Multiscale Model. Simul.
Vol. 22, No. 4, pp. 1421–1448

A MULTIFIDELITY MACHINE LEARNING BASED SEMI-LAGRANGIAN FINITE VOLUME SCHEME FOR LINEAR TRANSPORT EQUATIONS AND THE NONLINEAR VLASOV–POISSON SYSTEM[*]

YONGSHENG CHEN[†], WEI GUO[‡], AND XINGHUI ZHONG[§]

Abstract. Machine-learning (ML) based discretization has been developed to simulate complex partial differential equations (PDEs) with tremendous success across various fields. These learned PDE solvers can effectively resolve the underlying solution structures of interest and achieve a level of accuracy which often requires an order-of-magnitude finer grid for a conventional numerical method using polynomial-based approximations. In a previous work [13], we introduced a learned finite volume discretization that further incorporates the semi-Lagrangian (SL) mechanism, enabling larger CFL numbers for stability. However, the efficiency and effectiveness of such a methodology heavily rely on the availability of abundant high-resolution training data, which can be prohibitively expensive to obtain. To address this challenge, in this paper, we propose a novel multifidelity ML-based SL method for transport equations. This method leverages a combination of a small amount of high-fidelity data and sufficient but cheaper low-fidelity data. The approach is designed based on a composite convolutional neural network architecture that explores the inherent correlation between high-fidelity and low-fidelity data. The proposed method demonstrates the capability to achieve a reasonable level of accuracy, particularly in scenarios where a single-fidelity model fails to generalize effectively. We further extend the method to the nonlinear Vlasov–Poisson system by employing high-order Runge–Kutta exponential integrators. A collection of numerical tests are provided to validate the efficiency and accuracy of the proposed method.

Key words. semi-Lagrangian, machine learning, convolutional neural network, multifidelity, Vlasov–Poisson system

MSC code. 65M08

DOI. 10.1137/23M1598039

1. Introduction.

The rapid development of machine learning (ML) has opened new research avenues for approximating complex partial differential equations (PDEs), and many successful ML-based PDE solvers are designed by leveraging the expressive power of neural networks (NNs) and advancements of automatic differentiation technology [38]. Among these developments, ML-based discretization has received substantial research attention. This approach combines the strengths of ML techniques and conventional numerical methods, leading to remarkable success across diverse scientific and engineering applications [45, 54, 46, 59, 57]. Such equation-specific ML-based discretization replaces the polynomial-based approximation with NNs, yielding more flexible and accurate representations of the underlying PDEs

[*]Received by the editors September 11, 2023; accepted for publication (in revised form) July 12, 2024; published electronically October 16, 2024.
https://doi.org/10.1137/23M1598039

Funding: The research work of the second author is partially supported by the NSF grant NSF-DMS-2111383, Air Force Office of Scientific Research FA9550-18-1-0257. The research work of the third author is partially supported by the NSFC grant 12272347.

[†]School of Mathematical Sciences, Zhejiang University, Hangzhou, 310027, China (22035024@zju.edu.cn).

[‡]Corresponding author. Department of Mathematics and Statistics, Texas Tech University, Lubbock, TX 70409 USA (Weimath.guo@ttu.edu).

[§]School of Mathematical Sciences, Zhejiang University, Hangzhou, 310027, China (zhongxh@zju.edu.cn).

[46, 59]. With high-fidelity data, optimal numerical discretization can be learned through the training process of NNs. This enables such ML-based discretization to achieve accurate and stable numerical solutions even with much coarser grid resolutions [3, 64, 31], reducing computational cost compared to traditional numerical methods that require a finer grid for comparable accuracy. Other approaches of ML-based PDE solvers include the physics-informed neural networks (PINNs), which utilize the physics-informed loss [61, 62, 41, 52, 51], neural operators [39, 34, 30, 35, 5, 58], autoregressive methods [3, 24, 27, 6, 14], etc. Compared to these methods, ML-based discretization offers an additional advantage: the ability to enforce inherent physical constraints, such as conservation of mass, momentum, and energy of the system, at the discrete level. The integration of such inductive biases plays a pivotal role in enhancing the generalization capabilities of the NN [3, 64, 31].

In our recent work [13], we proposed an autoregressive ML finite volume (FV) method under the ML-based discretization framework. This method couples with the semi-Lagrangian (SL) mechanism for solving linear transport equations. By incorporating a specific inductive bias into the NN, the method aims to learn the SL discretization from the data, avoiding the need for costly upstream cells tracking. The proposed method inherits all the advantages of the ML-based discretization approach. Additionally, it allows for a larger Courant–Friedrichs–Lewy (CFL) number for stability compared to the Eulerian method-of-line approach, thereby enhancing the efficiency. However, the success of this method heavily relies on the availability of sufficient high-fidelity data, which can be prohibitively expensive to generate, especially for complex PDE simulations. This is known as a primary challenge for ML-based methods. While high-fidelity data is ideal for training the NN, it is often difficult to acquire and scarce in quantity. On the other hand, the low-fidelity data is easily obtainable but is insufficient for training ML models with satisfactory accuracy. Therefore, using a single-fidelity model is prone to generalization failure. To address this limitation, it becomes necessary to leverage data with multiple fidelity levels for multifidelity modeling. This approach can enhance both the accuracy and generalization capability of the model [21, 47].

In the literature, multifidelity modeling with NNs can generally be categorized into three approaches. The first approach focuses on learning the correlation between the low-fidelity and high-fidelity data. For instance, it has been employed to approximate the linear or nonlinear correlations between low-fidelity and high-fidelity solutions for PINNs [37, 43, 53] and for operator learning using the DeepONet approach [40, 19, 26]. Multifidelity Bayesian NNs [42], which integrate the Bayesian framework and PINNs, provide an example of capturing the cross-correlation with uncertainty quantification between the low- and high-fidelity data. Recently, the authors in [10] proposed a convolutional NN (CNN) architecture to exploit the correlation among the multifidelity data. More recently, the authors in [11] proposed a multifidelity PINN, where the low- and high-fidelity solutions are projected onto the same feature space using an encoder, and their projections are adjacent by constraining their relative distance with the NN. The second approach in multifidelity learning involves utilizing transfer learning [60]. In this approach, the NN is initially trained with low-fidelity information and subsequently fine-tuned with high-fidelity information, aiming to improve the overall accuracy of the model. Several related works on PDE solving tasks with transfer learning can be found in [1, 2, 9, 12, 17, 18, 29, 55, 44, 36]. The third approach involves learning low-fidelity and high-fidelity solutions using the same NN. This approach can be achieved by using the low-fidelity solution as an intermediate value of the network [25] or by imposing constraints on the solution with any available high-fidelity data [4].

In this paper, we propose a novel multifidelity ML-based method for the ML-based SL FV method solving transport equations, with the focus on scenarios where there is an abundance of low-fidelity data and limited high-fidelity data. Our method belongs to the first category and aims to take advantage of the accuracy of the high-fidelity data and the accessibility of the low-fidelity data simultaneously. To achieve this, we introduce a composite NN architecture that consists of two subnetworks: the low-fidelity network and the high-fidelity network. The low-fidelity network shares the same structure as the ML-based SL FV scheme presented in [13], and it is intended to predict low-fidelity solutions. The high-fidelity network also has a similar structure but incorporates the output of the low-fidelity network as an additional input to approximate the correlation between the low-fidelity and high-fidelity solutions. By integrating the two networks, our method achieves improved stability and accuracy compared to using networks trained solely on low-fidelity or high-fidelity data. It also exhibits satisfactory generalization capabilities. Moreover, our method inherits the advantages of the ML-based SL FV scheme [13], such as mass conservation, translational equivariance, avoiding the need to track upstream cells, the ability to allow for large CFL numbers, and attaining an accuracy level that exceeds that of traditional numerical algorithms with the same mesh resolution.

In addition to our proposed multifidelity ML-based SL FV method for solving linear transport equations, we also propose to extend this method for simulating the nonlinear Vlasov–Poisson (VP) system. The VP system presents an additional challenge for accurate tracking of its characteristic equations due to the inherent nonlinearity. To overcome this challenge, we combine the high-order Runge–Kutta (RK) exponential integrators (RKEI), introduced in [8, 7]. By employing the RKEI, the VP system can be decomposed into a sequence of linearized transport equations with frozen coefficients [7, 63]. This decomposition allows us to apply the multifidelity model, resulting in a nonlinear data-driven multifidelity SL FV scheme for the VP system.

The rest of the paper is organized as follows. In section 2, we review the ML-based SL FV scheme proposed in [13]. Section 3 is devoted to presenting our multifidelity data-driven SL FV scheme. In particular, we lay out the details for linear transport equations in section 3.1 and discuss its extension to the nonlinear VP system by coupling with the RKEI method in section 3.2. In section 4, numerical results are provided to demonstrate the performance of our multifidelity method. The conclusion and future work are discussed in section 5.

2. ML-based SL FV scheme.

In this section, we provide a brief overview of the ML-based SL method for solving linear transport equations developed in [13]. The methodology will serve as a building block of the proposed multifidelity solver discussed in section 3.

Consider the following one-dimensional (1D) transport equation

$$u_t + (a(x,t)u)_x = 0, \quad x \in \Omega, \tag{2.1}$$

where $a(x,t)$ denotes the velocity field. Here, the domain Ω is a bounded interval. Periodic boundary conditions are imposed for simplicity, but the proposed method is also capable of handling other types of boundary conditions, as shown in section 4. Ω is uniformly partitioned into N cells, where each cell is represented by $I_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$, and the mesh size is denoted by $h = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$. In an SL FV scheme, the solution is represented by the cell-averaged values U_i^m at time t^m in the cell I_i. The scheme then advances $\{U_i^m\}$ using characteristic information. Specifically, the scheme traces the characteristics backward in time from time step t^{m+1} to t^m by

$$\begin{cases} \dfrac{dx(t)}{dt} &= a(x(t),t), \\ x(t^{m+1}) &= x_{i+\frac{1}{2}}, \ i=1,\dots,N, \end{cases} \tag{2.2}$$

to obtain the endpoints of the upstream cell, denoted by $\tilde{x}^m_{i+\frac{1}{2}} = x(t^m)$, as shown in Figure 2.1. The advantage of integrating characteristic information in the algorithm design is that it allows for a larger CFL number and achieves higher accuracy and efficiency. Then $\{U_i^{m+1}\}$ can be updated based on the fact that the exact solution $u(x,t)$ satisfies

$$\int_{I_i} u\left(x,t^{m+1}\right) dx = \int_{\tilde{x}^m_{i-\frac{1}{2}}}^{\tilde{x}^m_{i+\frac{1}{2}}} u\left(x,t^m\right) dx, \ i=1,\dots,N,$$

and the right-hand side can be approximated by integrating the polynomials in $[\tilde{x}^m_{i-\frac{1}{2}}, \tilde{x}^m_{i+\frac{1}{2}}]$ reconstructed under the standard FV framework. With the normalized shift defined by

$$\xi^m_{i+\frac{1}{2}} = \frac{\tilde{x}^m_{i+\frac{1}{2}} - x_{i+\frac{1}{2}}}{h}, \quad i=1,\dots,N,$$

the SL FV schemes can be rewritten as the following formulation:

$$U_i^{m+1} = \sum_{\ell \in \mathcal{S}_i^m} d^m_{i,\ell} U^m_\ell, \tag{2.3}$$

where $\mathcal{S}_i^m$ denotes the stencil used for updating U_i^{m+1}, and $\{d^m_{i,\ell}\}$ are the associated coefficients fully determined by $\{U_i^m\}$ and $\{\xi_i^m\}$. For example, by employing quadratic polynomial reconstruction and assuming $\xi^m_{i\pm\frac{1}{2}} \in [-1,0]$, U_i^{m+1} is given by

$$\begin{aligned} U_i^{m+1} = &\left(\frac{1}{6}\xi^m_{i-\frac{1}{2}} - \frac{1}{6}\left(\xi^m_{i-\frac{1}{2}}\right)^3\right) U^m_{i-2} \\ &+ \left(-\frac{5}{6}\xi^m_{i-\frac{1}{2}} + \frac{1}{2}\left(\xi^m_{i-\frac{1}{2}}\right)^2 + \frac{1}{3}\left(\xi^m_{i-\frac{1}{2}}\right)^3 - \frac{1}{6}\xi^m_{i+\frac{1}{2}} + \frac{1}{6}\left(\xi^m_{i+\frac{1}{2}}\right)^3\right) U^m_{i-1} \\ &+ \left(1 - \frac{1}{3}\xi^m_{i-\frac{1}{2}} - \frac{1}{2}\left(\xi^m_{i-\frac{1}{2}}\right)^2 - \frac{1}{6}\left(\xi^m_{i-\frac{1}{2}}\right)^3 + \frac{5}{6}\xi^m_{i+\frac{1}{2}} - \frac{1}{2}\left(\xi^m_{i+\frac{1}{2}}\right)^2 - \frac{1}{3}\left(\xi^m_{i+\frac{1}{2}}\right)^3\right) U^m_i \\ &+ \left(\frac{1}{3}\xi^m_{i+\frac{1}{2}} + \frac{1}{2}\left(\xi^m_{i+\frac{1}{2}}\right)^2 + \frac{1}{6}\left(\xi^m_{i+\frac{1}{2}}\right)^3\right) U^m_{i+1}, \end{aligned}$$

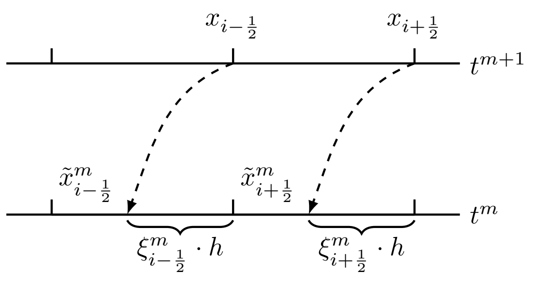

$x_{i-\frac{1}{2}}$
$x_{i+\frac{1}{2}}$
t^{m+1}
$\tilde{x}^m_{i-\frac{1}{2}}$
$\tilde{x}^m_{i+\frac{1}{2}}$
t^m
$\xi^m_{i-\frac{1}{2}} \cdot h$
$\xi^m_{i+\frac{1}{2}} \cdot h$

FIG. 2.1. *Schematic illustration of the* 1D *SL FV scheme.*

which is third-order accurate. If nonlinear reconstruction is utilized, the coefficients may also depend on $\{U_i^m\}$. More details can be found in [49]. Moreover, the scheme (2.3) is provably mass conservative when

$$\sum_i d_{i,\ell}^m = 1 \quad \forall \ell \tag{2.4}$$

(see, e.g., [32, 20, 13]).

