-dimensional space of width \(w\). A sequence of Fourier layers follow to update the latent representations by

\[
h_i = \sigma(W_1 h_{i-1}(x) + (K_\theta(h_{i-1})(x))) \quad (6)
\]

with \(i = 1, 2, \ldots, N\). Since \(R_\theta, W\) differ in each layer, they are indexed by \(i\) as well. Finally, layer \(Q\) projects \(h_N\) to the solution function space. In our case, the output is the wavefield. Notably, FNO consists of a sequence of nonlocal kernel integral operators combined with the activation function \(\sigma\) so that it is capable of approximating highly nonlinear operators. In this sense, it generalizes Green’s function.

As FNO maps the parameter \(v\) to the solution \(u\), it succeeds in solving PDEs with variable parameters. For comparison, frameworks such as [21] only solve one instance of the PDE after fixing the parameter. Another advantage of FNO compared with [27] is it is mesh-independent. When the resolution of the discretization changes, FNO generalizes to unseen locations. Our work leverages FNO to map velocity models
to the corresponding wavefields, respecting the Helmholtz equation in (2).

III. Method

Given multiple velocity models, we have $N$ data samples denoted by $\{V^i, u^i\}^N_{i=1}$, where $V^i \in \mathbb{R}^{n_z \times n_x}$ is a (discretized) velocity model sample on an $n_z \times n_x$ grid. Of note, the location is represented by $x = (x, z)$. In particular, $x$ and $z$ denote the horizontal offset and the depth, respectively. Label $u^i \in \mathbb{C}^{n_z \times n_x}$ is the ground-truth pressure wavefield in the frequency domain. Our goal is to solve the Helmholtz equation numerically by learning the wavefield with FNO. For convenience, we denote the shape of a data sample by $[n_z, n_x]$. Since FNO incorporates the coordinates into its input, a complete input is a 3-D tensor of the shape $[n_z, n_x, 3]$, as illustrated in Fig. 3. The last dimension contains $v, x, z$ values, making it an analog to the dimension of channels as in image processing. Specifically, we have three “channels” here.

The forward modeling is formulated by the following optimization problem:

$$\min_{\tilde{\phi} \in \phi} \frac{1}{N} \sum_{k=1}^{N} \| \tilde{u}^i_k - u^k \|_2^2$$  \hspace{1cm} (7)

where $\tilde{u}^i_k = \text{FNO}(T^k; \phi)$. Here, $\phi = \{R_{\theta}, W, P, Q\}$, $R_{\theta} = \{R_0\}$, $W = \{W_i\}$, with $i$ indexing the Fourier layers, and $T^k = V^k \oplus X^k \oplus Z^k$ in which $\oplus$ denotes concatenation.

In real-life applications, multiple sources at different locations are usually required. Seismic data simulated by multiple sources have a better representation (or illumination) of the velocity model compared with the single source case because different source locations provide various incident waves that illuminate the same spatial location. Multifrequency simulation is essential as well. Lower frequency components in the data are only sensitive to the longer wavelength parts of the velocity model, which represents the smoothed background. In comparison, higher frequencies are more sensitive to the shorter wavelengths that represent the sharp scatters in the velocity model. Thus, the model should be capable of solving the Helmholtz equation with multiple source locations and multiple seismic frequencies to have an accurate seismic data representation of the velocity model. Accordingly, FNO incorporates $x, \omega$ as another two variable parameters. Regarding the source location, we use binary encoding: the source locations are encoded by 1, whereas all other locations are 0. On the other hand, the frequency is set as a constant across the spatial domain since it relies on neither the coordinates nor the velocity. As a result, a data sample ends up in the shape $[n_z, n_x, n_z, n_x, 5]$. A naive way of training is to feed these 5-D tensors into a single FNO. Nevertheless, different source locations and frequencies significantly increase the complexity of the mapping between the inputs and the wavefields. To partially counteract the increasing mapping complexity, we hereby propose the PFNO, a structure in which we stack multiple FNOs in parallel and let each of them specialize in only one frequency. In other words, each FNO only takes two variable parameters, the source location and the velocity, instead of three. Fig. 4 depicts the data flow through PFNO.

IV. Numerical Experiments

A. Datasets

In this section, we demonstrate the performance of FNO and PFNO using six datasets from the dataset collection OpenFWI, namely, CurveFault-A & CurveFault-B, Style-A & Style-B, and FlatVel-A & FlatVel-B. More specifically, CurveFault-A and CurveFault-B contain discontinuities (geological faults) in the velocity models. These faults are induced by shifting the rock layers. Of note, CurveFault-B exhibits more complicated changes in the velocities than CurveFault-A. In contrast, FlatVel-A and FlatVel-B have flat interfaces without faults, but the velocity values are randomly distributed across different depths in FlatVel-B, which increases the complexity. The velocity models in Style-A and Style-B are generated with style transfer, where the Marmousi model [38] serves as the style image while the content images are drawn from the COCO dataset [39]. The difference between them is that the velocity models in Style-A are smoother. In the above datasets, the spatial domain is defined on a $70 \times 70$ grid with $\Delta x = \Delta z = 10$ m, and all the velocity values fall in the interval $[1500 \text{ m/s}, 4500 \text{ m/s}]$. Fig. 5 displays examples from each of the datasets.

