Learning to Solve PDE-constrained Inverse Problems with Graph Networks

Zhao Qingqing [1] **David B. Lindell** [1] **Gordon Wetzstein** [1]

Abstract

Learned graph neural networks (GNNs) have recently been established as fast and accurate alternatives for principled solvers in simulating the dynamics of physical systems. In many application domains across science and engineering, however, we are not only interested in a forward simulation but also in solving inverse problems with constraints defined by a partial differential equation (PDE). Here we explore GNNs to solve such PDE-constrained inverse problems. We demonstrate that GNNs combined with autodecoder-style priors are well-suited for these tasks, achieving more accurate estimates of initial conditions or physical parameters than other learned approaches when applied to the wave equation or Navier–Stokes equations. We also demonstrate computational speedups of up to $90\times$ using GNNs compared to principled solvers.

1. Introduction

Understanding and modeling the dynamics of physical systems is crucial across science and engineering. Among the most popular approaches to solving partial differential equations (PDEs) are mesh-based finite element simulations, which are widely used in electromagnetism (Pardo et al., 2007), aerodynamics (Economon et al., 2016; Ramamurti & Sandberg, 2001), weather prediction (Bauer et al., 2015), and geophysics (Schwarzbach et al., 2011). Learning-based methods for mesh-based simulations have recently made much progress (Pfaff et al., 2021), offering faster runtimes than principled solvers, better adaptivity to the simulation domain compared to grid-based convolutional neural network (CNNs) (Thürey et al., 2020; Wandel et al., 2021), and generalization across simulations. State-of-the-art learning-based mesh simulators operate on adaptive meshes using graph networks (Pfaff et al., 2021). While this approach has been successful in simulating the dynamics of physical systems, it remains unclear how to leverage these network architectures to efficiently solve PDE-constrained inverse problems of the form in Eq. (1).

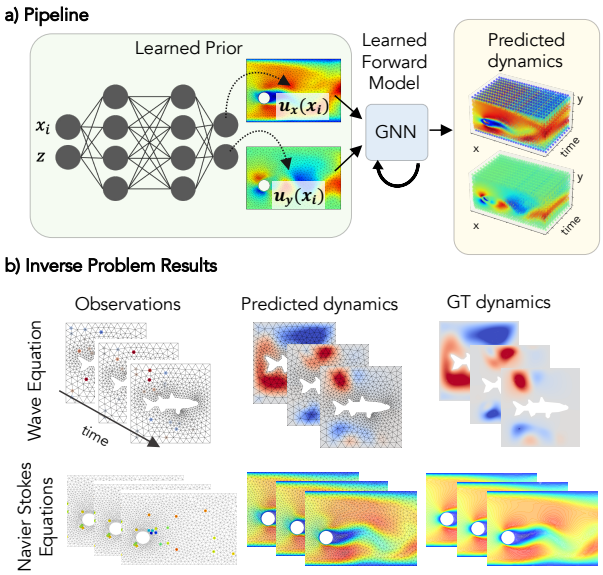

Figure 1. **(a)** Pipeline for our approach: we first pre-train the prior and a Graph Neural Network (GNN) for a given class of problems in a supervised manner using a dataset generated from classical FEM solvers. Here, the GNN is a fully-differentiable forward simulator and the prior is a generative model learned over the space of physics parameters of interest. At test time, the generative model maps the latent code to the estimated physics parameter that is passed to the GNN to obtained the predicted dynamics. The latent code is then optimized to minimize the difference between predicted and observed dynamics on the graph. **(b)** We demonstrate the proposed method for solving inverse problems governed by the wave equation or Navier–Stokes equations. From sparse observations, the method recovers the complete evolution of the dynamics.

$$\underset{\{u_0,\theta\}}{\text{minimize}}\ \mathcal{L}\left(\mathcal{H}\left(u_{t=1\ldots T}\right), y_{t=1\ldots T}\right), \quad s.t.\ u_{t+1} = \mathcal{M}_\theta\left(u_t\right), \tag{1}$$

where $u_t \in \mathbb{C}^N$ is the state of the simulation modeled at N nodes of the mesh at time $t = 1 \ldots T$; the operator $\mathcal{H} : \mathbb{C}^N \to \mathbb{C}^M$ models the measurement process of the real or

[1]Stanford University. Correspondence to: Zhao Qingqing <cyanzhao@stanford.edu>, David Lindell <lindell@stanford.edu>, Gordon Wetzstein <gordon.wetzstein@stanford.edu>.

2nd AI4Science Workshop at the 39th International Conference on Machine Learning (ICML), 2022. Copyright 2022 by the author(s).

complex-valued state using a set of M sensors, giving rise to the observations $y_t \in \mathbb{C}^M$; $\mathcal{M}_\theta : \mathbb{C}^N \to \mathbb{C}^N$ is an operator representing the PDE constraints that govern the temporal dynamics of the system with $\theta \in \mathbb{R}^P$ parameterizing $\mathcal{M}$ at time t. In the context of inverse problems, we seek the initial state u_0 or the parameters θ given a set of observations $y_{t=1\ldots T}$. These problem are often ill-posed and considerably more difficult than learning the forward simulation.

Here we explore efficient approaches that leverage recently proposed graph networks to solve PDE-constrained inverse problems. For this purpose, we consider learning-based approaches to the simulation, where $\mathcal{M}$ is modeled by a graph network $\text{GNN} : \mathbb{C}^N \to \mathbb{C}^N$. Moreover, we formulate priors for both initial state $u_0 = \mathcal{G}_u(z_u)$ and the parameters $\theta = \mathcal{G}_\theta(z_\theta)$ that constrain these quantities to be fully defined in a lower-dimensional subspace by the latent codes $z_u \in \mathbb{R}^{L_u}$, $L_u < N$, and $z_\theta \in \mathbb{R}^{L_\theta}$, $L_\theta < P$. The fact that the graph network GNN, and finite element methods in general, operate on irregular meshes motivates us to explore emerging coordinate network architectures (Park et al., 2019; Tancik et al., 2020; Sitzmann et al., 2020) as suitable priors. Coordinate networks operate on the continuous simulation domain and map coordinates to a quantity of interest, such as the initial condition or the parameters of a specific problem.