It is worth emphasizing that in general the computation of $\{d_{i,\ell}^m\}$ requires expensive tracking of the geometries for upstream cells. This procedure not only demands sophisticated implementation but also significantly dominates the overall computational cost, especially in high dimensions [32, 20]. To address this challenge, we proposed an ML-based SL FV scheme in [13], for which the discretization coefficients $\{d_{i,\ell}^m\}$ are learned from the data, drawing inspiration from related works [3, 64, 31]. In particular, the proposed scheme incorporates the normalized shifts $\{\xi_i^m\}$ as an inductive bias and uses the NN $f_{\boldsymbol{W}}$ to infer the discretization coefficients:

$$\boldsymbol{d}^m = f_{\boldsymbol{W}}(\boldsymbol{U}^m, \boldsymbol{\xi}^m), \tag{2.5}$$

where $\boldsymbol{U}^m$, $\boldsymbol{\xi}^m$, and $\boldsymbol{d}^m$ denote the collection of $\{U_i^m\}$, $\{\xi_{i-\frac{1}{2}}^m\}$, and $\{d_{i,\ell}^m\}$, respectively. The NN $f_{\boldsymbol{W}}$ takes $\boldsymbol{U}^m$ and $\boldsymbol{\xi}^m$ as two-channel inputs and is constructed as a stack of convolutional layers with trainable parameters $\boldsymbol{W}$ and nonlinear activation functions, such as ReLU. In addition, periodic padding is applied to incorporate periodic boundary conditions. A constraint layer is incorporated to enforce (2.4), ensuring exact mass conservation. Once $\boldsymbol{d}^m$ is obtained, the solution $\boldsymbol{U}^{m+1}$ is updated with (2.3). The scheme is schematically illustrated in Figure 2.2. Note that CNNs rely on a shared-weight architecture of the convolution filters to capture the characteristics of the solution structures, necessitating a uniform mesh discretization. In addition, the ML-based scheme has limitations for the mesh refinement as opposed to the standard FV methods. However, thanks to the underlying CNN architecture, the scheme is able to be generalized to larger computational domains by keeping the same mesh size.

Unlike the traditional SL FV schemes, this ML-based scheme unitizes a set of fixed centered stencils. For example, we can choose the 5-cell stencil $\mathcal{S}_i^m = \{i-2, i-1, i, i+1, i+2\}$ to update the solution U_i^{m+1}. In this case, $\boldsymbol{d}^m$, $\boldsymbol{U}^m$, and $\boldsymbol{\xi}^m$ are 2D tensors of dimensions $N \times 5$, $N \times 1$, and $N \times 1$, respectively. In this way, this approach offers the benefits of simplifying algorithm development and conveniently satisfying the mass conservation constraint, though it compromises the unconditional stability of a traditional SL FV scheme, which relies on local stencils for reconstruction. Numerical evidence demonstrates that the ML-based SL FV scheme

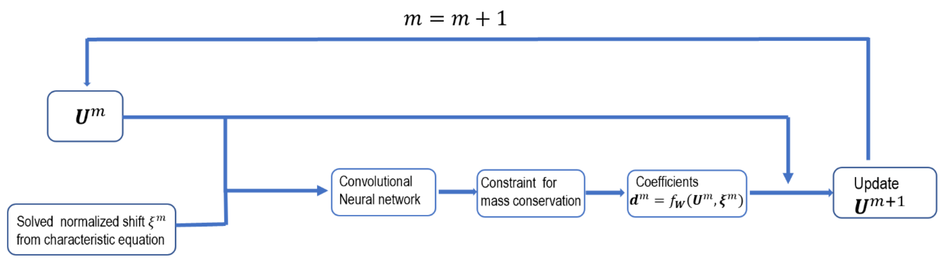

$m = m + 1$
$\boldsymbol{U}^m$
Solved normalized shift ξ^m from characteristic equation
Convolutional Neural network
Constraint for mass conservation
Coefficients $\boldsymbol{d}^m = f_W(\boldsymbol{U}^m, \boldsymbol{\xi}^m)$
Update $\boldsymbol{U}^{m+1}$

FIG. 2.2. *Illustration of the proposed ML-based SL FV method in* [13].

can maintain stability for CFL numbers up to 1.8. In addition, the scheme features the translational equivariance, which is highly desirable for data efficiency and improved generalization capability.

The method can be easily adapted for 2D linear transport problems with slight modification. We consider the 2D linear transport equation with variable coefficients

$$u_t + \nabla \cdot (\boldsymbol{v}(x,y,t)u) = 0, \quad (x,y) \in \Omega, \tag{2.6}$$

where $\boldsymbol{v}$ denotes the velocity field $\boldsymbol{v}(x,y,t) = (a(x,y,t), b(x,y,t))$. The computational domain Ω is a bounded rectangle, and periodic boundary conditions are imposed in both x and y directions. The associated characteristic system writes

$$\begin{cases} \dfrac{dx(t)}{dt} = a(x(t),y(t),t), \\ \dfrac{dy(t)}{dt} = b(x(t),y(t),t). \end{cases} \tag{2.7}$$

Assume Ω is partitioned uniformly with a collection of rectangular cells, i.e., $\Omega = \bigcup_{i,j} I_{ij}$, where $I_{ij} = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}] \times [y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}]$. Denote by $h_x = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$ and by $h_y = y_{j+\frac{1}{2}} - y_{j-\frac{1}{2}}$ the mesh sizes in x and y directions, respectively. Similarly to the 1D case, a 2D SL FV scheme needs to evolve (2.7) backward in time from t^{m+1} to t^m to obtain the upstream cell $\tilde{I}_{ij}$ of I_{ij}, as shown in Figure 2.3. Denote the upstream point of each grid point $(x_{i-\frac{1}{2}}, y_{j-\frac{1}{2}})$ as $(\tilde{x}_{i-\frac{1}{2},j-\frac{1}{2}}, \tilde{y}_{i-\frac{1}{2},j-\frac{1}{2}})$. Then we define

$$\xi_{i-\frac{1}{2},j-\frac{1}{2}} = \frac{\tilde{x}_{i-\frac{1}{2},j-\frac{1}{2}} - x_{i-\frac{1}{2}}}{h_x}, \quad \eta_{i-\frac{1}{2},j-\frac{1}{2}} = \frac{\tilde{y}_{i-\frac{1}{2},j-\frac{1}{2}} - y_{j-\frac{1}{2}}}{h_y}$$

as the normalized shifts of $(x_{i-\frac{1}{2}}, y_{j-\frac{1}{2}})$ in the x and y directions, respectively. The traditional 2D SL FV scheme is then formulated with the following identity:

$$\iint_{I_{ij}} u(x,y,t^{m+1})dxdy = \iint_{\tilde{I}_{ij}} u(x,y,t^m)dxdy. \tag{2.8}$$

Denote by U_{ij}^m the cell average of the numerical solution in the cell I_{ij} at time step $t = t^m$. Similarly to the 1D case, to update the numerical solution, a polynomial is reconstructed over each cell using the cell averages of itself and the neighbors, and the integral on the right-hand side of (2.8) is computed in a subcell-by-subcell fashion. The scheme can be written as

$$U_{ij}^{m+1} = \sum_{\ell \in \mathcal{S}_{ij}^m} d_{ij,\ell}^m U_\ell^m, \tag{2.9}$$

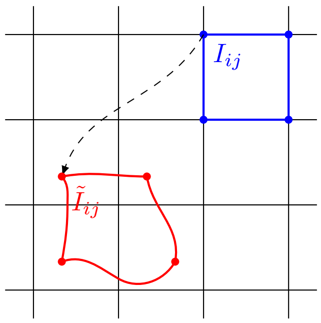


FIG. 2.3. *Schematic illustration of the 2D SL FV scheme.*

where $\mathcal{S}_{ij}^m$ denotes the stencil employed to update U_{ij}^{m+1}. Again, the coefficients $\{d_{ij,\ell}^m\}$ are determined by the solution averages $\{U_{ij}^m\}$ together with normalized shifts $\{\xi_{i-\frac{1}{2},j-\frac{1}{2}}^m\}$, $\{\eta_{i-\frac{1}{2},j-\frac{1}{2}}^m\}$. In addition, as with the 1D case, the scheme (2.9) is mass conservative if

$$\sum_{(i,j)\in\mathcal{E}_\ell^m} d_{ij,\ell}^m = 1 \quad \forall l, \tag{2.10}$$

where $\mathcal{E}_\ell^m = \{(i,j) : \ell \in \mathcal{S}_{ij}^m\}$. Motivated by the traditional SL FV scheme, in [13] we proposed a 2D ML-based SL FV scheme as follows:

$$\boldsymbol{d}^m = f_{\boldsymbol{W}}(\boldsymbol{U}^m, \boldsymbol{\xi}^m, \boldsymbol{\eta}^m), \tag{2.11}$$

where the tensors $\boldsymbol{U}^m, \boldsymbol{\xi}^m, \boldsymbol{\eta}^m$ are formed by collecting $\{U_{ij}^m\}$, $\{\xi_{i-\frac{1}{2},j-\frac{1}{2}}^m\}$, $\{\eta_{i-\frac{1}{2},j-\frac{1}{2}}^m\}$, respectively. $f_{\boldsymbol{W}}$ takes a 3-channel input and outputs the coefficient tensor $\boldsymbol{d}^m$, which is used to update the solution. The NN $f_{\boldsymbol{W}}$ is constructed by stacking a series of 2D convolutional layers, combined with a constraint layer to enforce the condition (2.10) for exact mass conservation. It is important to note that, similarly to the 1D case, we employ a set of squared fixed stencils. In particular, if a fixed 5×5-cell stencil and an $N \times N$ cell grid are used, the dimensions of $\boldsymbol{d}^m, \boldsymbol{U}^m, \boldsymbol{\xi}^m$, and $\boldsymbol{\eta}^m$ are tensors of size $N \times N \times 5 \times 5$, $N \times N \times 1$, $N \times N \times 1$, and $N \times N \times 1$, respectively. Again, due to the use of fixed stencils, the proposed ML-based SL formulation is constrained by the CFL condition.

The rationale behind the ML-based PDE discretization approach lies in the observation that the solution manifold often exhibits low-dimensional structures, including recurrent patterns and coherent structures. By leveraging high-resolution training data, the underlying NN effectively learns an optimal discretization that can achieve significantly higher accuracy compared with a traditional polynomial-based method. However, similarly to other ML-based discretization schemes, the effectiveness of this method heavily relies on the availability of abundant high-resolution training data. In particular, to generate the training data, we sample a set of initial conditions from a prescribed distribution. From each sampled initial condition, we employ an accurate and reliable FV method, such as an SL WENO method or Eulerian WENO method, to generate solution trajectories on a high-resolution mesh. It is crucial to adequately resolve the solution structures of interest during this process. Subsequently, we obtain the training data by downsampling these solution trajectories to a mesh with reduced resolution by an order of magnitude. It is worth noting that acquiring such training data can often be prohibitively expensive, especially for complex problems such as plasma simulations. Moreover, the scarcity of high-resolution data can significantly limit the performance of the scheme and impede its ability to generalize effectively.

3. Multifidelity data-driven SL FV scheme.

In this section, we present a novel data-driven multifidelity SL FV scheme aimed at reducing the reliance on high-resolution data for the ML-based SL FV method introduced in section 2, while maintaining a reasonable level of accuracy. We begin by introducing the scheme for transport equations in section 3.1 and subsequently extend the scheme to the nonlinear VP system in section 3.2.

3.1. Transport equations.

In this subsection, we lay out the details of the multifidelity approach for linear transport equation (2.1) with the goal of enhancing the accuracy and efficiency of the ML-based SL FV method while minimizing the reliance on high-resolution data.