To access the ground-truth wavefields (labels) for the training, a two-step process is adopted. Based on the velocity models, we first solve (1) by a high-order finite difference
scheme [15] to obtain the time-domain wavefields, the details of which are included in Appendix A. Then, we apply the Fourier transform applied to the $t$ dimension and convert the time-domain wavefields into the frequency-domain wavefields which will be the labels. Notably, the frequency-domain wavefields are complex-valued. Hence, in our implementation, the complex numbers are split into the real and imaginary parts. The training loss is then obtained by averaging the losses from these two parts.

As mentioned in Section III, we intend to incorporate multiple source locations and frequencies. Particularly, the datasets have five sources at a depth of 10 m below the surface. Horizontally, they are distributed evenly from 0 to 690 m. On the other hand, the selection of the frequencies is slightly more sophisticated. With an FFT from the time domain, there are 1000 frequencies corresponding to the 1000 time steps in the finite difference. However, it is computationally prohibitive to learn the wavefields of all the frequencies. Hence, a common technique in the field of signal processing is adopted in this work. Specifically, we attend to the magnitude of the wavefields since it reflects the energy level of the frequency domain signal. Then, we only focus on the spectrum that contributes to the majority of the total energy. Based on CurveFault-A, we plot the magnitude against the frequencies in Fig. 6. Over 90% of the energy is contributed by the signals of the spectrum 1–30 Hz. Of note, the magnitude has an almost identical distribution over frequencies in other datasets. Therefore, this range is chosen to be the spectrum of interest.
Therefore, we propose the PFNO structure where multiple FNOs are paralleled. Each of these FNOs only handles data of a single frequency (but still of multiple source locations). Hence, the source location and the velocity are the variable parameters for each FNO. In the numerical experiments, the datasets have the aforementioned five source locations and the following ten frequencies: \{1, 3, 5, 7, 9, 12, 15, 19, 25, 30\} Hz. We choose smaller intervals for lower frequencies and larger intervals for higher ones in the experiments mainly to solve seismic imaging problems such as FWI, where the lower frequency components are more important than the higher ones. It is because the lower frequencies represent the fundamental low-wavenumber part of subsurface kinematics, and lack of such information leads to the higher frequencies cycle-skipping problems [41].

Figs. 8–10 visualize the performance of PFNO on the CurveFault datasets, Style datasets, and FlatVel datasets, respectively. For demonstration purposes, only the real part of the 25-Hz wavefields is shown since the imaginary part is similar. As depicted in Fig. 8, the geological faults induce a number of scattering artifacts that form the cobwebs in some local areas. Given smooth features in the velocity models, Fig. 9(a) exhibits smooth curves in the wavefields as expected. In comparison, ruffled layers manifest in Fig. 9(b). The irregularities in Fig. 10 reveal the reflections of energy between layer interfaces, although these interfaces are flat. Since Style and FlatVel have relatively simpler wavefields, PFNO performs better on these two families than on CurveFault. Notably, the resolution of the wavefield at a certain point is inverse-proportional to its wavelength and the velocity value at that point if we fix the frequency. Thus, the blurry parts are caused by the small wavenumber at the current frequency. In addition, we also visualize the time-domain wavefields reconstructed by inverse Fourier transform in Appendix B.

As a comparison, we also implement a single FNO of width 96 that takes source location, frequency, and velocity into its inputs. In addition, an encoder–decoder architecture is introduced as an image-to-image method. This architecture is inspired by InversionNet [42]. With the purpose of solving the forward problem, an encoder of 11 convolutional layers maps the velocity model to a latent space, and then a decoder also maps the wavefield. For convenience, we refer to this architecture as ForwardNet. The specific structure and hyperparameters of ForwardNet can be found in Appendix C. Similarly, we examine the performance of FNO and ForwardNet on the six datasets and compare them with PFNO. Here, the mse is used to measure how PFNO performs on the test set. As Table I shows, PFNO consistently outperforms the other two models on all the datasets. To provide an intuitive comparison, Fig. 11 visualizes the predicted wavefields and the ground truth using a 25-Hz example from CurveFault-A. Highly agitated patterns of the layer edges are recognized by PFNO, whereas FNO and ForwardNet only capture the smooth part. Moreover, the wavefields predicted by FNO turn blurry as the wave spreads rightward, but PFNO still gives sharp scatterings, as the ground truth suggests.
C. Generalization

The model’s ability to generalize to other datasets is of importance. If a trained model only approximates certain patterns within a dataset instead of the intrinsic properties of the Helmholtz operator, it needs to be retrained once the features of the velocity model change, which offsets the benefit of fast inference. To examine the generalization of PFNO, we hereby perform a cross-dataset generalization test. More specifically, we test the performance of each trained PFNO in Section IV-B on the other five datasets. In Table II, the diagonal marks the intradataset test results. As expected, the smallest error for each test set appears on the diagonal. Since CurveFault-A and CurveFault-B share similar features in their velocity models, PFNO trained on either one generalizes to the other. Furthermore, PFNOs trained on them can adapt to all other datasets except FlatVel-B because the CurveFault family admits features seen in the other two families, such as curves and layered structures. Interestingly, the Style-A-trained PFNO has a mild generalization to CurveFault-A, but it fails to generalize to CurveFault-B. In addition, it can generalize to Style-B despite sharper features in it. In contrast, its Style-B-trained counterpart only adapts to Style-A. This is
presumably caused by the unique sharp features in Style-B. Among the three families, FlatVel seems to be biased to the other two distributions because it contains only flat layers and no lateral variation. Thus, PFNOs trained on them are unable to generalize to the others. Although the FlatVel-B-trained PFNO makes accurate predictions on FlatVel-A, the opposite direction does not apply. The reason is likely rooted in the randomly distributed velocities across layers in FlatVel-B.