To our knowledge, this is the first approach to combining continuous coordinate networks with graph networks for learning to solve PDE-constrained inverse problems. The summary of the proposed approach is shown in Figure 1. We demonstrate that this architecture affords faster runtimes and better quality for fewer observations than the principled solvers we tested, while offering the same benefits of improved accuracy over grid-based CNNs to inverse problems that graph networks offer for forward simulations (Figure 1).

2. Related Work

Machine learning methods have emerged as a powerful tool for modeling the dynamics of physical systems. Compared to conventional methods (see e.g., Quarteroni & Valli (2008)), physics-based machine learning techniques may offer improved computational performance, ease of implementation, and learning priors from data to solve ill-posed physics-based inverse problems. Previous techniques span a spectrum of being entirely data-driven (i.e., modeling dynamics with a feedforward pass through a network) (Bhatnagar et al., 2019; Li et al., 2021; Tompson et al., 2017; Lu et al., 2021; Lino et al., 2021; Pfaff et al., 2021; Sanchez-Gonzalez et al., 2020; Seo et al., 2020) to hybrid approaches that either directly incorporate conventional solvers (Um et al., 2020; Belbute-Peres et al., 2020; Bar-Sinai et al., 2019; Kochkov et al., 2021) or adapt similar optimization schemes or constraints (Karniadakis et al., 2021; Wang & Perdikaris, 2021; Sirignano & Spiliopoulos, 2018; Raissi et al., 2019; Wang et al., 2021; Wandel et al., 2021).

Many of the aforementioned methods operate on regular grids, for example using convolutional neural networks (CNNs). A major disadvantage of this approach is its inability to allocate resolution adaptively throughout the simulation domain. Learning-based finite element methods (FEMs) (Xue et al., 2020) and *graph neural networks* (GNNs) (Kipf & Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018; Wu et al., 2020; Scarselli et al., 2008) have been shown to be a powerful method for simulation on irregular grids with adaptive resolution. GNNs process data structured into a graph of nodes and edges, and have been demonstrated for modeling complex physical dynamics with particle- or mesh-based simulation (Lino et al., 2021; Seo et al., 2020; Li et al., 2020b;a). Our approach is inspired by recent GNN architectures for modeling time-resolved dynamics (Pfaff et al., 2021; Sanchez-Gonzalez et al., 2020) that generalize to unseen initial conditions and solution domains. Whereas these approaches focus on learning the forward simulation given a set of initial or boundary conditions, ours aims at solving ill-posed PDE-constrained inverse problems that require a learned simulation model as part of the framework.

Machine learning techniques have recently also been adapted for solving ill-posed physics-based inverse problems. These problems often involve recovering solutions to PDEs (Raissi et al., 2020) or physical quantities, such as the density, viscosity, or other material parameters (Mosser et al., 2020; He & Wang, 2021; Fan et al., 2020), from a sparse set of measurements. Other methods can super-resolve PDE solutions from estimated solutions at coarse resolution (Esmaeilzadeh et al., 2020) or recover initial conditions from PDE solutions observed at coarse resolution later in time (Li et al., 2020b; Frerix et al., 2021). Yet another class of techniques are tailored to the inverse design problem, which aims to optimize material properties such that the PDE solution satisfies certain useful properties, e.g., for the design of nanophotonic elements (Molesky et al., 2018). Our approach is most similar to other methods that infer material properties or initial conditions from sparse measurements of the PDE solution over time. Similar to some of these methods (Mosser et al., 2020; Jiang & Fan, 2019; Hand et al., 2018), we use a generative model to learn a prior over the solution space of material parameters or initial conditions; however, our framework is the first to learn to map a latent code to material parameters or initial conditions compatible with a GNN forward model. This allows our model to solve physics-based inverse problems on irregular grids with adaptive resolution, leading to improved computational efficiency and accuracy.

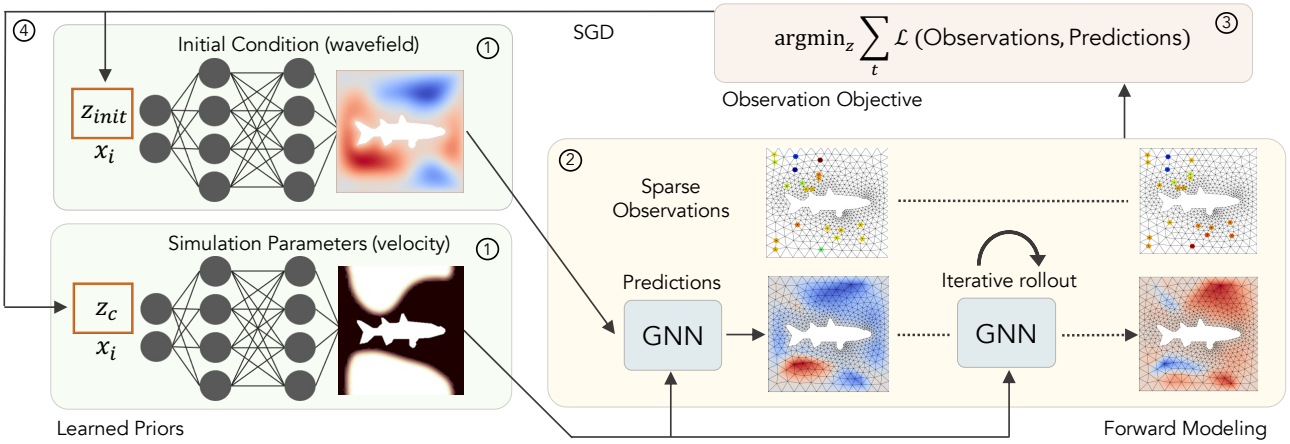

Figure 2. Pipeline for solving inverse problems (illustrated for the 2D wave equation). The forward simulator GNN (the blue boxes) and the prior networks (green boxes) are pre-trained with a dataset of wave trajectories generated using classical FEM solvers. We aim to recover the initial wavefield or the velocity distribution. At test time, the generative model first maps a latent code to the estimated physics parameters (step 1) that is passed to the GNN to obtain the predicted dynamics (step 2). The latent code is optimized to minimize the difference between the predicted and the observed dynamics on the graph (steps 3 and 4).