The training dataset for our multifidelity approach is composed of both high-fidelity and low-fidelity data. In particular, it includes a set of N_L low-fidelity trajectories $\{\{\boldsymbol{U}_L^{(i),m}\}_{m=0}^{T_L}\}_{i=1}^{N_L}$ and N_H high-fidelity trajectories $\{\{\boldsymbol{U}_H^{(j),n}\}_{n=0}^{T_H}\}_{j=1}^{N_H}$, with N_L typically being greater than N_H. Here, $\boldsymbol{U}_L^{(i),m}$ represents the solution data in the ith low-fidelity trajectory at time step t^m, and $\boldsymbol{U}_H^{(j),n}$ represents the solution data in the jth high-fidelity trajectory at time step t^n. The high-fidelity data comprises coarsened high-resolution numerical solution trajectories, which are accurate but computationally expensive and scarce. On the other hand, the low-fidelity data consists of numerical solution trajectories computed by a numerical scheme on a low-resolution mesh. These low-fidelity trajectories are less accurate but more affordable compared to the high-fidelity ones. In addition, both the high-fidelity and low-fidelity solution trajectories are generated by independently sampling initial conditions from a prescribed distribution, and all trajectories share the same time steps and spatial mesh size. The key assumption behind the multifidelity architecture is that there exists a correlation between data of different fidelity levels. In particular, given two consecutive solutions $\{\boldsymbol{U}_L^m, \boldsymbol{U}_L^{m+1}\}$ and $\{\boldsymbol{U}_H^m, \boldsymbol{U}_H^{m+1}\}$ with $\boldsymbol{U}_L^m = \boldsymbol{U}_H^m$, we assume that the relation between $\boldsymbol{U}_L^{m+1}$ and $\boldsymbol{U}_H^{m+1}$ is given by

$$\boldsymbol{U}_H^{m+1} = g(\boldsymbol{U}_H^m, \boldsymbol{\xi}^m, \boldsymbol{U}_L^{m+1}), \tag{3.1}$$

where the function g takes the previous high-fidelity solution $\boldsymbol{U}_H^m$, the normalized shifts $\boldsymbol{\xi}^m$, and the low-fidelity prediction $\boldsymbol{U}_L^{m+1}$ as inputs and generates the high-fidelity solution $\boldsymbol{U}_H^{m+1}$ with enhanced accuracy. While it is feasible to train a single-fidelity model using abundant high-fidelity data, where the model directly takes $\boldsymbol{U}_H^m$ and $\boldsymbol{\xi}^m$ as inputs and computes the approximation $\boldsymbol{U}_H^{m+1}$ with high accuracy, in scenarios with limited availability of high-fidelity data, it is more effective to incorporate $\boldsymbol{U}_L^{m+1}$ and explore the inherent correlation with (3.1). Despite the low accuracy of the low-fidelity data $\boldsymbol{U}_L^{m+1}$, it can still provide valuable information and greatly accelerate the training process of the high-fidelity component in the model. By leveraging the inherent correlation between data of different fidelity levels, our multifidelity approach allows us to improve the accuracy of high-fidelity predictions even when high-fidelity data is limited. Moreover, by combining both high-fidelity and low-fidelity data in our training data set, we can benefit from the accuracy of the high-fidelity data and the accessibility of the low-fidelity data simultaneously. It enables us to develop a robust model that generalizes well across different fidelity levels and maximizes the utility of available data resources, leading to more efficient and accurate solutions in practice.

Our multifidelity data-driven SL FV scheme employs a composite architecture, drawing inspiration from previous works under various frameworks such as Gaussian process regression [50, 48], PINNs [43], and DeepONets [26]. Figure 3.1 provides a schematic diagram illustrating this architecture. It consists of two key components: the low-fidelity network, denoted as f_L, and the high-fidelity network, denoted as g_H. The low-fidelity network f_L follows the same structure as the ML-based SL FV scheme described in section 2, which takes a two-channel input $(\boldsymbol{U}^m, \boldsymbol{\xi}^m)$ and predicts the solution at the next time level, capturing the underlying trends in the data. Motivated by (3.1), the high-fidelity network g_H shares a similar structure to f_L, but it takes a three-channel input $(\boldsymbol{U}_H^m, \boldsymbol{\xi}^m, \boldsymbol{U}_{H,t}^{m+1})$, where $\boldsymbol{U}_{H,t}^{m+1}$ represents the intermediate solution computed by f_L using the input $(\boldsymbol{U}_H^m, \boldsymbol{\xi}^m)$. The objective of g_H is to approximate the correlation and generate an enhanced solution $\boldsymbol{U}_H^{m+1}$ with an improved accuracy compared to $\boldsymbol{U}_{H,t}^{m+1}$. By integrating both low-fidelity and high-fidelity components,

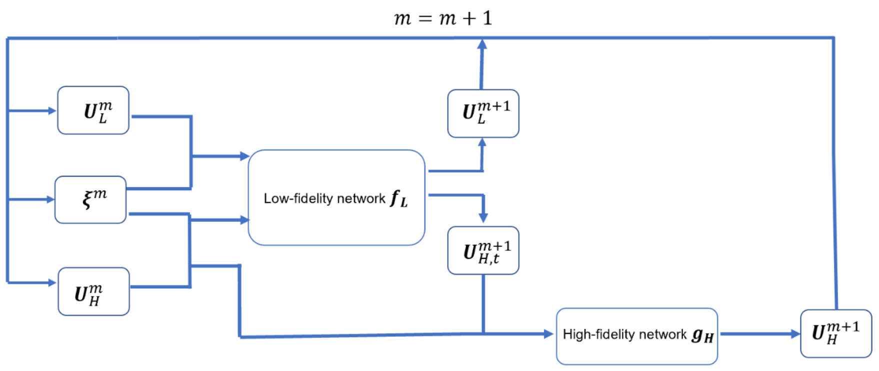


FIG. 3.1. *Illustration of the multifidelity architecture.*

our multifidelity architecture leverages the underlying trends captured by f_L and incorporates the correlation approximation from g_H. This integration aims to enhance the accuracy of the solution beyond what each component can achieve individually. The composite architecture effectively combines the strengths of both components, leading to improved accuracy and reliability in our data-driven SL FV scheme.

The training procedure for our multifidelity data-driven SL FV scheme involves the simultaneous training of the low-fidelity network f_L and the high-fidelity network g_H. The two networks, f_L and g_H, are trained by minimizing the one-step loss given by

$$\mathcal{L}_{MF}(f_L, g_H) = \lambda_1 \mathcal{L}_{LF} + \lambda_2 \mathcal{L}_{HF}, \tag{3.2}$$

where λ_1 and λ_2 are hyperparameters that balance low-fidelity loss $\mathcal{L}_{LF}$ and high-fidelity loss $\mathcal{L}_{HF}$. The low-fidelity loss is defined as

$$\mathcal{L}_{LF}(f_L) = \frac{1}{N_L}\frac{1}{T_L}\sum_{i=1}^{N_L}\sum_{m=0}^{T_L-1}\left| f_L(\boldsymbol{U}_L^{(i),m}, \boldsymbol{\xi}^m) - \boldsymbol{U}_L^{(i),m+1}\right|^2, \tag{3.3}$$

where $f_L(\boldsymbol{U}_L^{(i),m}, \boldsymbol{\xi}^m)$ represents the prediction of the low-fidelity network f_L for the next time step given the two-channel input $(\boldsymbol{U}_L^{(i),m}, \boldsymbol{\xi}^m)$. The high-fidelity loss is defined as

$$\mathcal{L}_{HF}(g_H) = \frac{1}{N_H}\frac{1}{T_H}\sum_{j=1}^{N_H}\sum_{n=0}^{T_H-1}\left| g_H\left(\boldsymbol{U}_H^{(j),n}, \boldsymbol{\xi}^n, \boldsymbol{U}_{H,t}^{(j),n+1}\right) - \boldsymbol{U}_H^{(j),n+1}\right|^2, \tag{3.4}$$

where $\boldsymbol{U}_{H,t}^{(j),n+1} = f_L(\boldsymbol{U}_H^{(j),n}, \boldsymbol{\xi}^n)$ represents the intermediate solution computed by the low-fidelity network f_L using the two-channel input $(\boldsymbol{U}_H^{(j),n}, \boldsymbol{\xi}^n)$, and $g_H(\boldsymbol{U}_H^{(j),n}, \boldsymbol{\xi}^n, \boldsymbol{U}_{H,t}^{(j),n+1})$ represents the prediction of the high-fidelity network g_H for the next time step given the three-channel input $(\boldsymbol{U}_H^{(j),n}, \boldsymbol{\xi}^n, \boldsymbol{U}_{H,t}^{(j),n+1})$. During training, the objective is to minimize the multifidelity loss by adjusting the parameters of both networks. The hyperparameters λ_1 and λ_2 control the relative importance of the low-fidelity and high-fidelity losses in the overall loss function. Although it is possible to first train the network f_L using the ML-based SL FV method discussed in section 2 and subsequently train the network g_H with fixed f_L by (3.2), simultaneous training of the NNs with multitask learning offers several benefits, including improved data efficiency and accelerated learning, as highlighted in [16]. It is worth noting that one-step training, as described above, may suffer slightly weaker generalization. On the other hand, unrolling the training over multiple time steps can improve the accuracy and stability at the cost of increased training difficulty, as discussed in [6]. In the conducted numerical experiments, one-step training is used for linear cases, while training is unrolled for 16 time steps when dealing with the nonlinear VP system.

For the implementation of the trained multifidelity SL FV scheme in the testing phase, the solutions are updated in time as follows: Starting from the solution $\boldsymbol{U}^m$ at time level m, where $\boldsymbol{U}^0$ is initialized as the cell averages of the given initial condition, the characteristic equation (2.2) is solved to obtain the normalized shifts $\boldsymbol{\xi}^m$. Next, using the trained low-fidelity network f_L, the intermediate solution $\boldsymbol{U}_t^{m+1}$ is computed by taking the two-channel inputs $(\boldsymbol{U}^m,\boldsymbol{\xi}^m)$, i.e., $\boldsymbol{U}_t^{m+1} = f_L(\boldsymbol{U}^m,\boldsymbol{\xi}^m)$. Finally, the solution $\boldsymbol{U}^{m+1}$ with enhanced accuracy is generated by feeding $\boldsymbol{U}^m$, $\boldsymbol{\xi}^m$, and $\boldsymbol{U}_t^{m+1}$ as inputs to the trained high-fidelity network g_H, resulting in $\boldsymbol{U}^{m+1} = g_H(\boldsymbol{U}^m,\boldsymbol{\xi}^m,\boldsymbol{U}_t^{m+1})$. It is numerically observed that the intermediate solution $\boldsymbol{U}_t^{m+1}$ achieves a higher level of accuracy compared to $\boldsymbol{U}_L^{m+1}$, which is obtained by the single-fidelity SL FV method trained solely with low-fidelity data. This difference in accuracy validates the effectiveness of the multifidelity approach and the accelerated learning process for the high-fidelity network g_H, which benefits from the more accurate intermediate solution as part of its inputs.

3.2. The Vlasov–Poisson system. In this subsection, we extend the proposed algorithm in section 3.1 for linear transport equations to the nonlinear VP system, to address the challenges posed by its inherent nonlinearity.

The VP system is a fundamental model in plasma physics that describes interactions between charged particles through self-consistent electric fields, modeled by Poisson's equations in the nonrelativistic zero-magnetic field limit. The dimensionless governing equations under a 1D in space and 1D in velocity (1D1V) setting is given by

$$f_t + vf_x + E(x,t)f_v = 0, \quad (x,v) \in \Omega_x \times \Omega_v, \tag{3.5}$$

$$E_x = \rho - 1, \quad \rho(x,t) = \int_{\Omega_v} f(x,v,t)dv, \tag{3.6}$$

where $f(x,v,t)$ is the probability distribution function of electrons at position x with velocity v at time t, and ρ denotes the density. Ω_x denotes the physical domain, while Ω_v represents the velocity domain. We consider periodic boundary conditions in the x-direction and zero boundary conditions in the v-direction. It is worth mentioning that in practice Ω_v is truncated to be finite and is taken large enough so that the solution $f \approx 0$ at $\partial\Omega_v$.

In addition, we assume a uniform background of fixed ions under a self-consistent electrostatic field E.

Unlike the linear transport equation (2.1), the Vlasov equation (3.5) is a nonlinear transport equation, which introduces additional challenges for accurately approximating its characteristic equations

$$\begin{cases} \dfrac{dx(t)}{dt} = v(t), \\ \dfrac{dv(t)}{dt} = E(x(t),t). \end{cases} \tag{3.7}$$

Consequently, designing an SL scheme becomes more complex for the VP system. To address these challenges, the splitting approach has gained popularity in the literature [15, 56]. This approach decouples the VP system into several linear transport equations, which are much easier to solve in the SL framework. However, it is important to note that the inherent splitting error can potentially compromise the overall accuracy of the solution. Recently, a nonsplitting SL methodology has been introduced for the VP system, utilizing the commutator-free RKEI [8]. By employing RKEI, the VP system is decomposed into a sequence of linearized transport equations with frozen coefficients [7, 63], which can be effectively solved using the proposed multifidelity ML-based SL FV scheme. Our novel approach combines the advantages of RKEI and multifidelity ML-based SL FV schemes, thereby avoiding the splitting errors inherent in traditional splitting approaches and enabling efficient and accurate solutions to the VP system.