Another pivotal generalization test is to examine how FNO/PFNO adapts to a smooth velocity model. The motivation lies in the fact that the FWI methods usually start with a smooth velocity model. By iterative updates, these methods gradually find sharper features. Therefore, the FNO-based solvers must be able to accurately predict the wavefields in the early stage if they are to be applied as the forward modeling module in an FWI method.

With the PFNO trained on CurveFault-A, we test its performance on velocity models that are smoothed by Gaussian filters. Particularly, the degree of smoothness is controlled by the standard deviation of the Gaussian kernel. As we smooth the velocity model, it becomes less similar to the training velocity models, and the neural network has never seen such smoothed velocity models in the training stage, which is usually described as out-of-distribution. Meanwhile, the smoothness leads to simpler wavefields as the high-wavenumber part of the velocity models has been smoothed out. The generalization is subject to the combination of these two factors. Considering all the source locations and frequencies, we compute the average mse over 3000 velocity models that are smoothed by the Gaussian filter with different standard deviations. The result is shown in Fig. 12. This experiment verifies that PFNO does generalize to smooth velocity models overall.

Although this work focuses on whether our FNO-based solvers generalize on velocity, it is also worth exploring their generalization on frequency. The detail can be found in Appendix D.

**D. Effect of FNO Complexity**

To better train PFNO, we need to understand how FNO’s complexity affects its performance. The complexity depends on the number of learnable parameters in the FNO structure. In our experiments, the hyperparameter \( w \) controls the dimensions and hence the number of learnable parameters. More specifically, it specifies the dimension of \( R_\theta \) in Fig. 1 as well as \( P, Q \) in Fig. 2. Recall that lifting layer \( P \in \mathbb{R}^{5 \times w} \) projects the tensors to the dimension \([n_b, n_s, n_w, n_z, n_x, w]\), where \( n_b, n_s, n_w, n_z, n_x \) stand for the batch size, number of source locations, number of frequencies, number of \( z \)-coordinates, and number of \( x \)-coordinates, respectively. They are then reshaped as \([n_b \cdot n_s \cdot n_w, w, n_z, n_x]\) for FFT. Accordingly, \( R_\theta \in \mathbb{C}^{w \times w \times n_z \times n_x} \) is defined in the Fourier space, while \( W \in \mathbb{R}^{w \times w} \) exists in the original space.


The dimension of projection \( Q \in \mathbb{R}^{w \times 2} \) corresponds to the real and imaginary parts of the wavefield. Thus, \( w \) linearly scales \( P, Q \) and quadratically scales \( R_\theta, W \). To this end, we target on different \( w \) and perform a set of tests to further examine its influence. The tests include three datasets so that we incorporate different velocity model interactions. Table III shows that a larger \( w \) leads to higher accuracy over all three types. On the other hand, the complexity of the wavefields can be partially measured by the scatters between wavenumbers. FNO has a better performance on the wavefields with simpler interactions between wavenumbers. This can be verified by the decreasing \( \text{mse} \) in each row.

**E. Effect of Velocity Model Complexity**

Although the complexity of velocity models has been mentioned, we have not analyzed its impact on the performance of PFNO. Hereby, we perform an experiment to quantify the relationship between velocity model complexity and prediction misfit (absolute difference between the prediction and the ground truth). The Shannon entropy is applied to measure the velocity model complexity [36]. In particular, we slide a window of size \( 20 \times 20 \) across the velocity model. In each step, we compute the regional entropy. Similarly, the regional average misfit is obtained. The results are two \( 50 \times 50 \) matrices shown in Fig. 13(c) and (d). In Fig. 13(c), the change in regional entropy is observable only near the boundaries because the velocity values are almost the same inside a layer. Since our goal is to investigate how the complexity relates to the misfit, we focus on the high-misfit area marked by the vertical dashed lines Fig. 13(d). We visualize their relationship in Fig. 13(e). There is an evident negative correlation between the two. As the complexity rises, more information is introduced, leading to a decrease in entropy. The relationship is consistent with our hypothesis that higher complexity causes larger misfit.

**F. Computational Efficiency**

One major motivation for applying deep learning (DL) models over the traditional methods is their fast inference. In the case of a large-scale dataset, i.e., tens of thousands of velocity models, it is time-consuming to solve the Helmholtz equation with the traditional methods. In comparison, FNO and PFNO predict a wavefield within several milliseconds on average. Nevertheless, the apparent drawback is that they demand hours for training which is the typical off-line efforts for DL-based scientific computing like PINN and DeepONet. Therefore, we take both the training and inference time into consideration when comparing the efficiency of FNO and PFNO with a traditional method, the “optimal” 9-point finite difference method [37].

The comparison is performed in two settings: 1) single source location and single frequency and 2) five source
locations and ten frequencies. In the first case, an FNO is trained with one GPU, while we exploit four GPUs to train a PFNO in the second case. However, both the cases use only one GPU for inference. Again, there are 15,000 velocity models for training and 3000 for inference. In the inference, the batch size of both FNO and PFNO is set to 64.