3. Method

We approach the task of solving physics-constrained inverse problems by first pre-training a GNN as a fast forward model that outputs solutions to partial differential equations given the initial states and boundary conditions. The physics-constrained inverse problems we aim to solve involve recovering initial conditions or other parameters that govern the evolution of the PDE. Since recovering these quantities is an ill-posed problem, we also learn a prior over the space of solutions using a deep generative model. Here, a network maps latent codes to a space of material parameters or initial conditions on the graph that comprise plausible solutions to the inverse problems. At test time, the GNN is a fully-differentiable forward simulator, and the latent code of the generative model is optimized to minimize the difference between the predicted and the observed dynamics on the graph. A detailed pipeline of our approach for solving inverse problem for the wave equation (Section 4) is shown in Figure 2.

3.1. GNN-based Simulator

GNNs have previously been found to perform well in simulating PDE models (Pfaff et al., 2021; Sanchez-Gonzalez et al., 2020). The excellent performance can be attributed to their adaptive resolution and ability to generalize by modeling complicated local interactions. Here, we leverage the state-of-art GNN (Pfaff et al., 2021) to learn a fast forward model that outputs solutions to partial differential equations given the initial states and boundary conditions.

The state of the system at time t is described as $u_t = (V, E)$ with nodes V and mesh edges E that define the mesh. We adopt the approach of (Pfaff et al., 2021) to model the temporal dynamics of the system, i.e., $\mathcal{M}_\theta$, using a GNN with Encoder-Processor-Decoder architecture followed by an integrator.

We encode the relative displacement vector (i.e., the vector that points from one node to another) and its norm as edge features, e_i. The node features comprise dynamics quantities that describe the state of the PDE and a one-hot vector, v_i, indicating the type of the nodes (domain boundaries, inlet, outlet, etc.). In the forward pass, the encoding step uses an edge encoder multilayer perceptron (MLP) and node encoder MLP to encode edge features and node features into latent vectors. The processing step consists of several message passing layers with residual connections. This step takes as input the set of node embeddings v_i, and edge embeddings e_{ij}, and outputs an updated embedding v_i and e_{ij}. The equations that define the layers of the message passing steps are given by the following.

$$e'_{ij} = \text{MLP}_e(e_{ij}, v_i, v_j), \tag{2}$$

$$v'_i = \text{MLP}_v(v_i, \sum_j e'_{ij}). \tag{3}$$

In the final decoding step, an MLP is used to transform the latent node features v_i to the output p_i, which we integrate at each time step to update the dynamic quantities $u_{t+1} = u_t + p_t$.

3.2. Learning Inverse Problems and Priors

The inverse problems we wish to solve are of the form outlined in Equation 1 with the dynamics being learned by the GNN described above. Solving for the initial condition or parameters may be ill-posed, so we restrict the solutions to lie in a lower-dimensional subspace of the autodecoder-type priors $\mathcal{G}_u$ or $\mathcal{G}_\theta$. When working on regular grids, these priors could easily be implemented as CNN-based generative model, but it is not clear how to model such priors for irregular meshes. To this end, we leverage recently proposed coordinate networks with ReLU activation functions, which use a conditioning-by-concatenation approach to generalize across signals:

$$\begin{aligned} \mathcal{G}_{u/\theta} &= \mathbf{W}_n \left(\phi_{n-1} \circ \phi_{n-2} \circ \ldots \circ \phi_0\right) \left(\begin{bmatrix} x \\ z_{u/\theta} \end{bmatrix} \right) + \mathbf{b}_n \\ \phi_i\left(x_i\right) &= \mathrm{ReLU}\left(\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i\right), \end{aligned} \tag{4}$$

where $\phi_i : \mathbb{R}^{M_i} \mapsto \mathbb{R}^{N_i}$ is the ith layer of the MLP consisting of the affine transform defined by the weight matrix $\mathbf{W}_i \in \mathbb{R}^{N_i \times M_i}$ and the biases $\mathbf{b}_i \in \mathbb{R}^{N_i}$. The input to the first layer concatenates the coordinate x with a scene-specific latent code vector $z_{u/\theta}$. This coordinate network, implemented by an MLP, maps continuous coordinates on the simulation domain to a quantity of interest. Thus, it can be evaluated on a regular grid or, more importantly, on the irregular locations of the graph nodes our GNN operates on.

We learn $\mathcal{G}_{u/\theta}$ in a pre-processing step, using training data relevant for a specific physical problem. We optimize the network parameter, denoted by $\psi_{\mathcal{G}}$, and the latent code, z, with respect to the individual training sample, indexed by i, to maximize the joint log posterior of all training samples:

$$\underset{\psi_{\mathcal{G}}, z}{\arg\min} \sum_{i=1}^{N} \left(\sum_{j=1}^{K} \mathcal{L}\left(\mathcal{G}\left(z^i, \mathbf{x}_j\right), u_j^i\right) + \frac{1}{\sigma^2} \|z^i\|_2^2 \right) \tag{5}$$

At test time, the pre-trained GNN modeling the dynamics and the pre-trained coordinate networks as the prior are fixed, and the latent code vectors $z_{u/\theta}$ are optimized for a given set of sparse observations using a standard Adam solver (Kingma & Ba, 2014).

4. Experiments

We demonstrate our approach on the wave equation and the Navier–Stokes equations for modeling acoustics or fluid dynamics. Our method leverages a learned simulator using a Graph Neural Network and a learned prior for fast and accurate recovery of the unknown parameters. The U-Net (CNN) solver baseline is comparable to existing approaches in the literature for forward simulation (Thürey et al., 2020; Wandel et al., 2021; Holl et al., 2019; Bhatnagar et al., 2019; Tompson et al., 2017); since we focus on solving physics-constrained inverse problems, we extend this approach accordingly and incorporate our learned prior. We also evaluate the effect of the learned prior. We find that our approach gives a favorable tradeoff between accuracy and speed, enabling accurate recovery of the unknown parameters while being up to nearly two orders of magnitude faster than the classical FEM solver.

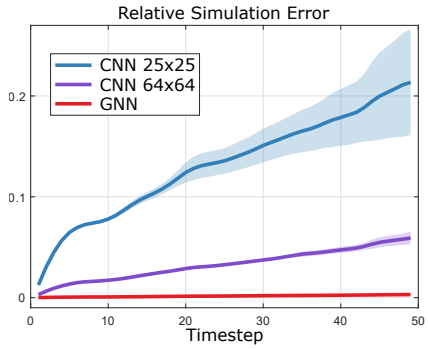

Figure 3. Accumulated relative error of the U-Net (CNN) (on both coarse and fine grids) and GNN-based forward simulators for varying number of time steps.