To introduce the algorithm, we begin by considering a uniform partition of the domain $\Omega_x \times \Omega_v$ with $N_x \times N_v$ cells, i.e., $\Omega_x \times \Omega_v = \bigcup_{ij} I_{ij}$. The mesh sizes in the x and v directions are represented by h_x and h_v, respectively. Let f_{ij}^m denote the numerical approximation of the cell average of f in the cell I_{ij} at time level t^m and $E_{i+\frac{1}{2}}^m$ be the numerical solution of E at the endpoint of the cell in the x direction. The collections $\{E_{i-\frac{1}{2}}^m\}$ and $\{f_{ij}^m\}$ are denoted by $\boldsymbol{E}^m$ and $\boldsymbol{F}^m$, respectively. Within the FV framework, the electric field $E(x)$ can be straightforwardly approximated from Poisson's equation (3.6) as

$$E_{i+\frac{1}{2}}^m = E_{-\frac{1}{2}}^m + h_x \sum_{l=1}^{i} \left(h_v \sum_{j=1}^{N_v} f_{ij}^m - 1 \right), \quad i = 1, \dots, N_x. \tag{3.8}$$

Here, $E_{-\frac{1}{2}}^m$ is determined by the given boundary conditions. For instance, in case of periodic boundary conditions, we have $E_{-\frac{1}{2}}^m = 0$.

Table 1
First-order RKEI.

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

Table 2
Second-order RKEI.

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array}$$

Our approach employs the RKEI technique to alleviate the difficulties associated with accurately tracking the characteristics for the nonlinear VP system. The RKEI method is represented by a Butcher tableau, which specifies the coefficients for the integration steps. The accuracy of the method is determined by order conditions. The simplest RKEI method is given by the Butcher tableau 1, which is first-order accurate [8]. With this first-order RKEI, we can develop a ML-based SL FV scheme for the VP system as follows.

(1) Compute $\boldsymbol{E}^m$ using (3.8) based on the cell average approximations $\boldsymbol{F}^m$.
(2) Linearize the Vlasov equation with the fixed electric field $\boldsymbol{E}^m$ and solve the characteristic equations (3.7) to obtain the normalized shifts.
(3) Use the ML-based SL FV method, together with $\boldsymbol{F}^m$ and the normalized shifts, to predict $\boldsymbol{F}^{m+1}$.

The procedure can be summarized as

$$\boldsymbol{F}^{m+1} = MLSL(\boldsymbol{E}^m, \Delta t)\boldsymbol{F}^m, \tag{3.9}$$

where $MLSL(\boldsymbol{E}^m, \Delta t)$ denotes the ML-based SL FV evolution operator for the Vlasov equation with the fixed electric field $\boldsymbol{E}^m$ and time step Δt. However, the low-order temporal accuracy of the first-order RKEI can limit its generalization capability.

In the following, we enhance the performance by employing a second-order RKEI [8], represented by the Butcher tableau 2. The corresponding ML-based SL FV algorithm for the VP system can be summarized as

$$\begin{cases} \boldsymbol{F}^{m,*} = MLSL(\boldsymbol{E}^m, \frac{1}{2}\Delta t)\boldsymbol{F}^m, \\ \boldsymbol{F}^{m+1} = MLSL(\boldsymbol{E}^{m,*}, \Delta t)\boldsymbol{F}^m, \end{cases} \tag{3.10}$$

where $\boldsymbol{E}^m$ and $\boldsymbol{E}^{m,*}$ are determined via (3.8) by $\boldsymbol{F}^m$ and $\boldsymbol{F}^{m,*}$, respectively. The second-order scheme involves an intermediate stage $\boldsymbol{F}^{m,*}$ and requires two applications of the ML-based SL FV evolution operator. It is numerically demonstrated that this second-order method (3.10) exhibits improved generalization capabilities and achieves a higher level of accuracy compared to the first counterpart (3.9). By combining the proposed multifidelity framework with the ML-based SL approach, we can effectively learn an optimal discretization for the VP system, while significantly reducing the dependence on high-fidelity data. Moreover, all the desired properties of the method for solving linear equations, such as mass conservation and translational equivariance, are preserved for the nonlinear VP system. The learned multifidelity model achieves a level of accuracy that surpasses traditional numerical algorithms with comparable mesh resolution, leading to significant computational savings.

4. Numerical results.

In this section, we carry out a series of numerical experiments to demonstrate the performance of the proposed multifidelity ML-assisted SL FV scheme for various benchmark 1D and 2D linear transport equations, as well as 1D1V nonlinear VP systems. Note that the performance of the proposed scheme is significantly influenced by the choice of hyperparameters for the NN structure. Here, we present numerical results using the following default settings. For the linear equations, we utilize 6 convolutional layers with 32 filters each, and a kernel size of 5 for both 1D and 2D cases. For the nonlinear VP systems, we employ 9 convolutional layers with 32 filters each, using a kernel size of 5. In addition, to ensure mass conservation, a constraint layer is added. Throughout the training process, we employ the ReLU function as the activation function and use the Adam optimization algorithm [38].

We are allowed to employ any accurate and stable FV schemes to generate both high-fidelity and low-fidelity data. Specifically, for linear transport equations, we adopt the Eulerian fifth-order FV WENO (WENO5) method [28], coupled with the third-order strong-stability-preserving RK time integrator [23]. For nonlinear VP systems, we employ the conservative SL FV WENO5 scheme coupled with a fourth-order RKEI [63]. The high-fidelity training data is obtained by coarsening high-resolution solution trajectories into a low-resolution mesh. The low-fidelity training data is generated by directly running the transport scheme on the low-resolution mesh. In all test examples, we primarily report the results of the proposed multifidelity learning approach and compare these results with the single-fidelity model developed in section 2 and the traditional WENO5 scheme. The plots presented below indicate the performance of each approach: "multifidelity" refers to the proposed method, "high-fidelity" denotes the single-fidelity ML-based SL FV model trained solely with high-fidelity data, and "low-fidelity" denotes the single-fidelity ML-based SL FV model trained solely with low-fidelity data.

4.1. Transport equations. In this subsection, we present numerical results for simulating several 1D transport equations.

Example 4.1. In this example, we consider the following advection equation with a constant coefficient,

$$u_t + u_x = 0, \quad x \in [0, 1], \tag{4.1}$$

and periodic conditions are imposed.

We generate 30 low-resolution solution trajectories with a 32-cell grid as low-fidelity training data. Each trajectory comprises 54 sequential time steps. For high-fidelity training data, we downsample 15 high-resolution solution trajectories computed with a 256-cell grid by a factor of 8. Each trajectory consists of 15 sequential time steps. Moreover, all trajectories, both low- and high-fidelity, are initialized as a square wave with the height randomly sampled from the interval $[0.1, 1]$ and the width randomly selected from $[0.2, 0.4]$. We use a CFL number of 1.8 for determining the time step and a centered 5-cell stencil for updating the solution. During the multifidelity model training, the parameters λ_1 and λ_2 are set as 0.1 and 1, respectively. To test, we randomly generate square functions with widths and heights within the same range as initial conditions. For comparison, we generate a reference solution following the same approach as the high-fidelity data creation.

Figure 4.1 displays three test samples obtained from forward integration at different time instances, computed by the proposed multifidelity model, traditional WENO5 scheme, as well as by single-fidelity models trained solely with high-fidelity or low-fidelity data. It is observed that the low-fidelity model exhibits significant smearing near discontinuities due to the inherent low accuracy of the training data. Meanwhile, the high-fidelity model generates highly inaccurate results due to the lack of sufficient high-fidelity data. In addition, the WENO5 method exhibits significant smearing near discontinuities, which deteriorates over time as a result of the accumulation of numerical diffusion. In contrast, our proposed multifidelity approach demonstrates superior shock resolution, with sharp shock transitions and no spurious oscillations. Notably, even after a long-time simulation of 256 time steps, which is about 17 times the number of time steps used for generating high-fidelity training data, our multifidelity method remains remarkably close to the reference solution. This validates the high stability and accuracy of the learned discretization.

Figure 4.2 shows the intermediate solution of our multifidelity model for the three test samples, obtained from the trained low-fidelity network f_L of the proposed multifidelity method. It is observed that the intermediate solution is significantly more accurate compared to the results obtained by the low-fidelity model alone. This indicates that the simultaneous training of the multifidelity model is beneficial for accelerating the learning process of the high-fidelity component. Figure 4.3(a) illustrates the training loss of the three models throughout their respective training epochs.

Example 4.2. In this example, we consider the advection equation (4.1) with a more complicated solution profile consisting of triangle and square waves.

Similarly to the previous example, we generate 30 low-resolution solution trajectories using a 32-cell grid as the low-fidelity training data. Each trajectory consists of 54 sequential time steps. To create high-fidelity training data, we downsample 15 trajectories of high-resolution solutions computed with a 256-cell grid by a factor of 8. Each trajectory contains 15 sequential time steps. For both low- and high-fidelity trajectories, the initial conditions include one triangle centered at 0.25 and one square

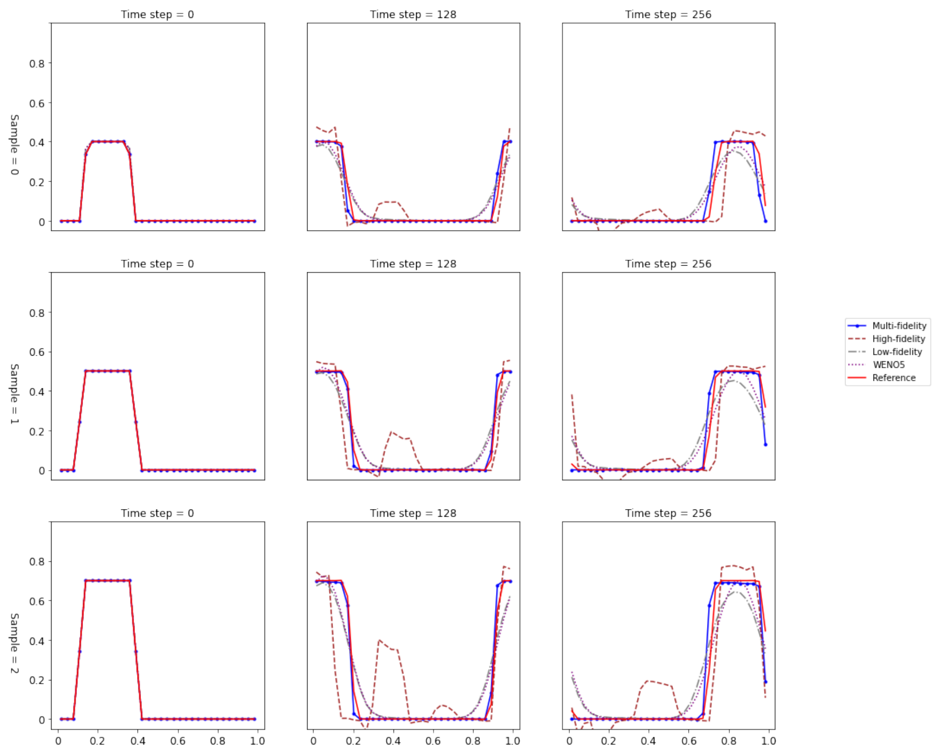


FIG. 4.1. *Three test samples for square waves in Example* 4.1.

wave centered at 0.75. The heights and widths of these shapes are randomly sampled from the ranges [0.2, 0.8] and [0.2, 0.3], respectively. We use a fixed stencil size of 3 and set the CFL number to 1.8. The parameters for training are chosen as $\lambda_1 = 0.05$ and $\lambda_2 = 1$. For testing, the initial conditions are randomly generated from the same distribution used to create the training data.

In Figure 4.4, we plot two test samples at several time instances during forward integration. It is observed that the proposed multifidelity solver generates numerical results with significantly higher resolution of the underlying nonsmooth structures compared to the low-fidelity model. Moreover, the high-fidelity model exhibits severe spurious oscillations and eventually blows up. Figure 4.5 presents the intermediate solution of our multifidelity method for the two test samples. Similarly to the previous example, we observe that the intermediate solution of our multifidelity method is much more accurate compared to the results obtained by the low-fidelity model. The training loss for three different methods during their respective training is shown in Figure 4.3(b).