Not surprisingly, FNO outputs a wavefield in $4.23 \times 10^{-4}$ s on average in case 1. On the other hand, the “optimal” 9-point method needs about $2.30 \times 10^{-2}$ s to solve the equation based on a velocity model. The speed advantage of PFNO is more significant with multiple sources and frequencies. For the multisource and multifrequency experiment setup on the same velocity model, PFNO averages $9.63 \times 10^{-3}$ s to predict the wavefield, while in contrast, the “optimal” 9-point method takes $145.65$ s on average. Considering that FNO and PFNO took $2.33$ and $12.02$ h for training, respectively, we consolidate them with their inference time, which makes the comparison more reasonable. As Fig. 14(a) illustrates, FNO has a negligible marginal increase in the wall time as the number of velocity models increases, leading to an almost flat line. Moreover, accounting for training time, FNO is more efficient if more than 3648 velocity models are to be approached, even though the training time is incorporated. A similar conclusion can be drawn from Fig. 14(b), only that PFNO is even more preferable. Provided that we have more than 298 velocity models in the dataset, PFNO prevails over the traditional method in terms of speed. From the evidence above, one may verify the efficiency of FNO and PFNO in large-scale inferences.

Furthermore, in practice, the domain size also has an impact on the inference speed. By varying the domain size, we show that FNO dominates the “optimal” 9-point method regarding the inference speed when $n_z = n_x \geq 200$. Unlike the multivelocity model setting above, this comparison only focuses on the one velocity model, generated by interpolating an instance in CurveFault-A. Fig. 15 reveals that the time of the “optimal” 9-point method soars up with the increasing domain size. Although the theoretical time complexity of the method is $O(n^3)$, the sparsity and the special structure
Fig. 11. Model performance comparison. (a) FNO. (b) ForwardNet. (c) PFNO. (d) Ground truth.

### TABLE II

**CROSS-DATASET GENERALIZATION: THE ROWS ARE THE TRAINING SETS AND THE COLUMNS ARE THE TEST SETS; ERRORS ARE MEASURED BY MSE**

|          | CurveFault-A | CurveFault-B | Style-A | Style-B | FlatVel-A | FlatVel-B |
|----------|--------------|--------------|---------|---------|-----------|-----------|
| CurveFault-A | 3.357e-05    | 8.873e-04    | 2.067e-04 | 1.269e-04 | 8.963e-05 | 3.451e-03 |
| CurveFault-B | 1.070e-04    | 1.959e-04    | 6.748e-05 | 4.835e-05 | 8.869e-05 | 2.034e-03 |
| Style-A    | 8.221e-04    | 1.835e-03    | 1.713e-05 | 2.834e-05 | 2.851e-03 | 6.607e-03 |
| Style-B    | 1.232e-02    | 2.768e-02    | 2.294e-04 | 8.478e-06 | 2.268e-02 | 3.550e-01 |
| FlatVel-A  | 3.040e-03    | 4.793e-03    | 2.377e-03 | 1.070e-03 | 3.209e-06 | 1.185e-03 |
| FlatVel-B  | 3.015e-03    | 4.704e-03    | 2.382e-03 | 1.054e-03 | 1.185e-05 | 2.302e-05 |

**Fig. 12.** Generalization on smoothed velocity models: the results are based on velocity models in *CurveFault-A*; sigma denotes the standard deviation in the Gaussian kernel.

of the impedance matrix yield an almost quadratic complexity in practice. In contrast, the inference speed of the

|          | CurveFault-A | CurveFault-B | Style-A | Style-B | FlatVel-A | FlatVel-B |
|----------|--------------|--------------|---------|---------|-----------|-----------|
| CurveFault-A | 3.357e-05    | 8.873e-04    | 2.067e-04 | 1.269e-04 | 8.963e-05 | 3.451e-03 |
| CurveFault-B | 1.070e-04    | 1.959e-04    | 6.748e-05 | 4.835e-05 | 8.869e-05 | 2.034e-03 |
| Style-A    | 8.221e-04    | 1.835e-03    | 1.713e-05 | 2.834e-05 | 2.851e-03 | 6.607e-03 |
| Style-B    | 1.232e-02    | 2.768e-02    | 2.294e-04 | 8.478e-06 | 2.268e-02 | 3.550e-01 |
| FlatVel-A  | 3.040e-03    | 4.793e-03    | 2.377e-03 | 1.070e-03 | 3.209e-06 | 1.185e-03 |
| FlatVel-B  | 3.015e-03    | 4.704e-03    | 2.382e-03 | 1.054e-03 | 1.185e-05 | 2.302e-05 |

**V. FURTHER DISCUSSION**

This section remarks further discussion of (P)FNO based on both brief analysis and empirical findings in the experiments.

FNO-based solver remains almost constant under the GPU setting.
A. Nonlocal Property of FNO

We first address the necessity of the operator learning scheme in the variable velocity setting. It seems plausible to add the velocity value into the inputs of a coordinate-based neural network and directly transition from fixed velocity to variable velocity. For example, one might use the input \((x, z, v)\) for PINN [21]. However, this pointwise consideration is insufficient because the effect of velocity is nonlocal. In other words, it is possible to see different wave pressures given the same velocity value \(v\) at the same location \((x, z)\) but on different velocity models. In the wave propagation, the wave pressure is essentially influenced by velocity values in a neighborhood of \((x, z)\). Following this rationale, an operator is to be learned given the velocity models that cover the entire spatial domain, which accounts for the input \((V, X, Z)\) of FNO.