4.1. Two-Dimensional Scalar Wave Equation

The wave equation, Eq. (6), is a second-order partial differential equation that describes the dynamics of acoustic or electromagnetic wave propagation.

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2} &= 0 \\ u_0' = \frac{\partial u}{\partial t}|_{t=0} &= 0 \\ u_0 &= u_{\text{init}} \end{aligned} \tag{6}$$

Here, u is the field amplitude and c is the velocity parameter that spatially characterizes the speed of the wave propagation inside the medium. The wave equation can be solved given initial conditions, u_0' and u_{init}, and the boundary conditions. In our experiments, the domain is chosen to be $\Omega \in [0, 1]^2$ with an obstacle at the center. The Dirichlet boundary conditions are used at all boundaries, $u|_{\partial\Omega} = 0$. The initial field velocity is set to be 0, and the initial field distribution is denoted by u_{init}.

The PDE-constrained inverse problem we are aiming to solve is to recover unknown parameters u_{init} or c from sparse measurements of u in space and time. Formally, it can be

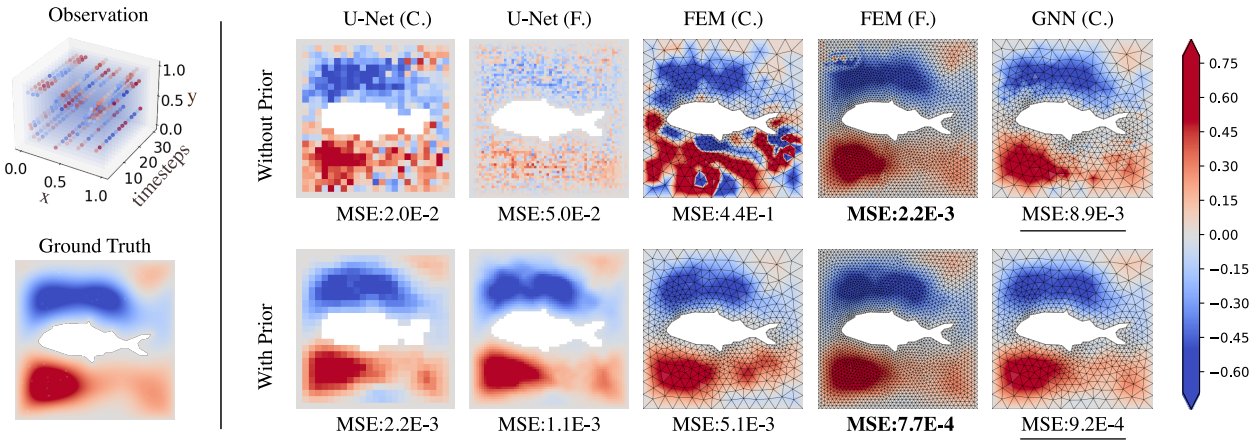

Figure 4. Qualitative results for the initial state recovery problem with the 2D wave equation. Here "C." refers to the coarse meshes (25×25 nodes for U-Net (CNN) or ≈ 600 nodes for GNN), and "F." refers to the fine meshes (64×64 nodes for U-Net (CNN) or ≈ 2800 nodes for the FEM solver). Without the learned prior, $\mathcal{G}$, all methods fail due to the ill-posed nature of the problem. Using the prior, we find the GNN yields a result that is closer to the ground truth compared to both U-Net (CNN) models. While the FEM solver on the fine grid outperforms the GNN, it is also $\approx 8\times$ slower. Thus the GNN with the prior gives a favorable tradeoff between speed and accuracy.

described as

$$\begin{aligned} \underset{\{z\}}{\text{minimize}} \quad & \sum_{t \in T_{\text{iter}}} \|\mathcal{H}(u_t^{\text{gt}}(\mathbf{x})) - \mathcal{H}(u_t^{\text{pred}}(\mathbf{x}))\|_2^2 \\ s.t. \quad & \underbrace{u_{t+1} = u_t + \mathcal{M}_c\left(u_t, u_t'\right)}_{u_0(\mathbf{x}) = \mathcal{G}_{u_{\text{init}}}(z, \mathbf{x}) \quad \text{or} \quad c(\mathbf{x}) = \mathcal{G}_c(z, \mathbf{x})} \end{aligned} \tag{7}$$

Here, we minimize the mean squared error (MSE) between the predicted field (u_t^{pred}) and the ground truth observed field (u_t^{gt}), summed over different observation time steps, t. We address two different tasks, (1) recovering the initial condition, u_{init}, and (2) full-waveform inversion (Virieux & Operto, 2009) in which we recover the velocity parameter, c. These optimized quantities are parameterized by the prior network $\mathcal{G}_{u_{\text{init}}/c}$. Finally, the measurement sampling operator is given by $\mathcal{H}$, and $\mathcal{M}_c$ is the forward model that solves the wave equation, Eq. (6), to produce the field at the next time step.

Learned Forward Simulation In our approach, $\mathcal{M}_c$ is given by a GNN, which learns the wave equation forward model on irregular meshes.

As a baseline, we train a U-Net (CNN) forward model based on (Thürey et al., 2020). Similar to (Pfaff et al., 2021), we train the GNN and U-Net (CNN) by directly supervising on a dataset of "ground truth" wave equation solutions generated with an open source FEM solver (FEniCS (Logg et al., 2012)). The FEM solver uses a fine irregular mesh with many more nodes ($\approx$2800) than the GNN ($\approx$600) on average across the dataset. The training dataset is composed of 1100 simulated time-series trajectories using 37 separate meshes. We evaluate on 40 held-out trajectories across 3 different meshes. We find that the FEM solver requires a timestep that is $5\times$ smaller than the GNN solver to achieve stable results. At this setting, the FEM solver simulated on a fine irregular mesh is roughly $8\times$ slower than the GNN; running the FEM solver on the same grid as the GNN is $2.5\times$ slower than the GNN with accuracy roughly $80\times$ worse in terms of MSE.

To compare the accuracy of different learned forward simulators, we plot the accumulated relative error, unrolled for 50 time steps and averaged over 96 trajectories in Fig. 3. The GNN-based simulator provides a robust solution. We evaluate two different cases for the U-Net (CNN). In one case, we restrict the number of grid nodes to be roughly the same as the GNN (25×25). Since the small number of nodes must be regularly and coarsely spaced, this provides poor performance. Second, we set the grid resolution of the U-Net (CNN) such that the spacing between grid points matches the minimum spacing in the GNN mesh. This requires $7\times$ more nodes than the GNN (64×64), but the performance is still relatively worse than the GNN.