Example 4.3. In this example, we consider the advection equation

$$u_t + u_x = 0, x \in [-\pi, \pi] \tag{4.2}$$

with initial conditions given in the form of a finite Fourier series as follows:

$$u(x,0) = a_0 + \sum_{n=1}^{N_c} (a_n \cos(nx) + b_n \sin(nx)). \tag{4.3}$$

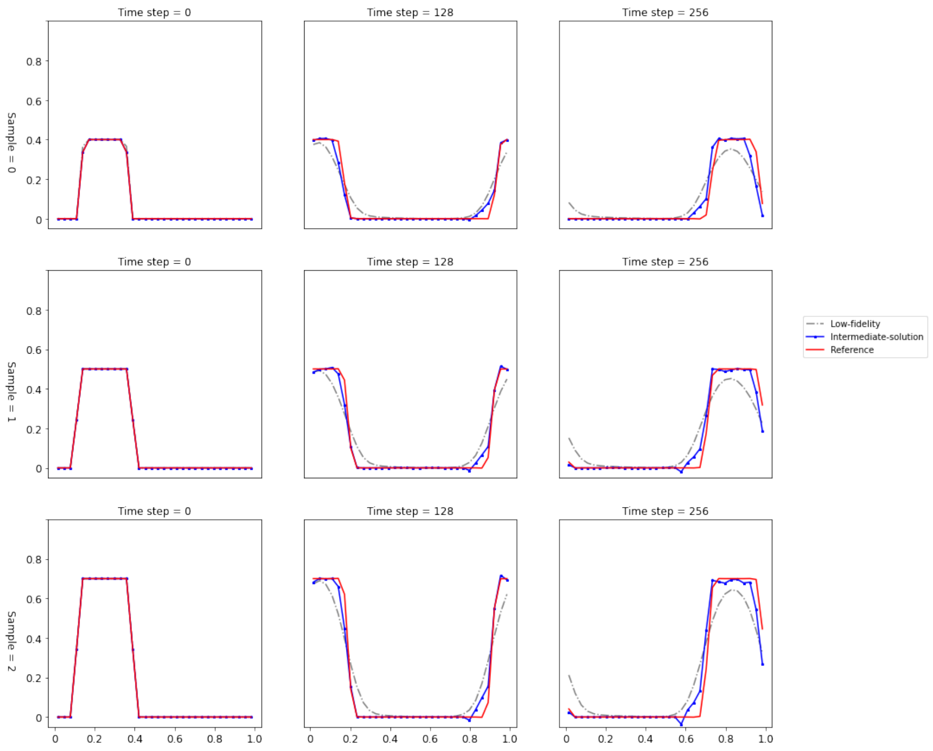
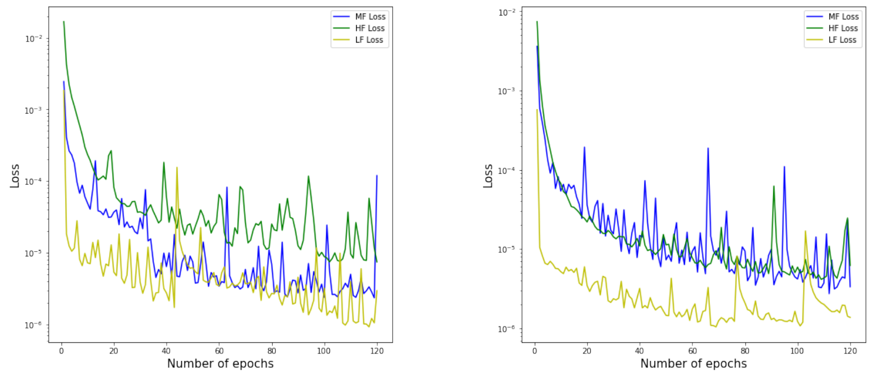


FIG. 4.2. *The intermediate solutions of three test samples for square waves in Example* 4.1.



(a) Example 4.1 (b) Example 4.2

FIG. 4.3. *The training loss for three different methods in Examples* 4.1 *(left) and* 4.2 *(right).*

We impose an inflow boundary condition at $x = -\pi$,

$$u(-\pi,t) = a_0 + \sum_{n=1}^{N_c} (a_n \cos(n\pi + nt) - b_n \sin(n\pi + nt)),$$

and an outflow boundary condition at $x = \pi$.

To create the training data, we generate 30 low-resolution solution trajectories with a 32-cell grid as the low-fidelity training data. Each trajectory comprises

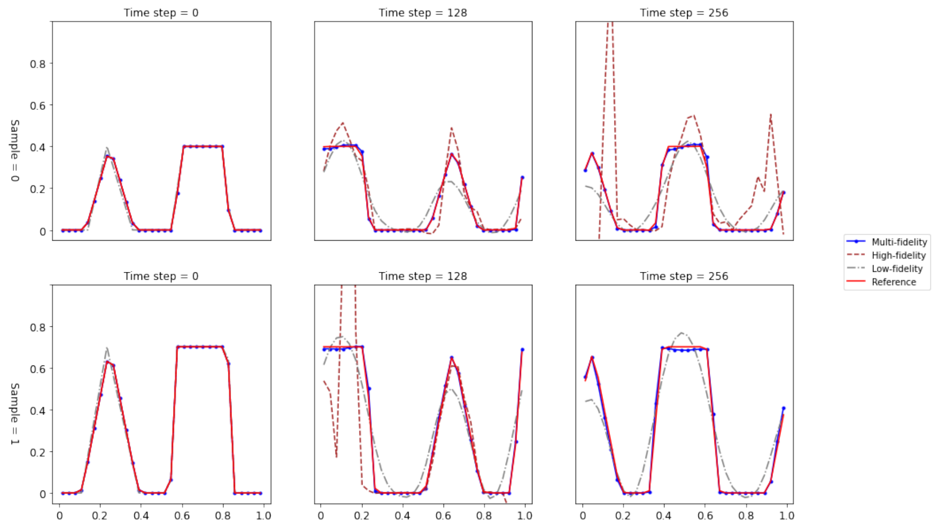


FIG. 4.4. *Numerical solutions of two test samples for advection of triangle and square waves in Example* 4.2. $CFL = 1.8$.

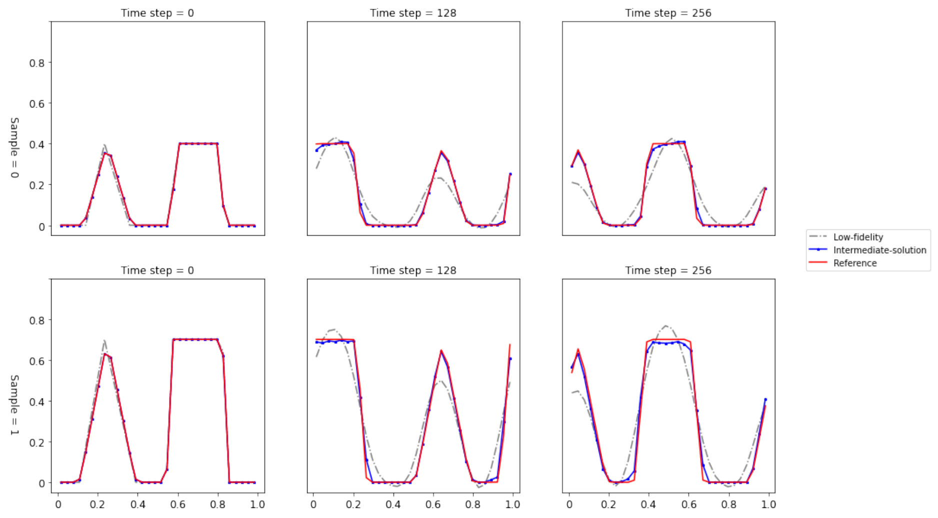


FIG. 4.5. *The intermediate solutions of two test samples for advection of triangle and square waves in Example* 4.2. $CFL = 1.8$.

30 sequential time steps. To create the high-fidelity training data, we downsample 15 trajectories of high-resolution solutions computed with a 256-cell grid by a factor of 8. Each trajectory contains 10 sequential time steps. For both the low- and high-fidelity trajectories, the initial conditions are randomly chosen with $N_c = 5$, $a_0 = 0$, and a_n, b_n, $(n \geq 1)$ from a uniform distribution over $[-1, 1]$. We set the stencil size to be 5, and the CFL number is 0.6. We choose the loss weights as $\lambda_1 = 0.1$ and $\lambda_2 = 1$. As a standard practice, the inflow-outflow boundary conditions are incorporated via ghost cells. In particular, at the inflow boundary, we create a few ghost cells (6 cells) by extending the solutions using the given boundary condition. Similarly, at

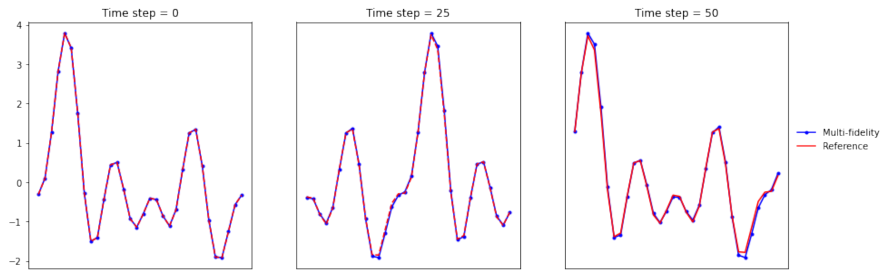


FIG. 4.6. *Numerical solutions of one test sample for advection of finite Fourier series in Example* 4.3.

the outflow boundary, the solutions are extended into the ghost cells, with each cell adopting the value of the rightmost cell in the computational domain.

In Figure 4.6, we present one test sample at several time instances during forward integration. It can be observed that the solution by our multifidelity model is remarkably close to the reference solution for this example with inflow and outflow boundary conditions, similarly to the scenario with periodic boundary conditions presented in previous examples.

Example 4.4. In this example, we simulate the following 1D advection equation with a variable coefficient

$$u_t + (u \sin(x+t))_x = 0, \quad x \in [0, 2\pi], \tag{4.4}$$

subject to periodic conditions. This example is more challenging than the previous two examples. In addition to the pure shift, the solution profiles will gradually deform over time and exhibit more complicated structures.

To generate low-fidelity training data, we produce 90 solution trajectories on a 32-cell grid, with each trajectory consisting of 12 sequential time steps. For high-fidelity training data, we coarsen 90 solution trajectories over a 256-cell grid by a factor of 8, with each trajectory consisting of 6 sequential time steps. As in the first example, the initial condition is a step function with heights randomly sampled from the range $[0.1, 1]$ and widths sampled from the range $[2.5, 3.5]$. The center of each step function is randomly selected from the entire domain of $[0, 2\pi]$. The stencil size is set to be 5, and the CFL number is 0.5. We choose the loss weights as $\lambda_1 = 0.5$ and $\lambda_2 = 1$.

Figure 4.7 shows three test samples plotted at various time instances during the forward integration. We run the trained models up to 20 time steps, which is greatly beyond the range of the training data, to assess their generalization capability. Our multifidelity model is able to accurately resolve such highly deformed solution structures that are not seen in the training data set. In contrast, both the high-fidelity and low-fidelity solvers fail to generalize effectively and produce irrelevant results.

Example 4.5. In this example, we simulate the 2D deformational flow problem proposed in [33], governed by the following transport equation,

$$u_t + (a(x,y,t)u)_x + (b(x,y,t)u)_y = 0, \quad (x,y) \in [0,1]^2, \tag{4.5}$$

with the velocity field being a periodic swirling flow,

$$\begin{aligned} a(x,y,t) &= \sin^2(\pi x)\sin(2\pi y)\cos(\pi t/T), \\ b(x,y,t) &= -\sin^2(\pi y)\sin(2\pi x)\cos(\pi t/T), \end{aligned} \tag{4.6}$$

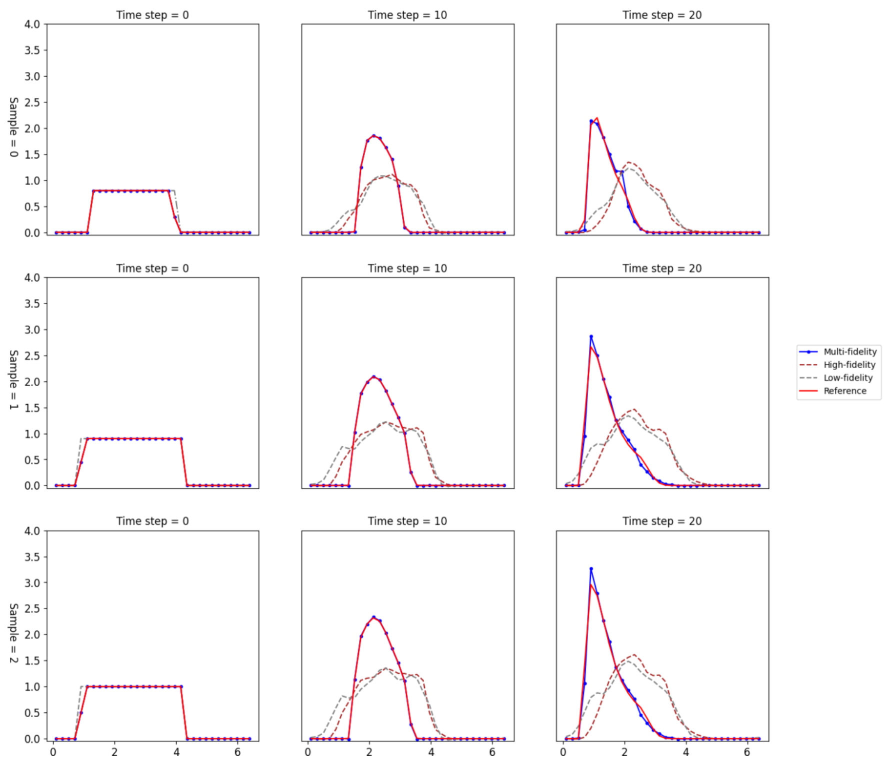


FIG. 4.7. *Numerical solutions of three test samples for the transport equation with a variable coefficient in Example* 4.4. $CFL = 0.5$.

where T is a constant. This is a widely recognized benchmark test for numerical transport solvers. It exhibits a distinct dynamics, in which the solution profile deforms over time as it follows the flow. The direction of the flow reverses at $t = T/2$, and the solution returns to its initial state at $t = T$, completing a full cycle of the evolution.