Illustrated in Fig. 1, 2-D Fourier transform on the latent representations \(h_i\) in the Fourier layers leads to the nonlocal features in the Fourier space (i.e., the wavenumber domain). This allows learnable network parameters in FNO to impact any point of the output. As a result, FNO can be thought of as a global interpolator from this perspective when it is trained successfully.

B. Potential Improvement on Generalization

PFNO is a purely data-driven framework, so its performance might downgrade in a more challenging setting, e.g., wavefields with the free surface condition (see Appendix E for details). Inspired by the physics-informed learning, imposing a PDE constraint in the loss function leads to better inference on unseen velocity models. There are two ways of computing the PDE residual. The first one is to use automatic differentiation directly, as PINN does. In particular, we can differentiate the output of PFNO with respect to the input coordinates to access the partial derivatives. The second method introduced in [43] is the functionwise differentiation. With the benefit comes the cost. The automatic differentiation causes tremendous computational costs because of the second-order derivatives, significantly slowing down the training. In addition, the PDE constraint leads to undesirable loss surface for optimization. A common practice is to use Adam optimizer for a number of epochs until the loss decreases no more, and then switch to a higher order method, e.g., L-BFGS. However, L-BFGS does not scale well. It is particularly restricted in our large datasets with multiple source locations and frequencies. We leave the exploration of functionwise differentiation in the future work.

C. Analysis of GPU Memory Requirement

PFNO had been expected to scale well to larger domain or to 3-D domain in that it should require considerably less memory than the traditional methods for the Helmholtz equation, since the impedance matrix [44] is theoretically in \(O(N^2)\) with \(N = n^3\) for a 3-D domain, assuming each dimension has \(n\) grid points. However, it turns out that GPU memory can be a bottleneck in training PFNO due to the following factors. First, the 2-D FFT and the inverse FFT (IFFT) inside FNO are memory-demanding. In our case, the input data are in the shape of \([n_b, n_x, n_y, n_z, 5]\), where 5 results from the concatenation of source location, frequency, velocity,
for $N$ Fourier layers in an FNO in practice. When $\mathbf{w}$ is large, the instantaneous GPU memory usage is high. Second, the batch data flow and the intermediate results of FFT and IFFT must be stored, occupying additional space in the GPU. Finally, gradient information is tracked for backpropagation in the training phase, which actually consumes a large proportion of the GPU memory. The factors above hinder us from scaling up PFNO at the current stage. In the implementation, we limited the batch size to 16 in our 32-GB GPUs. Of note, just the 2-D FFT and its intermediate results take up around 12-GB GPU memory when $\mathbf{w} = 96$.

VI. CONCLUSION

In this work, the FNO is applied to efficiently solve the Helmholtz equation under the context of variable velocity. Furthermore, we extend FNO to the multisource and multifrequency setting and propose the PFNO that consists of a group of FNOs. The paralleled structure distributes the data of different frequencies to different FNOs such that each FNO concentrates on a single frequency but multiple sources. PFNO achieves high accuracy even on the complicated high-frequency wavefields, while FNO has less impressive performance. In the numerical experiments on the six OpenFWI datasets, we demonstrate the accuracy of PFNO with wavefields visualization and show that it outperforms FNO and ForwardNet on all the six datasets. Moreover, a trained PFNO is able to generalize to other datasets, provided that the velocity models share similar features. In terms of generalization on frequency, our experiments show only a weak ability to generalize to unseen frequencies, which might be a limitation of the current models. We will leave the realization of generalization on frequency to future work. To understand the effect of the hyperparameter $\mathbf{w}$, we perform a group of tests over three datasets with different $\mathbf{w}$ for a comprehensive comparison, showing that a greater $\mathbf{w}$ is needed to learn more complicated wavefields. In addition, the negative relationship between the velocity model complexity and the accuracy is verified in our experiment. Finally, the “optimal” 9-point method is compared with FNO in terms of computational efficiency. FNO embraces fast inferences as expected, taking milliseconds on average to solve for the wavefield corresponding to a velocity model. Even when the training time is incorporated, PFNO still gains an edge over the “optimal” 9-point method when solving a large number of velocity models and when solving a velocity model on a large spatial domain.

With the strengths discussed above, FNO or PFNO can be a building block of an accurate and fast numerical solver for the Helmholtz equation when the model is expected to solve a large number of velocity models.