Initial State Recovery For the initial state recover task, we solve Eq. (7) where $u_{\text{init}}(\mathbf{x}) = \mathcal{G}_{u_{\text{init}}}(z, \mathbf{x})$ is the unknown parameter and initial velocity is given to be 0. For both this problem and full-waveform inversion, the prior is pre-trained on a dataset of 10,000 values of u_{init} (or c), generated by sampling a Gaussian random field and tapering the solution to zero near the boundaries to satisfy the Dirichlet boundary condition.

Forward Model	# nodes	Initial State Recovery		FWI		Runtimes
		MSE	MSE (with $\mathcal{G}$)	MSE	MSE (with $\mathcal{G}$)	(s)
FEM (Irr. F.)	2827	**1.55e-3**	**7.82e-4**	3.91e-1	1.19e-1	1.25e1
FEM (Irr. C.)	611	1.72e-1	4.18e-3	3.88e-1	1.64e-1	6.03e0
U-Net (Reg. C.)	625	1.32e-2	3.52e-3	4.19e-1	1.54e-1	**1.71e-1**
U-Net (Reg. F.)	4096	3.16e-2	1.49e-3	5.14e-1	1.08-1	1.91e-1
GNN (Irr. C.)	611	1.05e-2	8.87e-4	**3.68e-1**	**1.05e-1**	1.44e0

Table 1. Quantitative results for the initial state recovery and full-waveform inversion (FWI) problems for the 2D wave equation. All metrics are averaged over 40 test samples and evaluated on coarse irregular grids. Observation setup: we use measurements every 2 GNN time steps from 2 to 30 GNN time steps with 20 sensors randomly sampled from the grid. Runtime measures the time taken per optimization iteration. We observe that the GNN outperforms the U-Net (CNN) with a similar number of nodes and provides comparable results with the U-Net (CNN) using $7\times$ more nodes. The classical FEM solver using a fine irregular mesh gives the lowest MSE, however it is at least $8\times$ slower than the learned simulator approaches.

We solve the inverse problems by optimizing the value of z using the ADAM optimizer (Kingma & Ba, 2014) for all experiments until convergence, or a maximum of 2000 iterations. In the case where we optimize without the prior, we directly optimize the unknown parameters, u_{init} or c.

Table (1) reports a quantitative comparison of different solvers with and without the prior, and a qualitative comparison is shown in Fig. 4. Here we unroll the GNN for 30 GNN time steps and take measurements every 2 GNN steps from 2 to 30 time steps. At every measurement, we have sensors placed at 20 randomly sampled nodes from the coarse irregular mesh. For baseline comparisons with other form of meshes (regular for U-Net (CNN) and fine irregular mesh for FEM solver), we use nearest neighbor sampling to find the corresponding sensor location. From the table we observe that with or without the prior, our approach provides a favorable trade-off between the accuracy and the speed. While the U-Net (CNN) is the fastest forward simulator, the GNN solver outperforms the U-Net (CNN) solver in terms of accuracy at resolution 25×25 and performs slightly better than the U-Net (CNN) at resolution 64×64. We attribute this to the stronger performance of the GNN forward model, as shown in Fig. 3. Given that the FEM solver operating on the finest mesh gives the most accurate forward model, it gives the best MSE in the inverse problem. However, it is at least $8\times$ slower than the learned simulator approaches. For all cases, using a learned prior significantly improves the final accuracy. By optimizing the latent code, we constrain the solution space to the manifold learned by $\mathcal{G}_{u_{\text{init}}/c}$ and avoid bad local optima that lie far outside the dataset distribution.

Full Waveform Inversion Full-waveform inversion (FWI) is a common problem in seismology (Virieux et al., 2017) and involves recovering the density of structures in the propagation medium. We are motivated by recent work that uses variants of a CNN as a prior (Mosser et al., 2020; Wu & McMechan, 2019) or forward operator (Yang et al., 2021) for seismic inversion. As we are primarily interested in a comparison between learned methods on regular and irregular meshes, we adopt the U-Net (CNN) baseline as a representative method that can be unrolled per timestep and integrated into our framework. Since the prior is agnostic to the particular grid setup, we use the same prior across all models.

For the FWI problem, we minimize Eq. (7) where $c(\mathbf{x}) = \mathcal{G}_c(z, \mathbf{x})$ is the unknown parameter. FWI is a highly nonlinear inverse problem, and the final optimization result depends heavily on the initial state and may easily fall into local minima. In order to improve the optimization, different forms of progressive training schemes have been used. For example, one can progressively fit the data in time by introducing a damping function as in (Chen et al., 2015). In frequency-domain full-waveform inversion, one can first optimize low-frequencies to avoid local minima (Aghamiry et al., 2019; Brenders & Pratt, 2007). In our approach, we adopt a progressive training scheme in time where we gradually increase the total number of observed time steps as the optimization proceeds. For the FWI task, we unroll the GNN for 30 GNN time steps and take measurements every 2 GNN steps. At the beginning of the optimization, we only have observations at time step $T = \{2\Delta t\}$, and we include one extra time step's measurement every 120 optimization iterations until $T = \{2\Delta t, 4\Delta t, \cdots, 30\Delta t\}$.

Table (1) reports a quantitative comparison of the performance of different solvers used with or without the prior. We observe similar trends as in the initial state recovery task, where using a learned prior significantly improves the final accuracy. For this task, the proposed GNN solver gives the lowest MSE compared to the baselines. Figure 5 shows qualitative comparisons between the different learned simulators and demonstrates that the proposed approach with the GNN forward model leads to a better recovery of the density distribution.

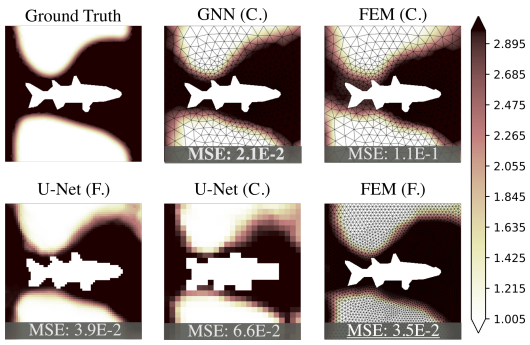

Figure 5. Qualitative results for full-waveform inversion experiments. Here "C." refers to the coarse meshes and "F." refers to the fine meshes, consistent with the notation used in Fig. 4. We observe that the GNN outperforms the U-Net (CNN) both quantitatively and qualitatively.