We set $T = 2$ and let the initial condition be a cosine bell centered at $[r_x, r_y]$:

$$\begin{aligned} u(x,y) &= \frac{1}{2}[1+\cos(\pi r)], \\ r(x,y) &= \min\left[1, 6\sqrt{(x-r_x)^2+(y-r_y)^2}\right]. \end{aligned} \tag{4.7}$$

We generate 4 trajectories using a high-resolution mesh of 256×256 cells, which are subsequently coarsened by a factor of 8 in each dimension to create the high-fidelity training data on the mesh of 32×32 cells. Each trajectory contains a sequence of 214 time steps from $t = 0$ to $t = T$. To obtain the low-fidelity training data, we generate 18 trajectories on the mesh of 32×32 cells. Each trajectory also contains a sequence of 214 time steps from $t = 0$ to $t = T$. The initial conditions for all trajectories are given by (4.7), where r_x and r_y are randomly sampled from the range $[0.25, 0.75]$. We set the CFL number to 1.8 and use a stencil of size 5×5. For the training setting, the loss weights are chosen as $\lambda_1 = 0.1$ and $\lambda_2 = 1$.

Although the training data consists of solution trajectories featuring a single bell, we demonstrate that the trained multifidelity model can generalize to simulate problems with an initial condition that contains two randomly placed bells centered at $[r_{1x}, r_{1y}]$ and $[r_{2x}, r_{2y}]$:

$$\begin{aligned} u(x,y) &= \frac{1}{2}[1+\cos(\pi r_1)+\cos(\pi r_2)], \\ r_1(x,y) &= \min\left[1, 6\sqrt{(x-r_{1x})^2+(y-r_{1y})^2}\right], \\ r_2(x,y) &= \min\left[1, 6\sqrt{(x-r_{2x})^2+(y-r_{2y})^2}\right]. \end{aligned} \tag{4.8}$$

In Figure 4.8, we show contour plots of numerical solutions to the initial condition (4.8) with $r_{1x} = 0.3, r_{1y} = 0.3, r_{2x} = 0.8, r_{2y} = 0.8$, computed by the proposed multifidelity

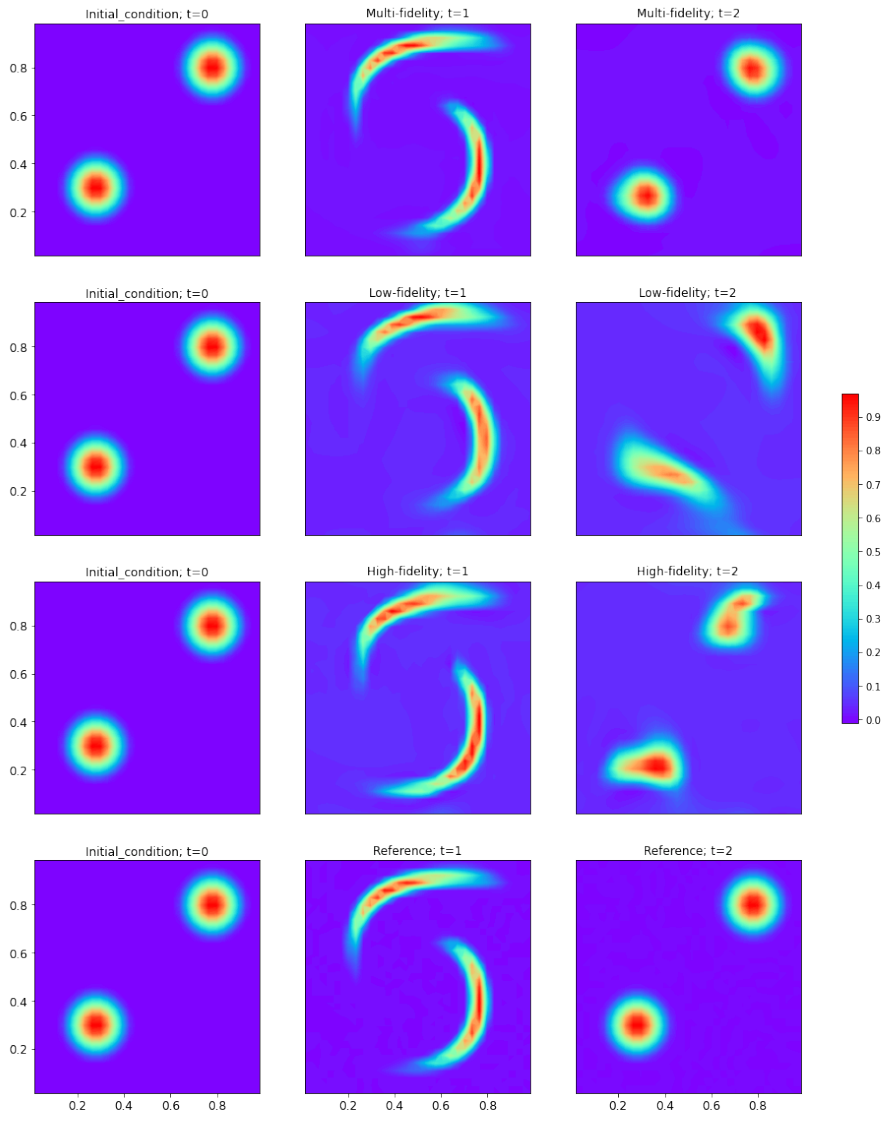


FIG. 4.8. *Contour plots of the numerical solutions for the 2D deformational flow at $t = 0,1,2$ in Example 4.5.*

TABLE 3

Run-time comparison of Example 4.5. *Run-time (seconds) measure for one period from $t = 0$ to $t = T$ on a GeForceRTX* 4070 *Ti GPU for the proposed multifidelity model and on a CPU for the traditional Eulerian RK WENO*5 *method using Python.*

Samples\method	Multifidelity(32×32)	WENO5(32×32)	WENO5(128×128)
Sample = 0	3.5734	1.3498	26.9169
Sample = 1	3.4712	1.3766	27.0657
Sample = 2	3.3923	1.3200	26.1897

TABLE 4

Mean square errors of Example 4.5.

Samples\method	Multifidelity(32×32)	WENO5(32×32)	WENO5(128×128)
Sample = 0	1.0249E-5	2.0531E-3	5.8772E-6
Sample = 1	1.8541E-5	1.6618E-3	5.1721E-6
Sample = 2	1.6547E-5	1.6547E-3	5.2394E-6

method, the high-fidelity model, and the low-fidelity model. The reference solution is generated with the same approach used to create the high-fidelity data. It is observed that the proposed method greatly surpasses single-fidelity models in capturing deformations and accurately recovering the initial profile at $t = T$.

Furthermore, we demonstrate the efficiency of the proposed method by providing the comparison of the error and run-time between the proposed multifidelity model and the traditional Eulerian RK WENO5 method. In Table 3 we provide the run-time for simulating three test samples up to $t = T$ using the multifidelity model with a mesh resoluton of 32×32 cells and CFL number of 1.8, as well as the RK WENO5 with mesh resolutions of 32×32 cells and 128×128 cells, both using the CFL number of 0.6. Table 4 reports the corresponding mean square errors. The computational time of the proposed multifidelity model is slightly higher than that of the RK WENO5 method with the same mesh resolution of 32×32 cells. However, it is significantly lower than the RK WENO5 over a finer mesh of 128×128 cells. Meanwhile, the multifidelity model achieves much smaller errors compared to the RK WENO5 method with the same mesh size and only slightly larger errors than the RK WENO5 method using the high-resolution mesh.

4.2. Nonlinear Vlasov–Possion system.

In this subsection, we present the numerical results for simulating the nonlinear 1D1V VP system. We demonstrate the efficiency and accuracy of the proposed multifidelity SL FV method coupled with the second-order RKEI by comparing it to single-fidelity models and the traditional SL FV WENO5 scheme. Additionally, we numerically verify the advantage of the second-order RKEI over the first-order scheme. The training data and reference solutions are generated by the fourth-order conservative SL FV WENO scheme [63].

Example 4.6. In this example, we consider the Landau damping with the initial condition

$$f(x, v, t = 0) = \frac{1}{\sqrt{2\pi}}(1 + \alpha\cos(kx))\exp\left(-\frac{v^2}{2}\right), \quad x \in [0, L], \quad v \in [-V_c, V_c], \tag{4.9}$$

where $k = 0.5$, $L = 4\pi$, and $V_c = 2\pi$.

We generate 6 solution trajectories with a 32×64 grid to obtain the low-fidelity training data. The high-fidelity training data is generated by coarsening 2 high-resolution solution trajectories on a 256×512-cell grid by a factor of 8 in each

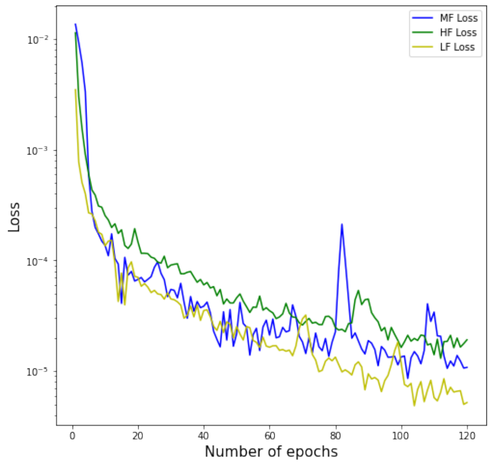


FIG. 4.9. *The training loss for three different methods in Example* 4.6.

dimension. The initial conditions for both the low- and high-fidelity training data sets are determined using (4.9), with α randomly selected from a uniform distribution in the range $[0.05, 0.45]$. We set the stencil size to be 5×5 and maintain a chosen CFL number of 1.8. For the training setting, the loss weights are chosen as $\lambda_1 = 0.1$ and $\lambda_2 = 1$. For the purpose of comparison, the reference solution is generated with the same approach used to create the high-fidelity data. The training loss for three different methods during their respective training is shown in Figure 4.9.

For testing, we set $\alpha = 0.5$, yielding the strong Landau damping, which lies outside the range of the training data. Figure 4.10 presents contour plots of the numerical solutions computed by our multifidelity method with the second-order RKEI, high-fidelity solver, low-fidelity solver, multifidelity scheme with the first-order RKEI, and traditional SL FV WENO method, all implemented on the same mesh with 32×64 cells. It can be observed that the proposed method with the second-order RKEI captures the filamentation structure, while both the single-fidelity models generate inaccurate results. In addition, the multifidelity scheme coupled with the first-order RKEI fails to capture the fine-scale structures of interest. This limitation arises due to the low-order accuracy in time, which restricts its ability to generalize effectively. The traditional SL FV WENO scheme produces reasonable results but exhibits smeared solution structures, primarily due to the low mesh resolution. In Figure 4.11, we plot the time histories of the electric energy for each approach. Our method with the second-order RKEI yields results that agree well with the reference solution, significantly outperforming all other methods considered in the comparison.

Last, we apply the trained model to simulate the weak Landau damping with $\alpha = 0.05$. In Figure 4.12, we present the time histories of the electric energy of the solutions by our multifidelity solver as well as the SL FV WENO method for comparison. For the SL FV WENO method based on the 32×64 grid, the recurrence occurs around the theoretically predicted time $T_R = 64$ (see [22]). Meanwhile, our multifidelity solver with the same mesh resolution produces a result that is consistent with the reference solution, and the recurrence effect is greatly mitigated.