APPENDIX A

GENERATING TIME-DOMAIN WAVEFIELDS WITH FINITE DIFFERENCE

To describe the generation of the time-domain wavefields, we hereby elaborate on the high-order finite difference and other related details. Let $p_{t,x,z}^i := p(t, x, z)$. The temporal
derivative is approximated with the second-order central difference
\[
\frac{\partial^2 p}{\partial t^2} \approx \frac{p_{t+\Delta t} - 2p_t + p_{t-\Delta t}}{\Delta t^2}.
\] (8)

To guarantee accuracy, the Laplacian is discretized with the fourth-order central difference
\[
\nabla^2 p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial z^2} \approx \frac{1}{\Delta x^2} \sum_{i=-2}^{2} c_i p'_{x+i,z} + \frac{1}{\Delta z^2} \sum_{i=-2}^{2} c_i p'_{x,z+i} - \frac{p'_{x+z} + p'_{x-z}}{\Delta x \Delta z}.
\] (9)

where \( c_{-1} = c_1 = -(1/12), c_{-2} = c_2 = (4/3), \) and \( c_0 = -5/2 \) are the coefficients. For convenience, we write \( x + i := x + i \Delta x \) and \( z + i := z + i \Delta z \). With \( v_{x,z} := v(x,z) \), \( s'_{x,z} := s(t, x, z) \), (1) can be discretized as follows:
\[
s'_{x,z} = \frac{1}{\Delta x^2} \sum_{i=-2}^{2} c_i p_{x+i,z} + \frac{1}{\Delta z^2} \sum_{i=-2}^{2} c_i p_{x,z+i} - \frac{p'_{x+z} + p'_{x-z}}{\Delta x \Delta z}.
\] (10)

To satisfy the Courant–Friedrichs–Lewy (CFL) condition for numerical stability, we take \( \Delta t = 0.001 \) and let the finite
difference scheme solve to $t = 1$. The source is generated by a Ricker wavelet whose peak frequency is set at 15 Hz. Moreover, 120 absorbing layers with damping parameters are applied around the spatial domain to avoid reflection artifacts.

**APPENDIX B**

**RECONSTRUCTED TIME-DOMAIN WAVEFIELDS**

The wave propagation in the time domain describes how the energy spreads outward from the source in a straightforward way. Hence, we attempt to visualize the time-domain wavefields at different time points. Since time-domain wavefields can be obtained by applying inverse Fourier transform to frequency-domain wavefields, we train a PFNO on data with a single source location and ten frequencies, 11–20 Hz. The frequencies are truncated due to the limit of the computational resources although there should be 1000 frequencies theoretically (or 501 by the Hermitian property) if the goal is to perfectly reconstruct the time-domain wavefields with $\Delta t = 0.001$ up to 1 s. For a fair comparison, we filter out other frequencies in the ground truth as well. Thereafter, the predictions of the PFNO and truncated ground truth are transformed to the time domain and the real part of them is preserved, leading to the reconstructed time-domain wavefields. As explained above, the reconstructed wavefields are not the same as the true time-domain wavefields because of the missing frequencies. An example is given in Fig. 16. As time elapses, the regular shape of the wave turns irregular as scatterers show up.

To better observe the error in the time domain, we select two locations, i.e., $(x, z) = (190, 550)$ and $(x, z) = (600, 110)$, for which the trace plot Fig. 17 depicts the time-varying pattern of the wave pressure at these two locations. It clearly shows that the error remains at a low level in a long time span.

**APPENDIX C**

**FORWARDNET ARCHITECTURE**

The architecture of the encoder is shown in Table IV, where conv stands for a convolutional layer. Each convolutional layer consists of a convolution, a batch normalization [45], and a leaky Rectified Linear Unit (ReLU) activation function [46] with slope 0.2. Of note, the kernel size is fixed at $3 \times 3$ for each convolution. The columns in_channel and out_channel represent the number of channels applied in each layer. In addition, we show the changes in data size throughout the layers in the last two columns. As a result, the latent representations have the dimension $512 \times 1 \times 1$.

On the other hand, Table V reveals the details of the decoder structure. The aforementioned convolutional layer is also a part of the building block in the decoder. In Table V, the kernel is set in size $3 \times 3$ as well. The stride and the padding are hidden since they are fixed at 1 and 1, respectively, in all the convolutions. Moreover, up_conv stands for the combination of an upsampling operation and a convolutional layer. To avoid the “checkerboard effect,” ForwardNet employs up-sampling instead of deconvolution. The up-sampling operation is implemented with the nearest-value mode and has a parameter scale that controls the output size. Besides up_conv and conv, the crop layer removes the surrounding part of the intermediate representation and retrieves the middle component to obtain the $70 \times 70$ size that matches the wavefield. Finally, the output channel is 2 because the wavefield is complex-valued. The two channels correspond to the real part and the imaginary part.

**APPENDIX D**

**GENERALIZATION ON FREQUENCY**

Since PFNO does not actually take frequency as a variable parameter, our experiments therefore target an FNO with a

---

TABLE IV

| layer | in_channel | out_channel | stride | padding | in_size | out_size |
|-------|------------|-------------|--------|---------|---------|----------|
| conv  | 3          | 32          | 2      | 1       | $70 \times 70$ | $35 \times 35$ |
| conv  | 32         | 32          | 1      | 1       | $35 \times 35$ | $35 \times 35$ |
| conv  | 32         | 64          | 2      | 1       | $35 \times 35$ | $18 \times 18$ |
| conv  | 64         | 64          | 1      | 1       | $18 \times 18$ | $18 \times 18$ |
| conv  | 64         | 128         | 2      | 1       | $18 \times 18$ | $9 \times 9$ |
| conv  | 128        | 128         | 1      | 1       | $9 \times 9$ | $5 \times 5$ |
| conv  | 128        | 256         | 2      | 1       | $9 \times 9$ | $5 \times 5$ |
| conv  | 256        | 256         | 1      | 1       | $5 \times 5$ | $3 \times 3$ |
| conv  | 256        | 512         | 2      | 1       | $5 \times 5$ | $3 \times 3$ |
| conv  | 512        | 512         | 1      | 1       | $3 \times 3$ | $3 \times 3$ |
| conv  | 512        | 512         | 2      | 0       | $3 \times 3$ | $1 \times 1$ |

---

(a)

(b)
width of 128 because of a higher complexity in the data when all the frequencies are included. It is trained with multiple frequencies, and only one source location is included in the training data to dampen the impact of the source location.