4.2. Two-Dimensional Incompressible Navier–Stokes Equation

The two-dimensional incompressible Navier–Stokes Equations are non-linear partial differential equations modeling the dynamics of fluids. Taking a fluid density of $\rho = 1.0$, the fluid is characterized by

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\nabla\mathbf{u} - \nu\nabla^2\mathbf{u} + \nabla p = 0 \tag{8}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{9}$$

Here, $\nu = 0.001$ is the kinematic viscosity, $\mathbf{u}$ represents the velocity of the fluid in the x and y directions, and p represents the pressure. These incompressible Navier–Stokes equations can be solved given the initial condition $\mathbf{u}$ and boundary conditions for the given domain. For our experiments, we use an irregular domain $\Omega \in [0, 1.6] \times [0, 0.4]$ with a cylinder of random radius at a random position near the flow inlet. No-slip boundary conditions are defined at the lower and upper edge, as well as at the boundary of the cylinder as shown in Eq. (10). On the left edge, a constant inflow profile is prescribed, Eq. (11); on the right edge, do-nothing boundary condition is used, Eq. (12).

$$\mathbf{u}(\mathbf{x}) = [0, 0] \quad \forall \mathbf{x} \in \partial\Omega_{\text{up, bottom, cylinder}} \tag{10}$$

$$\mathbf{u}(\mathbf{x}) = [1, 0] \quad \forall \mathbf{x} \in \partial\Omega_{\text{in}} \tag{11}$$

$$\nu\partial_\eta\mathbf{u} - p\eta = 0 \quad \forall \mathbf{x} \in \partial\Omega_{\text{out}} \tag{12}$$

We carry out flow assimilation on the velocity field $\mathbf{u}$ given sparse velocity measurements in space and time. Formally, we have

$$\begin{aligned} \underset{\{z\}}{\text{minimize}} \quad & \sum_{t \in T_{\text{iter}}} \|\mathcal{H}(\mathbf{u}_t^{\text{gt}}(\mathbf{x})) - \mathcal{H}(\mathbf{u}_t^{\text{pred}}(\mathbf{x}))\|_1 \\ s.t. \quad & \mathbf{u}_{t+1} = \mathbf{u}_t + \mathcal{M}(\mathbf{u}_t) \\ & \mathbf{u}_0(\mathbf{x}) = \mathcal{G}_{\mathbf{u}_{\text{init}}}(z, \mathbf{x}). \end{aligned} \tag{13}$$

where $\mathcal{H}$ is the measurement sampling operator, and $\mathcal{M}$ is the forward model that solves the incompressible Navier–Stokes Equations.

Our assimilation trajectories are random flow clips of length T = 10 time intervals from the test dataset. We take measurements every 2 time intervals for a total of 5 measurement snapshots where each time intervals are equivalent to 1 learned solver timesteps, or 12.5 FEM solver timesteps. At every measurements, we have 50 sensors measuring the velocity field, $[u_x, u_y]$, as shown in Fig. 6.

Forward Model	# nodes	MSE	Runtime (s)
FEM (Irr. C.)	2732	**3.92e-4**	3.87e1
U-Net (Reg. C.)	2916	5.78e-3	**1.11e-1**
U-Net (Reg. F.)	16384	2.55e-3	1.65e-1
GNN (Irr. C.)	2732	9.73e-4	1.10e0

Table 2. Errors of different forward models for 50-steps rollout averaged over 50 test trajectories. The proposed method with the GNN provides better accuracy than the U-Net (CNN) at both coarse and fine resolution and is $35\times$ faster than the classical FEM solver.

Learned Forward Simulation As described in Sec. 3, our method uses a GNN as a fast and accurate learned simulator operating on irregular meshes. Similar to the 2D scalar wave equation experiment, we train as baselines a U-Net (CNN) for an input resolution of 54×54, which matches the average number of nodes of the irregular grid used by the GNN ($\approx$2732 nodes). We also train a U-Net (CNN) at 128×128 resolution, which contains roughly $6\times$ more nodes compared to the coarse irregular grid (Irr. C.) used by the GNN. All learned simulators are trained in a supervised manner on a dataset obtained using an open source FEM solver (Logg et al., 2012). Our dataset consists of 850 training trajectories on 55 meshes, and 50 test trajectories on 5 meshes. The unrolled MSE is shown in Table (2). We observe a similar trend as in the wave equation example: the GNN gives the best accuracy among learned solvers and is approximately $35\times$ faster than the FEM solver.

Fluid Data Assimilation We train our prior $\mathcal{G}_{\mathbf{u}}$ on 34000 fluid snapshots from the training trajectories, where $\mathcal{G}_{\mathbf{u}}(z, \mathbf{x}) = \mathbf{u}(\mathbf{x}) = [u_x(\mathbf{x}), u_y(\mathbf{x})]$. Similarly to the wave equation experiment, we solve the optimization problem given in Eq. (13) using the ADAM optimizer (Kingma & Ba, 2014). Different from the wave equation experiments we add a fine-tuning stage to the optimization where we update the parameters of the generative prior $\mathcal{G}_{\mathbf{u}}$, which helps to improve the generalization performance in this problem. Similar techniques have been widely used in GAN inversion problems (Daras et al., 2020; Roich et al., 2021). Figure 7 shows the averaged observation objective with optimiza-