Example 4.7. In this example, we simulate the symmetric two stream instability with the initial condition

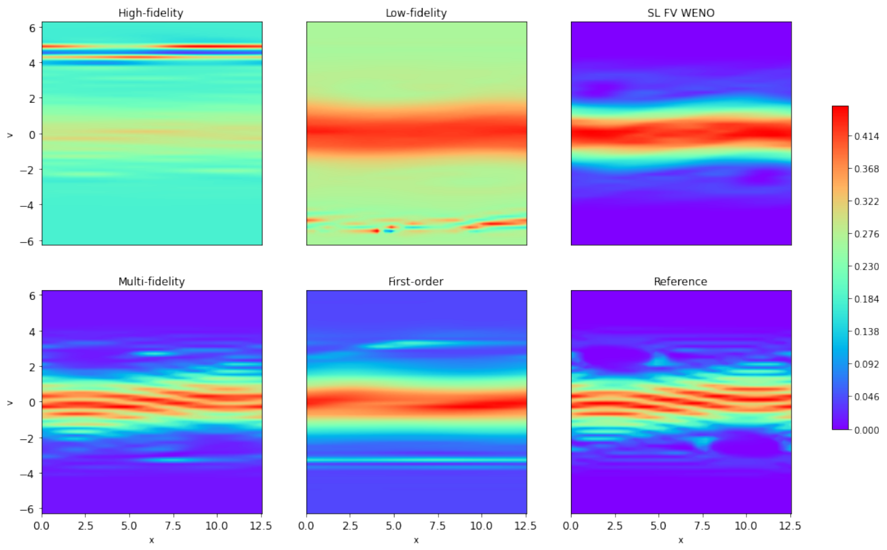


FIG. 4.10. *Contour plots of numerical solutions of the strong Landau damping at $t = 40$ in Example 4.6 with $\alpha = 0.5$. "Multifidelity" denotes our multifidelity method coupled with the second-order RKEI. "First-order" denotes our multifidelity method with the first-order RKEI.*

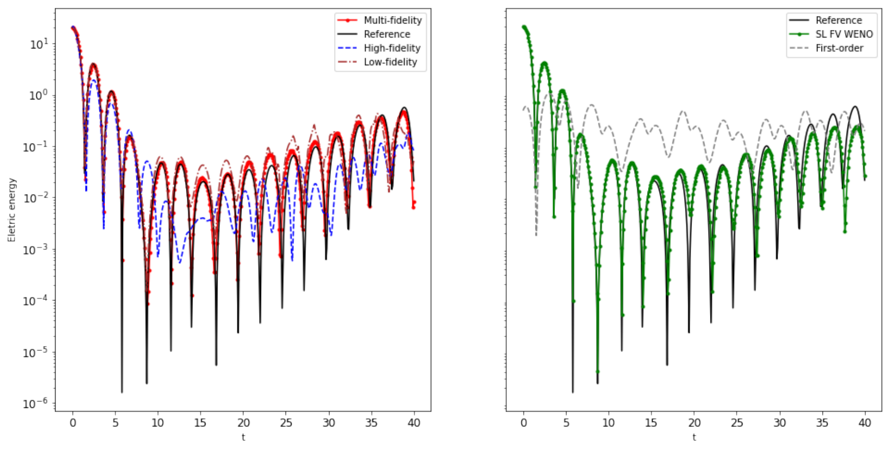


FIG. 4.11. *Time histories of the electric energy of the strong Landau damping in Example 4.6 with $\alpha = 0.5$.*

(4.10)
$$f(x,v,t=0) = \frac{1}{\sqrt{2\pi}}(1+\alpha\cos(kx))v^2\exp\left(-\frac{v^2}{2}\right), \quad x \in [0,L], \quad v \in [-V_c, V_c],$$

where $k = 0.5$, $L = 4\pi$, and $V_c = 2\pi$.

We generate the low-fidelity training data by obtaining five solution trajectories over a 32×64 grid. To generate the high-fidelity training data, we coarsen two high-resolution solution trajectories over a 256×512-cell grid by a factor of 8 in each dimension. The initial conditions are determined using (4.10), with α randomly sampled from a uniform distribution in the range $[0.01, 0.05]$. We use a CFL number

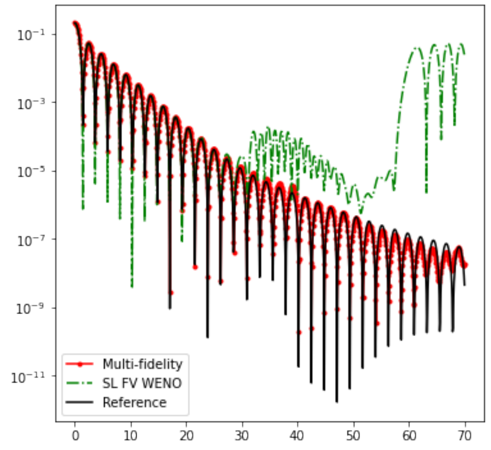


FIG. 4.12. *Time histories of the electric energy of the weak Landau damping in Example* 4.6 *with* $\alpha = 0.05$.

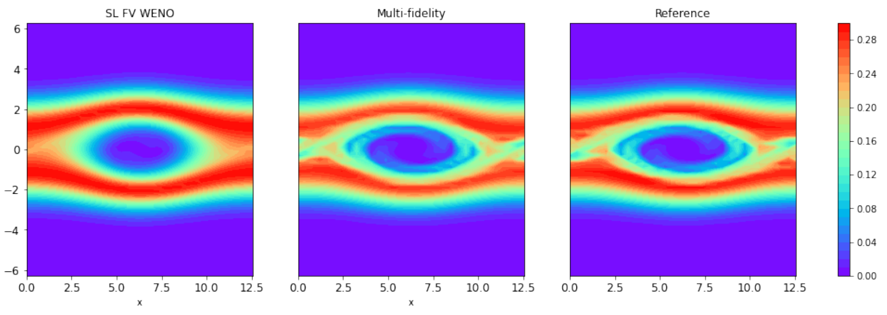


FIG. 4.13. *Contour plots of the numerical solutions for the two stream instability at* $t = 53$ *in Example* 4.7 *with* $\alpha = 0.01$.

of 1.8 and set the stencil size to be 5×5. We choose the loss weights as $\lambda_1 = 0.2$ and $\lambda_2 = 1$. The reference solution is generated with the same approach used to create the high-fidelity data.

For testing, we report contour plots of the two stream instability with $\alpha = 0.01$ at $T = 53$ in Figure 4.13 for the multifidelity method as well as the SL FV WENO scheme for comparison. The proposed multifidelity method produces high-quality numerical results that agree well with the reference solution, similarly to the previous example. However, the SL FV WENO scheme fails to capture fine-scale structures of interest, such as the roll-up at the center of the solution. We also present the absolute error between the numerical solutions and the reference solution in Figure 4.14.

Example 4.8. In this example, we consider another two stream instability with the following initial condition:

$$\begin{aligned} & f(x,v,t=0) \\ (4.11)\quad & = \frac{2}{7\sqrt{2\pi}}(1+5v^2)(1+\alpha_1\cos(kx)+\alpha_2\cos(2kx)+\alpha_3\cos(3kx))\exp\left(-\frac{v^2}{2}\right), \\ & x \in [0,L], v \in [-V_c, V_c], \end{aligned}$$

where $k = 0.5$, $L = 4\pi$, and $V_c = 2\pi$.

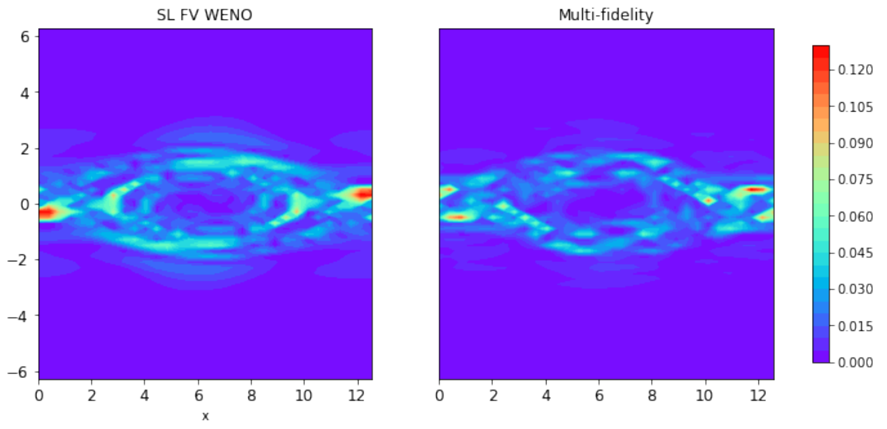


FIG. 4.14. *The error of the numerical solutions for the two stream instability at $t = 53$ in Example 4.7 with $\alpha = 0.01$.*

This example is more challenging than the previous two examples, as the initial condition is defined by perturbing the first three Fourier modes of a equilibrium with magnitudes α_1, α_2, and α_3, respectively. We generate the low-fidelity training data by obtaining 6 solution trajectories on a 32×64 grid. For the high-fidelity data, we downsample two high-resolution trajectories on a 256×512-cell grid by a factor of 8 in each dimension. The initial conditions for the training data sets are determined using (4.11) with α_1, α_2, and α_3 randomly selected from a uniform distribution in the range $[0.01, 0.02]$. For comparison, the reference solution is generated with the same approach used to create the high-fidelity data. We use a CFL number of 1.8 and a stencil size of 5×5. λ_1 and λ_2 in (3.2) are chosen as 0.1 and 1, respectively.

During testing, we consider the initial condition with $\alpha_1 = 0.01, \alpha_2 = 0.01/1.2, \alpha_3 = 0.01/1.2$, which is a widely used benchmark configuration in the literature. Note that such a parameter choice is outside the range of the training data. In Figure 4.15, we present contour plots of the numerical solutions computed by our multifidelity method and the SL FV WENO scheme. It is observed that the results by our method qualitatively agree with the reference solution, effectively capturing the underlying fine-scale structures of interest. The SL FV WENO scheme, on the other hand, can provide reasonable results but tends to smear out the small-scale structures in the solution. This demonstrates that the proposed multifidelity model is capable of producing results with reasonable accuracy, highlighting its generalization capabilities. Figure 4.16 presents the errors between the numerical solutions and the reference solution.

5. Conclusion.

In this paper, we have proposed a novel multifidelity ML-based SL FV scheme for solving transport equations. This method is specifically designed for scenarios where there is an abundance of low-fidelity data and a limited amount of high-fidelity data. By using a composite NN architecture, our method can effectively approximate the inherent correlation between the high-fidelity and low-fidelity data. Numerical experiments conducted in this study show that the proposed method achieves improved stability and accuracy compared to networks trained solely on either low-fidelity data or high-fidelity data. This indicates that the multifidelity approach enhances the performance and capabilities of the SL FV scheme for solving

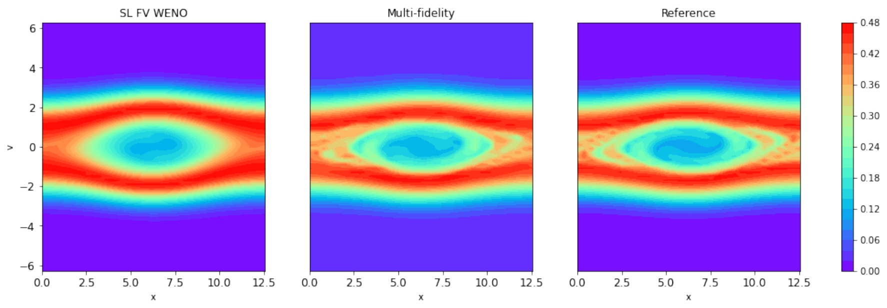


FIG. 4.15. *Contour plots of the numerical solutions of the two stream instability at $t = 53$ in Example* 4.8 *with* $\alpha_1 = 0.01, \alpha_2 = 0.01/1.2, \alpha_3 = 0.01/1.2$.

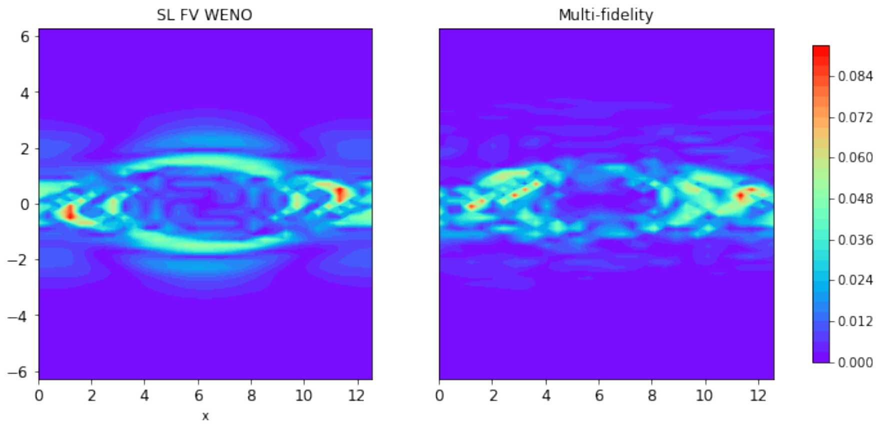


FIG. 4.16. *The error of the numerical solutions of the two stream instability at $t = 53$ in Example* 4.8 *with* $\alpha = 0.01$.

transport equations. Furthermore, we have extended this multifidelity method to the simulation of nonlinear VP systems, coupled with the high-order RKEI. This extension allows for accurate and efficient simulations of complex physical phenomena for this multifidelity approach. Future work includes exploring the possibility of applying the method to more complicated systems, further improving the generalization capabilities of the method, investigating the use of graph NNs for accommodating unstructured meshes, and addressing challenges related to adaptivity and complex geometries, among other potential research directions.

REFERENCES

[1] M. Aliakbari, M. Mahmoudi, P. Vadasz, and A. Arzani, *Predicting high-fidelity multiphysics data from low-fidelity fluid flow and transport solvers using physics-informed neural networks*, Int. J. Heat Fluid Flow, 96 (2022), 109002, https://doi.org/10.1016/j.ijheatfluidflow.2022.109002.

[2] M. Ashouri and A. Hashemi, *A transfer learning metamodel using artificial neural networks for natural convection flows in enclosures*, Case Stud. Thermal Eng., 36 (2022), 102179, https://doi.org/10.1016/j.csite.2022.102179.

[3] Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. P. Brenner, *Learning data-driven discretizations for partial differential equations*, Proc. Natl. Acad. Sci. USA, 116 (2019), pp. 15344–15349, https://doi.org/10.1073/pnas.1814058116.

[4] S. Basir and I. Senocak, *Physics and equality constrained artificial neural networks: Application to forward and inverse problems with multi-fidelity data fusion*, J. Comput. Phys., 463 (2022), 111301, https://doi.org/10.1016/j.jcp.2022.111301.

[5] K. Bhattacharya, B. Hosseini, N. B. Kovachki, and A. M. Stuart, *Model reduction and neural networks for parametric PDEs*, SMAI J. Comput. Math., 7 (2021), pp. 121–157, https://doi.org/10.5802/smai-jcm.74.