More specifically, the FNO is trained on the CurveFaultA data of 4–33 Hz with interval 1 Hz. However, the interval is set to be 0.1 Hz in the testing since we only expect the testing results to be accurate near the training frequencies. In Fig. 18, we test whether the FNO generalizes to neighboring frequencies by measuring the relative mse corresponding to the wavefields of different frequencies. In particular, relative mse = mse/Avg(|μ|) is computed to exclude the impact of the magnitude of the wavefields across the frequencies. Since only 4 Hz belongs to the training frequencies, Fig. 18(a) illustrates how FNO generalizes toward lower frequencies. Similarly, Fig. 18(f) shows the generalization toward higher frequencies. It is not surprising to see a decrease in accuracy as we move far away from the boundary. In comparison, the two boundary frequencies in Fig. 18(b)–(e) are training frequencies. Hence, the error is anticipated to increase as we move toward the middle, and a bell-shaped relative mse curve should occur. Although Fig. 18(c) turns out to be an exception, the relative mse remains smaller than that of the right boundary.

**Appendix E**

**Wavefields With the Free Surface Condition**

To test the performance of PFNO on the wavefields with the free surface condition, we add an experiment based on the CurveFault-A dataset. The exact same configuration is applied to generate the wavefields except that the new dataset considers the free surface condition. In general, the performance of PFNO degrades due to the higher complexity in the wavefields. The testing mse increases from $3.36 \times 10^{-5}$ to $1.14 \times 10^{-4}$. Fig. 19 illustrates two examples of the predicted wavefields. Although the performance of PFNO remains similar in the simpler wavefields, the accuracy decreases in the harder cases. It is not surprising since these wavefields have more complex scatters than the wavefields without the free surface condition. A potential solution is to follow the physics-informed learning

---

**Table V**

| layer       | in_channel | out_channel | scale | in_size     | out_size     |
|-------------|------------|-------------|-------|-------------|--------------|
| up_conv     | 512        | 512         | 5     | 1 x 1       | 5 x 5        |
| conv        | 512        | 256         | 2     | 5 x 5       | 10 x 10      |
| up_conv     | 256        | 256         | 2     | 10 x 10     | 20 x 20      |
| conv        | 128        | 128         | 2     | 20 x 20     | 40 x 40      |
| up_conv     | 64         | 64          | 2     | 40 x 40     | 80 x 80      |
| conv        | 32         | 32          | 2     | 80 x 80     | 160 x 160    |
| crop        | 80 x 80    | 70 x 70     |       |             |              |
| conv        | 32         | 2           |       | 70 x 70     |              |
framework where we enforce a PDE constraint to adapt to the free surface condition.

REFERENCES

[1] A. R. Barron, “Approximation and estimation bounds for artificial neural networks,” Mach. Learn., vol. 14, no. 1, pp. 115–133, Jan. 1994.

[2] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” Math. Control, Signals Syst., vol. 2, no. 4, pp. 303–314, 1989.

[3] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” Neural Netw., vol. 2, no. 5, pp. 359–366, Jan. 1989.

[4] T. Chen and H. Chen, “Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems,” IEEE Trans. Neural Netw., vol. 6, no. 4, pp. 911–917, Jul. 1995.

[5] S. Lanthaler, S. Mishra, and G. E. Karniadakis, “Error estimates for DeepONets: A deep learning framework in infinite dimensions,” Trans. Math. Appl., vol. 6, no. 1, Mar. 2022, Art. no. tmac001.

[6] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, “Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators,” Nature Mach. Intell., vol. 3, pp. 218–229, Mar. 2021.

[7] Z. Li et al., “Fourier neural operator for parametric partial differential equations,” 2020, arXiv:2010.08895.

[8] Z. Li et al., “Neural operator: Graph kernel network for partial differential equations,” 2020, arXiv:2003.03485.

[9] N. Kovachki, S. Lanthaler, and S. Mishra, “On universal approximation and error bounds for Fourier neural operators,” J. Mach. Learn. Res., vol. 22, no. 1, pp. 13237–13312, 2021.

[10] N. Kovachki et al., “Neural operator: Learning maps between function spaces with applications to PDEs,” J. Mach. Learn. Res., vol. 24, no. 89, pp. 1–97, 2023.

[11] A. Tarantola, “Inversion of seismic reflection data in the acoustic approximation,” Geophysics, vol. 49, no. 8, pp. 1259–1266, Aug. 1984.

[12] J. Virieux and S. Operto, “An overview of full-waveform inversion in exploration geophysics,” Geophysics, vol. 74, no. 6, pp. 1–26, Nov. 2009.

[13] H. Wang and T. Alkhalifah, “Microseismic imaging using a source function independent full waveform inversion method,” Geophys. J. Int., vol. 214, no. 1, pp. 46–57, Jul. 2018.