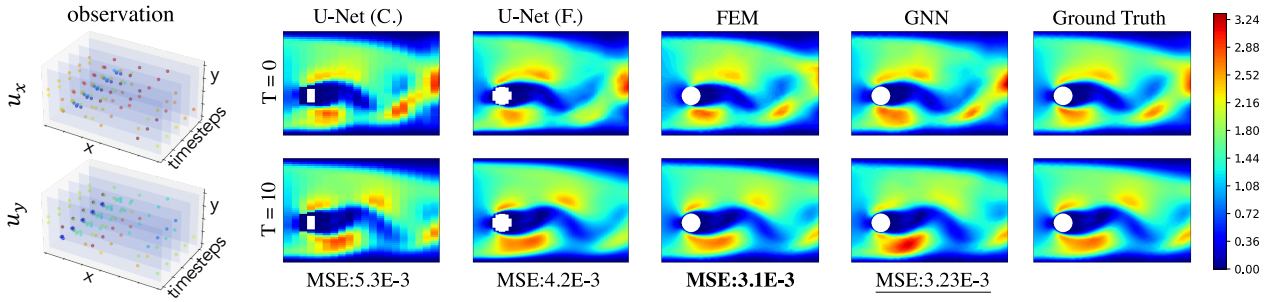

Figure 6. Fluid data assimilation results using different forward models with the prior. Left: we have 50 sensors measuring the velocity field and we take measurements every 2 time steps over a flow clip of total 10 time steps. Right: qualitative comparison of recovered flows using different methods at $t = 0$ (beginning of assimilation window) and $t = 10$ (end of the assimilation window). We visualize the magnitude of the velocity ($\|\mathbf{u}\|_2^2$) for comparison (see supplement for additional comparisons). The reported MSE are the average mean square error over the entire flow assimilation window. While the FEM solver yields the best results, it is roughly $90\times$ slower than the GNN.

Forward Model	**Fluid Data Assimilation**		
	MSE (u_x)	MSE (u_y)	Runtime (s)
FEM (Irr. C.)	**4.46e-3**	1.94e-3	6.39e1
U-Net (Reg. C.)	1.02e-2	5.40e-3	**1.27e-1**
U-Net (Reg. F.)	7.97e-3	3.31e-3	3.25e-1
GNN (Irr. C.)	6.55e-3	**1.86e-3**	6.94e-1

Table 3. Fluid assimilation results averaged over 50 test samples on 5 unseen meshes. Here, we simulate for 10 time steps taking measurement of 50 sampled sensors every 2 time steps. Runtime is the per-iteration optimization time. We observe that the GNN (Irr. C.) performs much better than the U-Net (Reg. C.) using a similar number of nodes (listed in Table (2)). U-Net (Irr. F.) uses $5\times$ more nodes than the GNN but has slightly worse performance. The FEM solver gives the best performance but is $90\times$ slower than the GNN.

tion steps for different forward model and with or without fine-tuning procedure. After beginning the fine-tuning procedure we notice a spike in the objective function, but the optimization ultimately converges to much lower residuals than without the fine-tuning procedure. In Table (3), we qualitatively compare the data assimilation performance for different forward models with the learned prior, and qualitative comparisons can be found in Fig. 6. We report the average MSE over all time steps of u_x and u_y as our evaluation metric. We observe similar trends as in the wave equation experiment: the learned GNN forward model with the learned prior achieves better accuracy compared to the coarse U-Net (CNN) model, which has a similar number of nodes. While the FEM solver gives the best fluid reconstruction, it is $50\times$ slower than the GNN. U-Net (Reg. F.) uses $5\times$ more nodes than the GNN, but achieves similar performance.

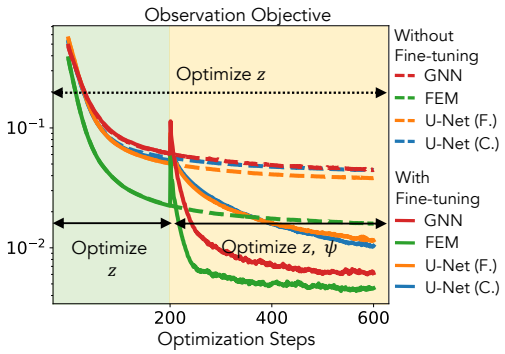

Figure 7. Observation objective during the optimization. Solid lines: during the first 200 iterations, we only optimize the latent code z and the prior, $\mathcal{G}$, parameterized by ψ is fixed. From 200 - 600 iterations, we start to fine tune the prior, i.e optimizing ψ, to increase the expressiveness of the prior to better match the observations. After fine-tuning, we observe a significant decrease in the objective function. Dashed lines: we only optimize the latent code z throughout the optimization. We observe that the final observation objective without fine-tuning is much higher than the cases with fine-tuning.

5. Discussions

In this paper we present a general framework for solving PDE-constrained inverse problems with a GNN-based forward solver and an autodecoder-style learned prior. Both the GNN-based learned simulator and the learned prior operate on irregular meshes with adaptive resolution, enabling representing and processing signals with fewer nodes compared to the regular grids used by convolutional networks. Our experiments on the wave equation and Navier–Stokes equations demonstrate that the proposed framework achieves improved performance compared to a conventional U-Net (CNN) operating on grids with the same number of nodes as the GNN. Moreover, our approach is up to $90\times$ faster compared to conventional FEM solvers.

We also mention a few limitations of the proposed approach. The current implementation is limited in the number of time-steps that can be modeled in the inverse problem due to memory required to unroll the solvers. Also, the method requires the forward model to be trained from scratch for changes to the underlying equations (e.g., significant changes to the boundary conditions or dynamics). Still, techniques like gradient checkpointing (Griewank, 1992) may be helpful to mitigate the memory constraints of the unrolled optimization, and generalizing learned physics solvers across problem settings is an promising direction for future work.

Overall, our approach works to integrate GNNs and learned priors for solving physics-constrained inverse problems. Our approach may be useful for solving ill-posed problems across a range of tasks related to physics-based simulation and modeling. GNNs are an attractive architecture for efficient modeling across multi-resolution domains and may yield even greater improvements compared to conventional CNNs, especially for larger domains or in 3D problem settings.

6. Acknowledgements

This project was supported in part by a PECASE by the ARO and Stanford HAI.

References

Aghamiry, H. S., Gholami, A., and Operto, S. Improving full-waveform inversion by wavefield reconstruction with the alternating direction method of multipliers. *Geophysics*, 84(1):R125–R148, 2019.

Bar-Sinai, Y., Hoyer, S., Hickey, J., and Brenner, M. P. Learning data-driven discretizations for partial differential equations. *Proc. Natl. Acad. Sci. U.S.A*, 116(31): 15344–15349, 2019.

Bauer, P., Thorpe, A., and Brunet, G. The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55, 2015.

Belbute-Peres, F. d. A., Economon, T., and Kolter, Z. Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *Proc. ICML*, 2020.

Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K., and Kaushik, S. Prediction of aerodynamic flow fields using convolutional neural networks. *Comput. Mech.*, 64 (2):525–545, 2019.

Brenders, A. J. and Pratt, R. G. Full waveform tomography for lithospheric imaging: results from a blind test in a realistic crustal model. *Geophys. J. Int.*, 168(1):133–151, 01 2007.