[6] J. Brandstetter, D. Worrall, and M. Welling, *Message passing neural PDE solvers*, in International Conference on Learning Representations, 2022.

[7] X. Cai, S. Boscarino, and J.-M. Qiu, *High order semi-Lagrangian discontinuous Galerkin method coupled with Runge-Kutta exponential integrators for nonlinear Vlasov dynamics*, J. Comput. Phys., 427 (2021), 110036, https://doi.org/10.1016/j.jcp.2020.110036.

[8] E. Celledoni, A. Marthinsen, and B. Owren, *Commutator-free lie group methods*, Future Generation Comput. Syst., 19 (2003), pp. 341–352.

[9] S. Chakraborty, *Transfer learning based multi-fidelity physics informed deep neural network*, J. Comput. Phys., 426 (2021), 109942, https://doi.org/10.1016/j.jcp.2020.109942.

[10] J. Chen, Y. Gao, and Y. Liu, *Multi-fidelity data aggregation using convolutional neural networks*, Comput. Methods Appl. Mech. Engrg., 391 (2022), 114490, https://doi.org/10.1016/j.cma.2021.114490.

[11] W. Chen and P. Stinis, *Feature-adjacent multi-fidelity physics-informed machine learning for partial differential equations*, J. Comput. Phys., 498 (2024), 112683.

[12] X. Chen, C. Gong, Q. Wan, L. Deng, Y. Wan, Y. Liu, B. Chen, and J. Liu, *Transfer learning for deep neural network-based partial differential equations solving*, Adv. Aerodyn., 3 (2021), 36, https://doi.org/10.1186/s42774-021-00094-7.

[13] Y. Chen, W. Guo, and X. Zhong, *A learned conservative semi-Lagrangian finite volume scheme for transport simulations*, J. Comput. Phys., 490 (2023), 112329, https://doi.org/10.1016/j.jcp.2023.112329.

[14] Y. Chen, J. Yan, and X. Zhong, *Cell-average based neural network method for third order and fifth order KdV type equations*, Front. Appl. Math. Statist., 8 (2022), https://doi.org/10.3389/fams.2022.1021069.

[15] C.-Z. Cheng and G. Knorr, *The integration of the Vlasov equation in configuration space*, J. Comput. Phys., 22 (1976), pp. 330–351.

[16] M. Crawshaw, *Multi-Task Learning with Deep Neural Networks: A Survey*, preprint, arXiv:2009.09796, 2020.

[17] S. De, J. Britton, M. Reynolds, R. Skinner, K. Jansen, and A. Doostan, *On transfer learning of neural networks using bi-fidelity data for uncertainty propagation*, Int. J. Uncertain. Quantif., 10 (2020), pp. 543–573, https://doi.org/10.1615/Int.J.UncertaintyQuantification.2020033267.

[18] S. De and A. Doostan, *Neural network training using ℓ_1-regularization and bi-fidelity data*, J. Comput. Phys., 458 (2022), 111010, https://doi.org/10.1016/j.jcp.2022.111010.

[19] S. De, M. Reynolds, M. Hassanaly, R. N. King, and A. Doostan, *Bi-fidelity modeling of uncertain and partially unknown systems using DeepONets*, Comput. Mech., 71 (2023), pp. 1251–1267, https://doi.org/10.1007/s00466-023-02272-4.

[20] C. Erath, P. H. Lauritzen, and H. M. Tufo, *On mass conservation in high-order high-resolution rigorous remapping schemes on the sphere*, Monthly Weather Rev., 141 (2013), pp. 2128–2133.

[21] M. G. Fernández-Godino, *Review of multi-fidelity models*, Adv. Comput. Sci. Eng., 1 (2023), pp. 351–400.

[22] F. Filbet, E. Sonnendrücker, and P. Bertrand, *Conservative numerical schemes for the Vlasov equation*, J. Comput. Phys., 172 (2001), pp. 166–187, https://doi.org/10.1006/jcph.2001.6818.

[23] S. Gottlieb, C.-W. Shu, and E. Tadmor, *Strong stability-preserving high-order time discretization methods*, SIAM Rev., 43 (2001), pp. 89–112.

[24] D. Greenfeld, M. Galun, R. Basri, I. Yavneh, and R. Kimmel, *Learning to optimize multigrid PDE solvers*, in Proceedings of the International Conference on Machine Learning, 2019, pp. 2415–2423.

[25] M. Guo, A. Manzoni, M. Amendt, P. Conti, and J. S. Hesthaven, *Multi-fidelity regression using artificial neural networks: Efficient approximation of parameter-dependent output quantities*, Comput. Methods Appl. Mech. Engrg., 389 (2022), 114378, https://doi.org/10.1016/j.cma.2021.114378.

[26] A. A. Howard, M. Perego, G. E. Karniadakis, and P. Stinis, *Multifidelity deep operator networks for data-driven and physics-informed problems*, J. Comput. Phys., 493 (2023), 112462.

[27] J.-T. Hsieh, S. Zhao, S. Eismann, L. Mirabella, and S. Ermon, *Learning neural PDE solvers with convergence guarantees*, in Proceedings of the International Conference on Learning Representations, 2019; also available online from https://openreview.net/forum?id=rklaWn0qK7.

[28] G.-S. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.

[29] S. Jiang and L. J. Durlofsky, *Use of multifidelity training data and transfer learning for efficient construction of subsurface flow surrogate models*, J. Comput. Phys., 474 (2023), 111800, https://doi.org/10.1016/j.jcp.2022.111800.

[30] G. Kissas, J. H. Seidman, L. F. Guilhoto, V. M. Preciado, G. J. Pappas, and P. Perdikaris, *Learning operators with coupled attention*, J. Mach. Learn. Res., 23 (2022), pp. 1–63.

[31] D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer, *Machine learning–accelerated computational fluid dynamics*, Proc. Natl. Acad. Sci. USA, 118 (2021), e2101784118.

[32] P. H. Lauritzen, R. D. Nair, and P. A. Ullrich, *A conservative semi-Lagrangian multi-tracer transport scheme (CSLAM) on the cubed-sphere grid*, J. Comput. Phys., 229 (2010), pp. 1401–1424.

[33] R. J. LeVeque, *High-resolution conservative algorithms for advection in incompressible flow*, SIAM J. Numer. Anal., 33 (1996), pp. 627–665, https://doi.org/10.1137/0733033.

[34] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. M. Stuart, and A. Anandkumar, *Fourier neural operator for parametric partial differential equations*, in Proceedings of the International Conference on Learning Representations, 2020, https://arxiv.org/abs/2010.08895.

[35] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. M. Stuart, and A. Anandkumar, *Neural operator: Graph kernel network for partial differential equations*, in Proceedings of ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations, 2020; also available online from https://arxiv.org/abs/2003.03485.

[36] Z. Li, S. Zhang, H. Li, K. Tian, Z. Cheng, Y. Chen, and B. Wang, *On-line transfer learning for multi-fidelity data fusion with ensemble of deep neural networks*, Adv. Eng. Inform., 53 (2022), 101689, https://doi.org/10.1016/j.aei.2022.101689.

[37] D. Liu and Y. Wang, *Multi-fidelity physics-constrained neural network and its application in materials modeling*, J. Mech. Design, 141 (2019), 121403, https://doi.org/10.1115/1.4044400.

[38] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, in Proceedings of the International Conference on Learning Representations, 2018.

[39] 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., 3 (2021), pp. 218–229, https://doi.org/10.1038/s42256-021-00302-5.

[40] L. Lu, R. Pestourie, S. G. Johnson, and G. Romano, *Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport*, Phys. Rev. Res., 4 (2022), 023210, https://doi.org/10.1103/PhysRevResearch.4.023210.

[41] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, *Physics-informed neural networks with hard constraints for inverse design*, SIAM J. Sci. Comput., 43 (2021), pp. B1105–B1132, https://doi.org/10.1137/21M1397908.

[42] X. Meng, H. Babaee, and G. E. Karniadakis, *Multi-fidelity Bayesian neural networks: Algorithms and applications*, J. Comput. Phys., 438 (2021), 110361, https://doi.org/10.1016/j.jcp.2021.110361.

[43] X. Meng and G. E. Karniadakis, *A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems*, J. Comput. Phys., 401 (2020), 109020, https://doi.org/10.1016/j.jcp.2019.109020.

[44] M. Motamed, *A multi-fidelity neural network surrogate sampling method for uncertainty quantification*, Int. J. Uncertain. Quantif., 10 (2020), pp. 315–332, https://doi.org/10.1615/Int.J.UncertaintyQuantification.2020031957.

[45] O. Obiols-Sales, A. Vishnu, N. Malaya, and A. Chandramowliswharan, *CFDNet: A deep learning-based accelerator for fluid simulations*, in Proceedings of the 34th ACM International Conference on Supercomputing, ACM, 2020, pp. 1–12, https://doi.org/10.1145/3392717.3392772.

[46] J. Pathak, M. Mustafa, K. Kashinath, E. Motheau, T. Kurth, and M. Day, *Using Machine Learning to Augment Coarse-Grid Computational Fluid Dynamics Simulations*, preprint, arXiv:2010.00072, 2020.

[47] B. Peherstorfer, K. Willcox, and M. Gunzburger, *Survey of multifidelity methods in uncertainty propagation, inference, and optimization*, SIAM Rev., 60 (2018), pp. 550–591, https://doi.org/10.1137/16M1082469.

[48] P. Perdikaris, M. Raissi, A. Damianou, N. D. Lawrence, and G. E. Karniadakis, *Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling*, Proc. A, 473 (2017), 20160751.

[49] J.-M. Qiu and C.-W. Shu, *Conservative semi-lagrangian finite difference WENO dormulations with applications to the Vlasov equation*, Commun. Comput. Phys., 10 (2011), pp. 979–1000, https://doi.org/10.4208/cicp.180210.251110a.

[50] M. Raissi and G. Karniadakis, *Deep Multi-Fidelity Gaussian Processes*, preprint, arXiv:1604.07484, 2016.

[51] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics Informed Deep Learning (Part* I*): Data-Driven Solutions of Nonlinear Partial Differential Equations*, 2017, http://arxiv.org/abs/1711.10561.

[52] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics Informed Deep Learning (Part* II*): Data-Driven Discovery of Nonlinear Partial Differential Equations*, 2017, http://arxiv.org/abs/1711.10566.

[53] M. Ramezankhani, A. Nazemi, A. Narayan, H. Voggenreiter, M. Harandi, R. Seethaler, and A. S. Milani, *A data-driven multi-fidelity physics-informed learning framework for smart manufacturing: A composites processing case study*, in Proceedings of the 5th International Conference on Industrial Cyber-Physical Systems (ICPS), IEEE, 2022, https://doi.org/10.1109/ICPS51978.2022.9816983.

[54] J. Sirignano, J. F. MacArt, and J. B. Freund, *DPM: A deep learning PDE augmentation method with application to large-eddy simulation*, J. Comput. Phys., 423 (2020), 109811, https://doi.org/10.1016/j.jcp.2020.109811.

[55] D. H. Song and D. M. Tartakovsky, *Transfer learning on multifidelity data*, J. Mach. Learn. Model. Comput., 3 (2022), pp. 31–47, https://doi.org/10.1615/JMachLearnModelComput.2021038925.

[56] E. Sonnendrücker, J. Roche, P. Bertrand, and A. Ghizzo, *The semi-Lagrangian method for the numerical resolution of the Vlasov equation*, J. Comput. Phys., 149 (1999), pp. 201–220.

[57] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin, *Accelerating Eulerian fluid simulation with convolutional networks*, in Proceedings of the International Conference on Machine Learning, 2017, pp. 3424–3433.

[58] N. Trask, R. G. Patel, B. J. Gross, and P. J. Atzberger, *GMLS-Nets: A Framework for Learning From Unstructured Data*, http://arxiv.org/abs/1909.05371, 2019.

[59] K. Um, R. Brand, Y. R. Fei, P. Holl, and N. Thuerey, *Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers*, Adv. Neural Inf. Process. Syst., 33 (2020), pp. 6111–6122.

[60] K. Weiss, T. M. Khoshgoftaar, and D. Wang, *A survey of transfer learning*, J. Big Data, 3 (2016), 9, https://doi.org/10.1186/s40537-016-0043-6.

[61] Z. Xiang, W. Peng, X. Liu, and W. Yao, *Self-adaptive loss balanced physics-informed neural networks*, Neurocomputing, 496 (2022), pp. 11–34, https://doi.org/10.1016/j.neucom.2022.05.015.

[62] J. Yu, L. Lu, X. Meng, and G. E. Karniadakis, *Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems*, Comput. Methods Appl. Mech. Engrg., 393 (2022), 114823, https://doi.org/10.1016/j.cma.2022.114823.

[63] N. Zheng, X. Cai, J.-M. Qiu, and J. Qiu, *A fourth-order conservative semi-Lagrangian finite volume WENO scheme without operator splitting for kinetic and fluid simulations*, Comput. Methods Appl. Mech. Engrg., 395 (2022), 114973.

[64] J. Zhuang, D. Kochkov, Y. Bar-Sinai, M. P. Brenner, and S. Hoyer, *Learned discretizations for passive scalar advection in a two-dimensional turbulent flow*, Phys. Rev. Fluids, 6 (2021), 064605.