[14] H. Wang, Q. Guo, T. Alkhalifah, and Z. Wu, “Regularized elastic passive equivalent source inversion with full-waveform inversion: Application to a field monitoring microseismic data set,” Geophysics, vol. 85, no. 6, pp. 207–219, Nov. 2020.

[15] P. Jin, X. Zhang, Y. Chen, S. Huang, Z. Liu, and Y. Lin, “Unsupervised learning of full-waveform inversion: Connecting CNN and partial differential equation in a loop,” in Proc. 10th Int. Conf. Learn. Represent. (ICLR), 2022, pp. 1–20.

[16] J. Sun, K. A. Innanen, and C. Huang, “Physics-guided deep learning for seismic inversion with hybrid training and uncertainty analysis,” Geophysics, vol. 86, no. 3, pp. 303–317, May 2021.

[17] J. Berg and K. Nystrom, “A unified deep artificial neural network approach to partial differential equations in complex geometries,” Neurocomputing, vol. 317, pp. 28–41, Nov. 2018.

[18] J. Han and A. Jentzen, “Solving high-dimensional partial differential equations using deep learning,” Proc. Nat. Acad. Sci. USA, vol. 115, no. 34, pp. 8505–8510, Aug. 2018.

[19] Z. Long, Y. Lu, X. Ma, and B. Dong, “PDE-Net: Learning PDEs from data,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 3208–3216.

[20] A. Malek and R. S. Beidokhti, “Numerical solution for high order differential equations using a hybrid neural network—Optimization method,” Appl. Math. Comput., vol. 183, no. 1, pp. 260–271, Dec. 2006.

[21] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” J. Comput. Phys., vol. 378, pp. 686–707, Feb. 2019.

[22] N. Wang, D. Zhang, H. Chang, and H. Li, “Deep learning of subsurface flow via theory-guided neural network,” J. Hydrol., vol. 584, May 2020, Art. no. 124700.

[23] D. Zhang, L. Lu, L. Guo, and G. E. Karniadakis, “Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems,” J. Comput. Phys., vol. 397, Nov. 2019, Art. no. 108850.
Bian Li received the B.S. degree in finance from the Zhongnan University of Economics and Law, Wuhan, China, in 2015, the M.S. degree in finance from Johns Hopkins University, Baltimore, MD, USA, in 2016, the M.S. degree in financial engineering from Lehigh University, Bethlehem, PA, USA, in 2018, and the Ph.D. degree in industrial engineering from Lehigh University, in 2023. He is also a Deep Learning Graduate Research Intern at the Los Alamos National Laboratory, Los Alamos, NM, USA, working on solving seismic wave equations with neural networks. His research interests include dynamical systems, physics-informed machine learning, and probabilistic modeling.

Hanchen Wang received the B.S. degree in geophysics from Tongji University, Shanghai, China, and the M.S. and Ph.D. degrees in geophysics from the King Abdullah University of Science and Technology, Thuwal, Saudi Arabia, in 2016 and 2021, respectively. In 2022, he joined the Los Alamos National Laboratory, Los Alamos, NM, USA, as a Post-Doctoral Research Scholar. His research interests include wave simulation, seismic imaging and inversion using physics-based and machine-learning-based methods, including full waveform inversion, passive seismic imaging, data-driven seismic inversion, and physics-informed machine learning.

Shihang Feng received the B.S. degree in geophysics from the China University of Petroleum, Beijing, China, in 2012, the M.S. degree in geophysics from The University of Utah, Salt Lake City, UT, USA, in 2014, and the Ph.D. degree in Earth science and engineering from the King Abdullah University of Science and Technology, Thuwal, Saudi Arabia, in 2019. He currently holds a post-doctoral position at the Los Alamos National Laboratory, Los Alamos, NM, USA.

Xiu Yang received the B.S. and M.Sc. degrees in computational mathematics from Peking University, Beijing, China, in 2005 and 2008, respectively, and the Ph.D. degree in applied mathematics from Brown University, Providence, RI, USA, in 2014. He joined Lehigh University, Bethlehem, PA, USA, from the Pacific Northwest National Laboratory (PNNL), Richland, WA, USA, where he has been a Scientist since 2016. His recent research has been centered around modern scientific computing, including uncertainty quantification, multiscale modeling, physics-informed machine learning, and quantum computing. Dr. Yang received the Faculty Early Career Development Program (CAREER) Award from the National Science Foundation in 2022 and the Outstanding Performance Award from PNNL in 2015 and 2016.

Youzuo Lin received the Ph.D. degree in computational and applied mathematics from Arizona State University, Tempe, AZ, USA, in 2010. He was a Staff Scientist with the Los Alamos National Laboratory, Los Alamos, NM, USA, for 12 years. He is currently an Associate Professor with The University of North Carolina at Chapel Hill, Chapel Hill, NC, USA. He has worked on applications including seismic inverse imaging, carbon sequestration and monitoring, event detection from time series, unmanned aerial vehicle imagery analysis, hydrology inverse modeling, and medical ultrasound tomography. He has authored more than 100 articles in top journals and conference proceedings. He has been a Co-Inventor on a couple of U.S. patents about ultrasound imaging techniques. His research interests include physics-informed machine learning, optimization, and numerical methods for ill-posed inverse problems. Dr. Lin is an Associate Editor of Machine Learning: Science and Technology (IOPScience). He coedited a 2021 special issue of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and a 2019 special issue of the Journal of Interpretation: Machine Learning in Seismic Data Analysis.