Chen, G., Chen, S., and Wu, R.-S. *Full Waveform Inversion in time domain using time-damping filters*, pp. 1451–1455. 2015.

Daras, G., Dean, J., Jalal, A., and Dimakis, A. G. Intermediate layer optimization for inverse problems using deep generative models. In *Proc. ICML*, 2020.

Economon, T. D., Palacios, F., Copeland, S. R., Lukaczyk, T. W., and Alonso, J. J. Su2: An open-source suite for multiphysics simulation and design. *AIAA J.*, 54(3):828–846, 2016.

Esmaeilzadeh, S., Azizzadenesheli, K., Kashinath, K., Mustafa, M., Tchelepi, H. A., Marcus, P., Prabhat, M., Anandkumar, A., et al. Meshfreeflownet: a physics-constrained deep continuous space-time super-resolution framework. In *SC20*, 2020.

Fan, T., Xu, K., Pathak, J., and Darve, E. Solving inverse problems in steady-state navier-stokes equations using deep neural networks. *AAAI*, 2020.

Frerix, T., Kochkov, D., Smith, J. A., Cremers, D., Brenner, M. P., and Hoyer, S. Variational data assimilation with a learned inverse observation operator. In *Proc. ICML*, 2021.

Griewank, A. Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optim Methods Softw*, 1:35–54, 1992.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Proc. NeurIPS*, 2017.

Hand, P., Leong, O., and Voroninski, V. Phase retrieval under a generative prior. In *Proc. NeurIPS*, 2018.

He, Q. and Wang, Y. Reparameterized full-waveform inversion using deep neural networks. *Geophysics*, 86(1): V1–V13, 2021.

Holl, P., Koltun, V., and Thuerey, N. Learning to control pdes with differentiable physics, 2019.

Jiang, J. and Fan, J. A. Global optimization of dielectric metasurfaces using a physics-driven neural network. *Nano Lett.*, 19(8):5366–5372, 2019.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nat. Rev. Phys.*, 3(6):422–440, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. 2017.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. Machine learning–accelerated computational fluid dynamics. *Proc. Natl. Acad. Sci. U.S.A*, 118(21), 2021.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Multipole graph neural operator for parametric partial differential equations. In *Proc. NeurIPS*, 2020a.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020b.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. 2021.

Lino, M., Cantwell, C., Bharath, A. A., and Fotiadis, S. Simulating continuum mechanics with multi-scale graph neural networks. *CoRR*, 2021.

Logg, A., Mardal, K.-A., Wells, G. N., et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.

Lu, L., Jin, P., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nat. Mach. Intell.*, 2021.

Molesky, S., Lin, Z., Piggott, A. Y., Jin, W., Vucković, J., and Rodriguez, A. W. Inverse design in nanophotonics. *Nat. Photonics*, 12(11):659–670, 2018.

Mosser, L., Dubrule, O., and Blunt, M. J. Stochastic seismic waveform inversion using generative adversarial networks as a geological prior. *Math. Geosci.*, 52(1):53–79, 2020.

Pardo, D., Demkowicz, L., Torres-Verdin, C., and Paszynski, M. A self-adaptive goal-oriented hp-finite element method with electromagnetic applications. part ii: Electrodynamics. *Comput Methods Appl Mech Eng*, 196 (37-40):3585–3597, 2007.

Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. CVPR*, 2019.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. 2021.

Quarteroni, A. M. and Valli, A. *Numerical Approximation of Partial Differential Equations*. Springer, 1st ed. 1994. 2nd printing edition, 2008.

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

Raissi, M., Yazdani, A., and Karniadakis, G. E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

Ramamurti, R. and Sandberg, W. Simulation of flow about flapping airfoils using finite element incompressible flow solver. *AIAA J.*, 39(2):253–260, 2001.

Roich, D., Mokady, R., Bermano, A. H., and Cohen-Or, D. Pivotal tuning for latent-based editing of real images. *arXiv preprint arXiv:2106.05744*, 2021.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *Proc. ICML*, 2020.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw. Learn. Syst.*, 20(1):61–80, 2008.

Schwarzbach, C., Börner, R.-U., and Spitzer, K. Three-dimensional adaptive higher order finite element simulation for geo-electromagnetics—a marine csem example. *Geophys. J. Int.*, 187(1):63–74, 2011.

Seo, S., Meng, C., and Liu, Y. Physics-aware difference graph networks for sparsely-observed dynamics. In *Proc. ICLR*, 2020.

Sirignano, J. and Spiliopoulos, K. Dgm: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.*, 375:1339–1364, 2018.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020.

Tancik, M., Srinivasan, P. P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J. T., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. In *Proc. NeurIPS*, 2020.

Thürey, N., Weißenow, K., Prantl, L., and Hu, X. Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA J.*, 58(1):25–36, 2020.

Tompson, J., Schlachter, K., Sprechmann, P., and Perlin, K. Accelerating eulerian fluid simulation with convolutional networks. In *Proc. ICML*, 2017.

Um, K., Brand, R., Holl, P., Thuerey, N., et al. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. In *Proc. NeurIPS*, 2020.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. 2018.

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

Virieux, J., Asnaashari, A., Brossier, R., Métivier, L., Ribodetti, A., and Zhou, W. *6. An introduction to full waveform inversion*, pp. R1–1–R1–40. 2017.

Wandel, N., Weinmann, M., and Klein, R. Learning incompressible fluid dynamics from scratch–towards fast, differentiable fluid models that generalize. 2021.

Wang, S. and Perdikaris, P. Long-time integration of parametric evolution equations with physics-informed deeponets. *CoRR*, abs/2106.05384, 2021.

Wang, S., Wang, H., and Perdikaris, P. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Sci. Adv.*, 7(40):eabi8605, 2021.

Wu, Y. and McMechan, G. A. Parametric convolutional neural network-domain full-waveform inversion. *Geophysics*, 84(6):R881–R896, 2019.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.*, 32(1):4–24, 2020.

Xue, T., Beatson, A., Adriaenssens, S., and Adams, R. Amortized finite element analysis for fast pde-constrained optimization. In *Proc. ICML*, 2020.

Yang, Y., Gao, A. F., Castellanos, J. C., Ross, Z. E., Azizzadenesheli, K., and Clayton, R. W. Seismic Wave Propagation and Inversion with Neural Operators. *TSR*, 1 (3):126–134, 11 2021.